

# TM4C123GH6PM Micro-controllers Programming Concepts

Dr. Munesh Singh

Indian Institute of Information Technology  
Design and Manufacturing,  
Kancheepuram  
Chennai-600127

January 31, 2020

# IAR Workbench Toolset

TM4C123GH6P  
Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## ARM Microcontroller

The screenshot displays the IAR Embedded Workbench IDE interface for an ARM microcontroller project. The main window is divided into several panes:

- Workspace:** Shows the project structure with files like `main.c`, `main.h`, `debug.c`, and `debug.h`.
- Debug:** Displays the current state of the program, including the `main` function and the `int main()` function.
- Registers:** A table showing the current CPU registers and their values. The `R12` register is highlighted with a red box, showing the value `0x00000000`.
- Disassembly:** Shows the assembly code for the `main` function, including instructions like `int count=0;`, `main:`, `++count;`, `exit:`, and `_exit:`.
- Memory:** A table showing the memory contents, with a red box highlighting the address `0x00000000` and the value `0x00000000`.
- Log:** Displays the debug log, showing messages like "Mon Feb 11, 2019 11:50:04 IAR Embedded Workbench 8.32.1 (C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.32\iar\bin\iarmpc.dll)" and "Mon Feb 11, 2019 11:50:04 Loaded target file: C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.32\iar\bin\debugger\Tweakshim\TM4C123.dmc".

The bottom status bar shows the system information: "Ready", "Type here to search", "psa 00000001", "System: CAP MEM OVR", and "11:56 AM 2/11/2019".

# IAR Workbench Toolset

TM4C123GH6P  
Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## Local versus Global Variable

The screenshot displays the IAR Embedded Workbench IDE interface. The main window shows a C program with a local variable 'count' defined inside a function 'main'. The program is as follows:

```
int count=0;
int main()
{
    while(count<21)
    {
        ++count;
    }
    return 0;
}
```

The 'Registers' window shows the current CPU registers, with R0 through R12, APSR, IPSR, EPSR, and PC. The 'Memory' window shows the memory address 0x20000000, which is highlighted in red. The 'Locals' window shows the local variable 'count' at address 0x20000000. The 'Disassembly' window shows the assembly code for the program, with the instruction 'ldr r0, #0' highlighted in green.

Log

```
Mon Feb 11, 2019 12:11:01: IAR Embedded Workbench 8.32.1 (C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.32\arm\bin\iarproc.dll)
Mon Feb 11, 2019 12:11:01: Loaded macro file C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.32\arm\config\debugger\Tosadstruments\TM4C123.dmc
Mon Feb 11, 2019 12:11:01: Download complete
Mon Feb 11, 2019 12:11:01: Loaded debugger C:\IAR\Debug\Exe\Lesson0.out
Mon Feb 11, 2019 12:11:01: Target reset
```

Build Debug Log

Ready

Type here to search

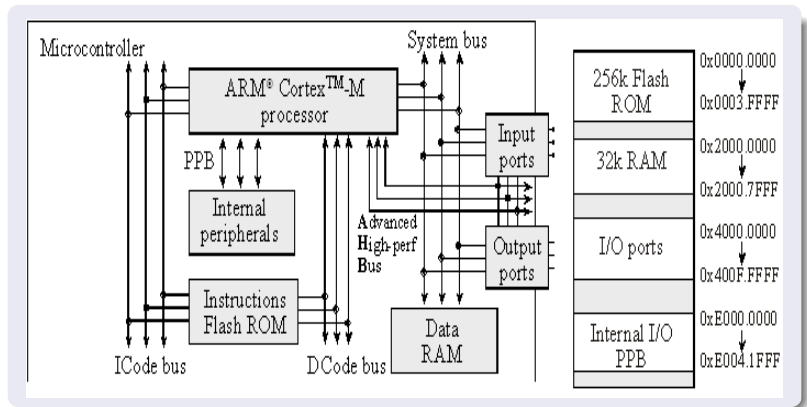
Ln4, Col3 System C:\IAR\Out 12:11 PM 2/11/2019

# Memory Map Buses

TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh



# Memory Map I/O Ports

TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

256k Flash ROM	0x0000.0000 ↓ 0x0003.FFFF
32k RAM	0x2000.0000 ↓ 0x2000.7FFF
I/O ports	0x4000.0000 ↓ 0x400F.FFFF
Internal I/O PPB	0xE000.0000 ↓ 0xE004.1FFF

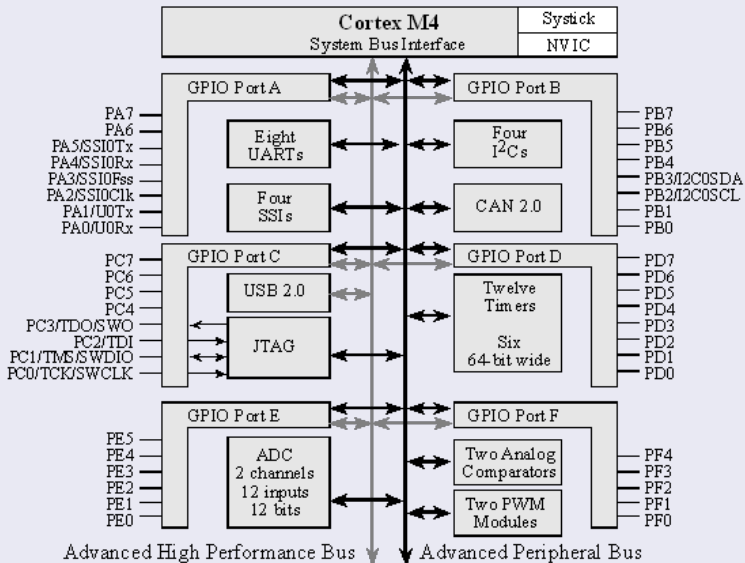
0x4000.4000	0x4000.4FFF	GPIO Port A
0x4000.5000	0x4000.5FFF	GPIO Port B
0x4000.6000	0x4000.6FFF	GPIO Port C
0x4000.7000	0x4000.7FFF	GPIO Port D
0x4002.4000	0x4002.4FFF	GPIO Port E
0x4002.5000	0x4002.5FFF	GPIO Port F

# Classification of I/O Ports

TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh



# I/O Port Initialization

TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## Seven Steps to Initialize a Port

- General-Purpose Input/Output Sleep Mode Clock Gating Control (SCGCGPIO)
- **Base** 0x400F.E000
- **Offset** 0x608
- Type RW, reset 0x0000.0000
- Actual address of this register calculated by adding the base address with offset
- Actual address= 0x400FE608

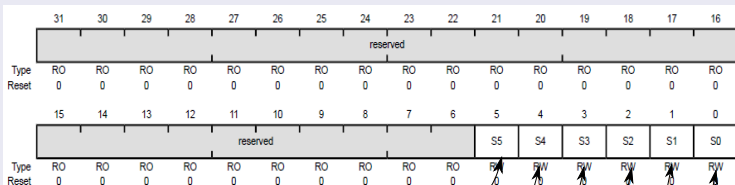
# SCGCGPIO Register

TM4C123GH6P  
Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## Register Information

- Activate the clock gating register



0---> disable

1---> enable

Port F

Port E

Port D

Port C

Port B

Port A



# Debugger Mode Memory View SCGCPIO Register and F Port

TM4C123GH6P  
Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## F Port Before Setting bit 5 of SCGCPIO register

### Symbolic Memory

Go to:	0x400FE608	▼	Memory	▼	Previous	Next
Location	Data	Variable	Value	Type		
0x400FE5FC	0x00000000					
0x400FE600	0x00000000					
0x400FE604	0x00000000					
0x400FE608	0x00000000					
0x400FE60C	0x00000000					
0x400FE610	0x00000000					
0x400FE614	0x00000001					

### Memory 1

Go to	0x40025000	▼	Memory	▼	▼	+	+
0x40024f90	----	----	----	----	----	----	----
0x40024fa0	----	----	----	----	----	----	----
0x40024fb0	----	----	----	----	----	----	----
0x40024fc0	----	----	----	----	----	----	----
0x40024fd0	----	----	----	----	----	----	----
0x40024fe0	----	----	----	----	----	----	----
0x40024ff0	----	----	----	----	----	----	----
0x40025000	----	----	----	----	----	----	----
0x40025010	----	----	----	----	----	----	----

# Dibugger Mode Memeory View SCGCGPIO Register and F Port

TM4C123GH6P  
Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## F Port after Setting bit 5 of SCGCGPIO register

### Symbolic Memory

Go to:	0x400FE608	Memory	Previous	Next
Location	Data	Variable	Value	Type
0x400FE5FC	0x00000000			
0x400FE600	0x00000000			
0x400FE604	0x00000000			
0x400FE608	0x00000020			
0x400FE60C	0x00000000			
0x400FE610	0x00000000			
0x400FE614	0x00000001			

### Memory 1

Go to	0x40025000	Memory			
0x40024f90	----	----	----	----	----
0x40024fa0	----	----	----	----	----
0x40024fb0	----	----	----	----	----
0x40024fc0	----	----	----	----	----
0x40024fd0	----	----	----	----	----
0x40024fe0	----	----	----	----	----
0x40024ff0	----	----	----	----	----
0x40025000	0000	0000	0000	0000	0000
0x40025010	0000	0000	0000	0000	0000

# GPIODIR Direction Register

TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## Register Information

	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

0---> Cooresponding pin input

1---> Cooresponding pin output

):7 Bits

### GPIO Direction (GPIODIR)

GPIO Port A (APB) base: 0x4000.4000  
GPIO Port A (AHB) base: 0x4005.8000  
GPIO Port B (APB) base: 0x4000.5000  
GPIO Port B (AHB) base: 0x4005.9000  
GPIO Port C (APB) base: 0x4000.6000  
GPIO Port C (AHB) base: 0x4005.A000  
GPIO Port D (APB) base: 0x4000.7000  
GPIO Port D (AHB) base: 0x4005.B000  
GPIO Port E (APB) base: 0x4002.4000  
GPIO Port E (AHB) base: 0x4005.C000  
GPIO Port F (APB) base: 0x4002.5000  
GPIO Port F (AHB) base: 0x4005.D000  
Offset 0x400  
Type RW, reset 0x0000.0000

# GPIODEN Digital Function Register

TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## Register Information

	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

0---> Disable Digital function

1---> Enable Digital function

0:7 Bits

## GPIODEN Digital Enable (GPIODEN)

GPIO Port A (APB) base: 0x4000.4000

GPIO Port A (AHB) base: 0x4005.8000

GPIO Port B (APB) base: 0x4000.5000

GPIO Port B (AHB) base: 0x4005.9000

GPIO Port C (APB) base: 0x4000.6000

GPIO Port C (AHB) base: 0x4005.A000

GPIO Port D (APB) base: 0x4000.7000

GPIO Port D (AHB) base: 0x4005.B000

GPIO Port E (APB) base: 0x4002.4000

GPIO Port E (AHB) base: 0x4005.C000

GPIO Port F (APB) base: 0x4002.5000

GPIO Port F (AHB) base: 0x4005.D000

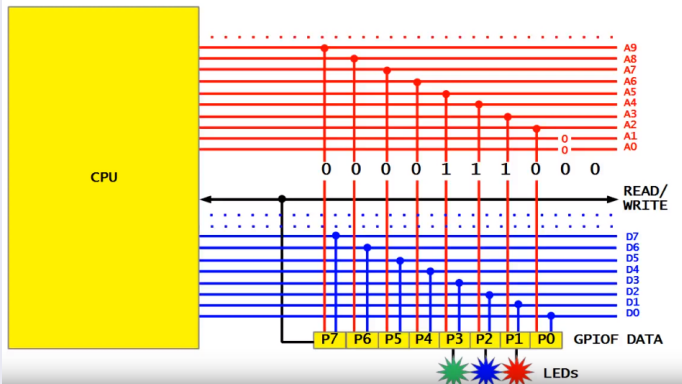
Offset 0x51C

Type RW, reset -

## GPIODATA Register

## Register Information

- In order to write to GPIODATA, the corresponding bits in the mask, resulting from the address bus bits [9:2], must be set.



# GPIODATA Register

TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## Register Information

	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

0:7 Bits

2 1 8 4 2 1 8 4 2 1  
9 8 7 6 5 4 3 2 1 0 Addr  
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓  
7 6 5 4 3 2 1 0 Pin

All seven port if want to use

3FC

0x400253FC

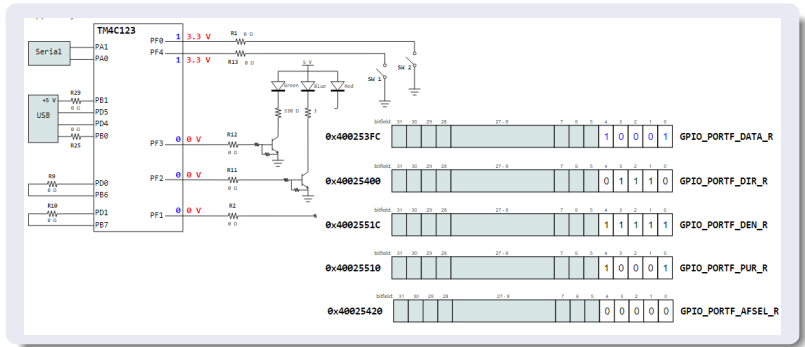
## GPIO Data (GPIODATA)

GPIO Port A (APB) base: 0x4000.4000  
GPIO Port A (AHB) base: 0x4005.8000  
GPIO Port B (APB) base: 0x4000.5000  
GPIO Port B (AHB) base: 0x4005.9000  
GPIO Port C (APB) base: 0x4000.6000  
GPIO Port C (AHB) base: 0x4005.A000  
GPIO Port D (APB) base: 0x4000.7000  
GPIO Port D (AHB) base: 0x4005.B000  
GPIO Port E (APB) base: 0x4002.4000  
GPIO Port E (AHB) base: 0x4005.C000  
GPIO Port F (APB) base: 0x4002.5000  
GPIO Port F (AHB) base: 0x4005.D000  
Offset 0x000  
Type RW, reset 0x0000.0000

# Pin Diagram of TM4C123

TM4C123GH6P  
Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

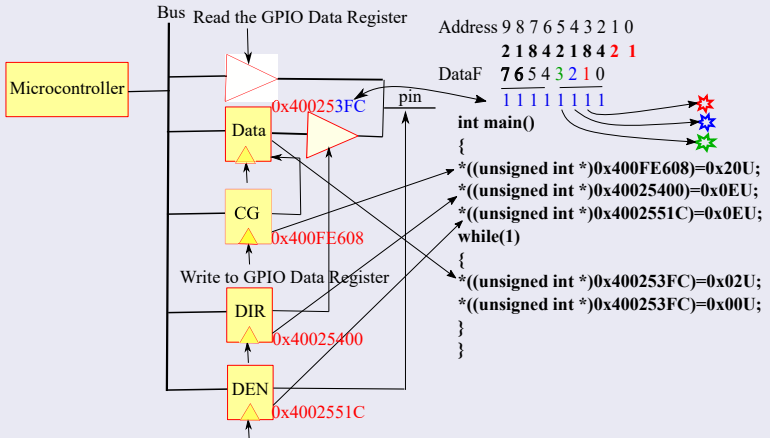


# Red LED Glow Program

TM4C123GH6P

# Micro-controllers Programming Concepts

## Simple Program





# PULLUP and PULL DOWN Resistance

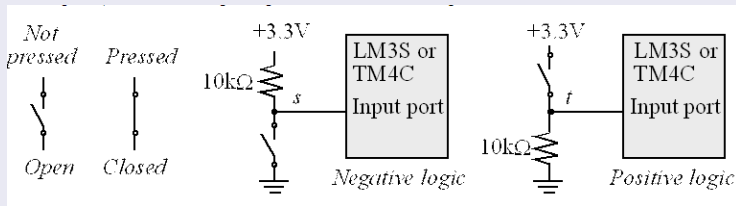
TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## Switch Interfacing

- Pull-up and Pull-down resistors are used to correctly bias the inputs of digital gates to stop them from floating about randomly when there is no input condition



# PULLUP Register

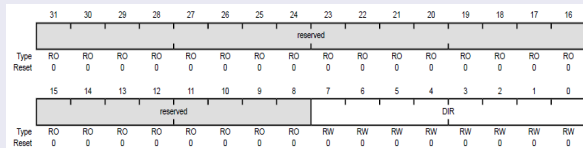
TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## Register Information

- Pull up register is write protected
- To remove the write protection, we need GPIOCR to set before PULLUP Register



0:7 Bits

PortF PULL UP Register Set  
 $0x40025514 = 0x10$

### GPIO Pull-Down Select (GPIOPDR)

GPIO Port A (APB) base: 0x4000.4000  
GPIO Port A (AHB) base: 0x4005.8000  
GPIO Port B (APB) base: 0x4000.5000  
GPIO Port B (AHB) base: 0x4005.9000  
GPIO Port C (APB) base: 0x4000.6000  
GPIO Port C (AHB) base: 0x4005.A000  
GPIO Port D (APB) base: 0x4000.7000  
GPIO Port D (AHB) base: 0x4005.B000  
GPIO Port E (APB) base: 0x4002.4000  
GPIO Port E (AHB) base: 0x4005.C000  
GPIO Port F (APB) base: 0x4002.5000  
GPIO Port F (AHB) base: 0x4005.D000  
Offset 0x514  
Type RW, reset 0x0000.0000

# GPIOCR Register

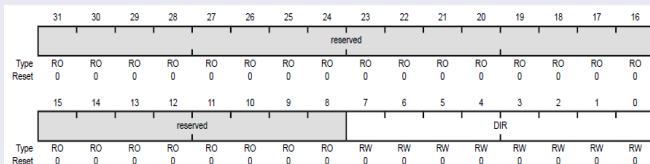
TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## Register Information

- The GPIOCR register is the commit register.



0:7 Bits

Default value is 0xFF

0x40025524=0xFF

- 0 The corresponding GPIOAFSEL, GPIOPUR, GPIOPDR, or GPIODEN bits cannot be written.
- 1 The corresponding GPIOAFSEL, GPIOPUR, GPIOPDR, or GPIODEN bits can be written.

### GPIO Commit (GPIOCR)

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

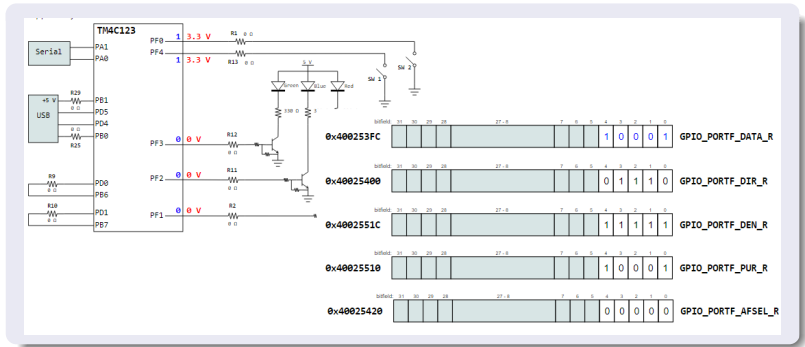
Offset 0x524

Type -, reset -

# Pin Diagram of TM4C123

TM4C123GH6P  
Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh



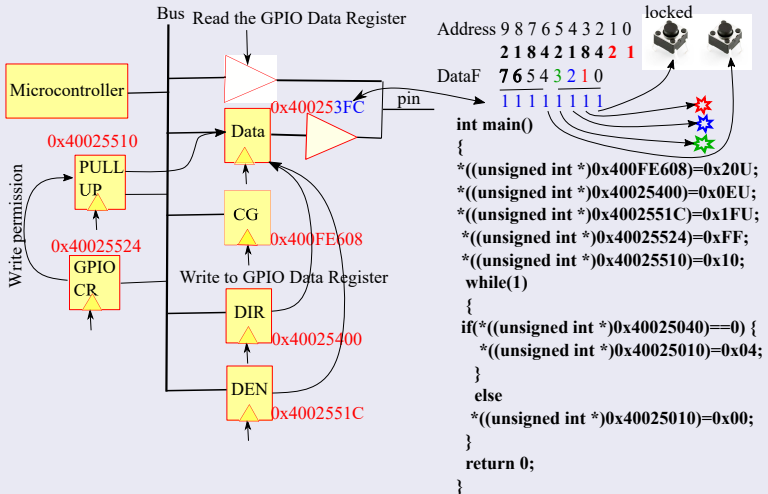
# Blue LED Control using Switch at Pin 4

TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## Simple Program



# Unlock Special Function Pin

TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## Register Setting

- To enable the use of these pins, we have to set two registers GPIOLOCK and GPIOCR

**Table 10-1. GPIO Pins With Special Considerations**

GPIO Pins	Default Reset State	GPIOAFSEL	GPIODEN	GPiopDR	GPiopUR	GPiopCTL	GPIOCR
PA[1:0]	UART0	0	0	0	0	0x1	1
PA[5:2]	SSI0	0	0	0	0	0x2	1
PB[3:2]	i <sup>2</sup> C0	0	0	0	0	0x3	1
PC[3:0]	JTAG/SWD	1	1	0	1	0x1	0
PD[7]	GPIO <sup>a</sup>	0	0	0	0	0x0	0
PF[0]	GPIO <sup>a</sup>	0	0	0	0	0x0	0

a. This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

# GPIO LOCK Register

TM4C123GH6P  
Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## GPIOLOCK

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	RW	RW	RW	RW	RW	RW	RW	RW	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

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

0:31 Bits

To Unlock the Special Port  
0x40025520=0x4C4F434B

Default value=0x4C4F434B

## GPIO Lock (GPIOLOCK)

GPIO Port A (APB) base: 0x4000.4000  
GPIO Port A (AHB) base: 0x4005.8000  
GPIO Port B (APB) base: 0x4000.5000  
GPIO Port B (AHB) base: 0x4005.9000  
GPIO Port C (APB) base: 0x4000.6000  
GPIO Port C (AHB) base: 0x4005.A000  
GPIO Port D (APB) base: 0x4000.7000  
GPIO Port D (AHB) base: 0x4005.B000  
GPIO Port E (APB) base: 0x4002.4000  
GPIO Port E (AHB) base: 0x4005.C000  
GPIO Port F (APB) base: 0x4002.5000  
GPIO Port F (AHB) base: 0x4005.D000  
Offset 0x520  
Type RW, reset 0x0000.0001

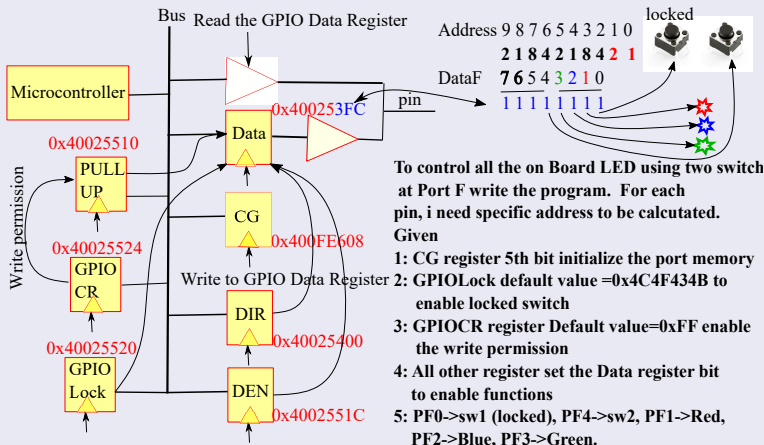
# Control RGB LED using On Board Switches

TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## Write a Program





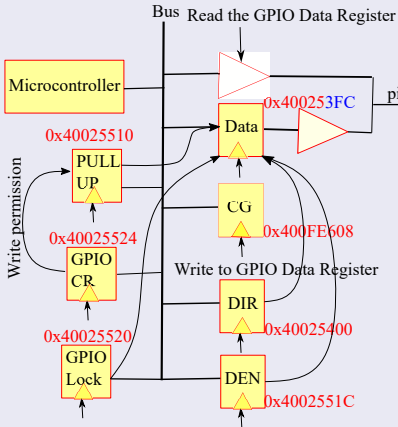
# Control all on Board LED using Switches

TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## Write a Program



```
int main()
{
    *((unsigned int *)0x400FE608)=0x20;
    *((unsigned int *)0x40025400)=0x0E;
    *((unsigned int *)0x4002551C)=0x1F;
    *((unsigned int *)0x40025520)=0x4C4F434B;
    *((unsigned int *)0x40025524)=0xFF;
    *((unsigned int *)0x40025510)=0x11;
    while(1) {
        if(*((unsigned int *)0x40025044)==0x00){
            *((unsigned int *)0x40025010)=0x04;
        }
        else if(*((unsigned int *)0x40025044)==0x10)
        {
            *((unsigned int *)0x40025008)=0x02; }
        else if(*((unsigned int *)0x40025044)==0x01)
        {
            *((unsigned int *)0x40025020)=0x08;
        }
        else{
            *((unsigned int *)0x40025010)=0x00;
            *((unsigned int *)0x40025008)=0x00;
            *((unsigned int *)0x40025020)=0x00;
        }
    }
}
```

# Simplify the Coding # directive and Volatile

TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## #define volatile Datatype

- The earlier written program uses the dereferencing of pointer address to access the port memory
- Same can be defined with a new name of user choice using Preprocessor Directive
- In the address pointer type casting, we have to include volatile keyword
  - **volatile** keyword use with register where R/W permission of bit is allowed
  - It is used to tell the compiler that register object changes frequently
  - volatile is a qualifier
- **int** and **long** data type in Arm Instruction classes is of 32 Bit.

# Use of # directive and Volatile

TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## #define volatile Datatype

```
#define CG *((volatile unsigned int *)0x400FE608) // clock gating
#define DIR *((volatile unsigned int *)0x40025400) // direction register (b01110)->portF
#define DEN *((volatile unsigned int *)0x4002551C) //digital enable (b11111)->portF
#define LOCK *((volatile unsigned int *)0x40025520) //Unlock the pinF[0]
#define CR *((volatile unsigned int *)0x40025524) //Commit register allow write permission
#define PULLUP *((volatile unsigned int *)0x40025510) //pull up register(b10001);
#define DATAF *((unsigned int *)0x40025044) //Data register of port F
#define RED_LED *((unsigned int *)0x40025008) // Red LED Address Pin
#define BLUE_LED *((unsigned int *)0x40025010) //Blue LED Address Pin
#define GREEN_LED *((unsigned int *)0x40025020) //Green LED Address Pin
int main()
{
    CG=0x20; DIR=0x0E; DEN=0x1F; LOCK=0x4C4F434B; CR=0xFF; PULLUP=0x11;
    while(1) {
        if(DATAF==0x00) {
            BLUE_LED=0x04;
        }
        else if*((unsigned int *)0x40025044)==0x10) {
            RED_LED=0x02; }
        else if*((unsigned int *)0x40025044)==0x01) {
            GREEN_LED=0x08; }
        else{
            RED_LED=0x00;
            BLUE_LED=0x00;
            GREEN_LED=0x00; }
    }
    return 0;
}
```

# Use of Bitwise Operator of C

TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## Bitwise Operators

- `c=a|b;` OR
- `c=a&b;` AND
- `c=a^ b;` Exclusive OR
- `c=b>>1;` right shift
- `c=b<<1;` left shift
- `c= ~b;` NOT
- Lower bit hex calculation is easier, but high order bit calculation of hex is time taking
- Let say i want to set bit 1 of GIPOF Data ports For RED, BLUE, GREEN LED.
  - `#define RED (1U<<2)`
  - `#define BLUE (1U<<3)`
  - `#define GREEN (1U<<4)`

# Use of Bitwise Operators

TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## Bitwise

```
#define CG *((volatile unsigned int *)0x400FE608) // clock gating
#define DIR *((volatile unsigned int *)0x40025400) // direction register (b01110)->portF
#define DEN *((volatile unsigned int *)0x4002551C) //digital enable (b11111)->portF
#define DATAF *((volatile unsigned int *)0x400253FC) //Data register of port F
```

```
#define RED (1U<<1)
#define BLUE (1U<<2)
#define GREEN (1U<<3)
int main()
```

### OR to Set Bit

XXXXXXXXTX

00000010

XXXXXXXX1X

reg

mask

reg set

### AND to Clear

XXXXXXXXTX

11111101

XXXXXXXX0X

reg

mask

reg clear

```
{
    CG=0x20;
    DIR=0x0E; // DIR|=(RED|BLUE|GREEN) = DIR|=((1U<<1)|(1U<<2)|(1U<<3))
    DEN=0x0E; // DEN|=(RED|BLUE|GREEN) = DEN|=((1U<<1)|(1U<<2)|(1U<<3))
    while(1)
    {
        DATAF|=RED; // DATAF|=(1U<<2)
        DATAF&=~RED; //DATAF&=~(1U<<2)
    }
    return 0;
}
```

# Read/Write/Modify Issue During the Interrupt

TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## Bitwise Operaotrs

- Read/write/modify is fast enough in most cases
- But during the interrupt the read/modify/write operation stuck the current execution and program counter jump to the new instruction address.
- During the interrupt service routine GPIODATA register bits may be modified.
- After the completion of ISR, the processor resume its current execution.
- Resume instruction further changes the GPIODATA bits, which lost the ISR bit change
- Through the pointer arithmetic this problem can be minimized to a certain extend

# Array and Pointer

TM4C123GH6P

Micro-  
controllers  
Programming  
Concepts

Dr. Munesh  
Singh

## Bitwise Operaotrs

- Arm instruction read/write/modify instruction is replace with an atomic write operation
- Each bit of GPIODATA register is addressable ( 8 bit GPIODATA register is addressed by 256 register)
- If we write the instruction to write the bit at the particular address, then we can access that address through pointer arithmetic or array base address

```
#define CG *((unsigned int *)0x400FE608U) //clock gating
#define DIR *((unsigned int *)0x40025400U) // Direction register
#define DEN *((unsigned int *)0x4002551CU) //Digital enable function
#define GPIODATAF *((unsigned int *)0x400253FCU) // GPIOF Register
#define GPIODATABITF ((unsigned int *)0x40025000U) //Base address of GPIOF Register
#define RED_LED (1U<<1)
#define BLUE_LED (1U<<2)
#define GREEN_LED (1U<<3)
```

```
int main()
{
    CG=0x20U;
    DIR=0x0F11U;
```

0x52	0x480d	LDR W	R0, [R0]
0x54	0x6801	LDR	R1, [R0]
0x56	0xf051 0x0102	ORRS.W	R1, R1, #2
0x5a	0x6001	STR	R1, [R0]

# Array and Pointer Use

## Bitwise Operations

```
#define CG *((unsigned int *)0x400FE608U) //clock gating
#define DIR *((unsigned int *)0x40025400U) // Direction register
#define DEN *((unsigned int *)0x4002551CU) //Digital enable function
#define GPIODATAF *((unsigned int *)0x400253FCU) // GPIOF Register
#define GPIODATABITF ((unsigned int *)0x40025000U) //Base address of GPIOF Register
#define RED_LED (1U<<1)
#define BLUE_LED (1U<<2)
#define GREEN_LED (1U<<3)
```

```
int main()
{
    CG=0x20U;
    DIR=0x0EU;
    DEN=0x0EU;
    while(1){
        int count=0;
        //GPIODATAF=RED_LED;
        *(GPIODATABITF+RED_LED)=RED_LED;
        // GPIODATABITF[RED_LED]=RED_LED;

        while(count<1000000){
            ++count;
        }
        count=0;
        GPIODATAF&=~RED_LED;
        while(count<1000000){
            ++count;
        }
    }
    return 0;
```

0x52: 0x480d	LDR.W	R0, [PC, #0x34]
0x54: 0x6801	LDR	R1, [R0]
0x56: 0xf051 0x0102	ORRS.W	R1, R1, #2
0x5a: 0x6001	STR	R1, [R0]

*(GPIODATABITF+RED_LED)=RED_LED; // GPIODATABITF[RED_L...		
0x54 0x2102	MWS	R1, #2
0x56: 0x4a0b	LDR.W	R2, [PC, #0x2c] ...
0x58: 0x6011	STR	R1, [R2]