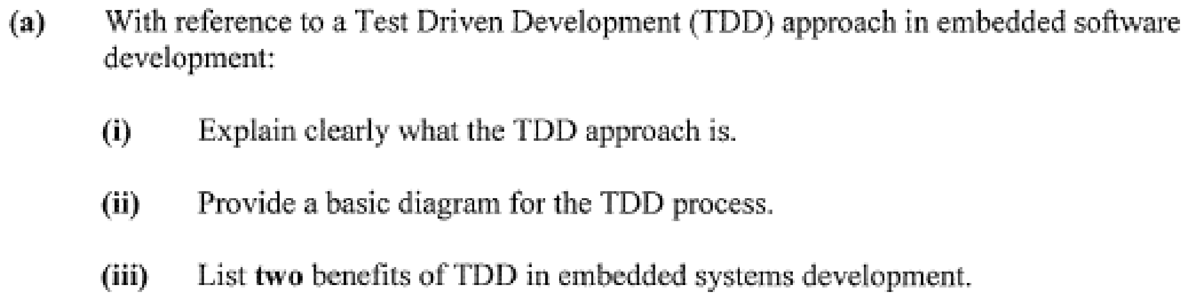
Example Questions:

Question 1

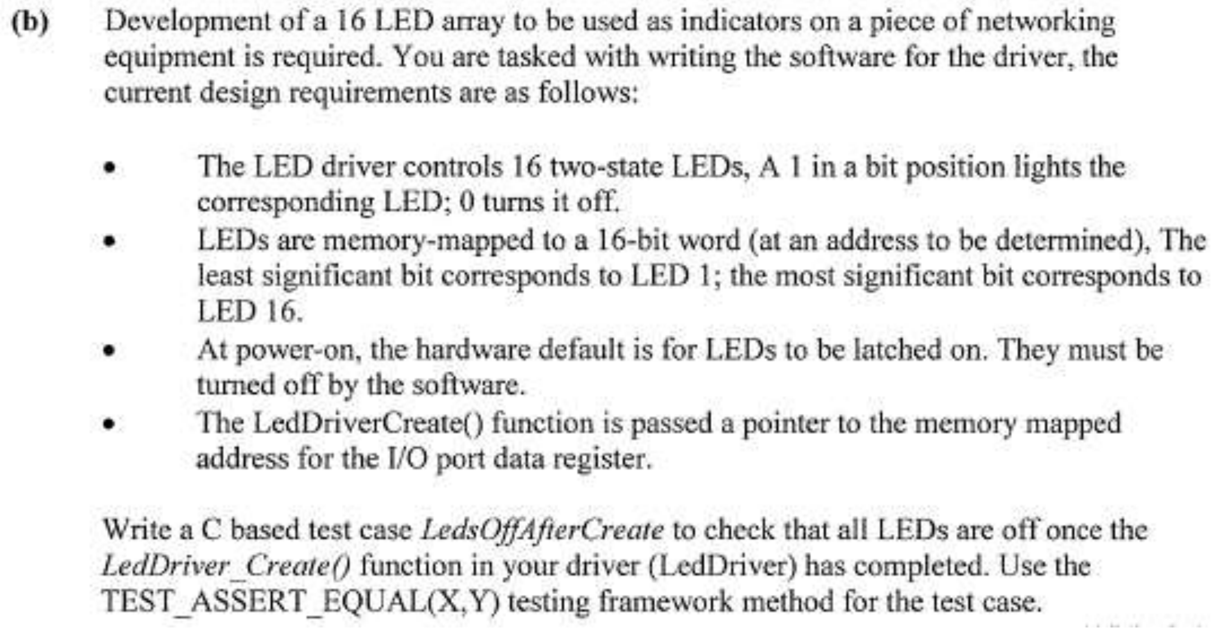


(a) With reference to a Test Driven Development (TDD) approach in embedded software development:

(i) Explain clearly what the TDD approach is.

(ii) Provide a basic diagram for the TDD process.

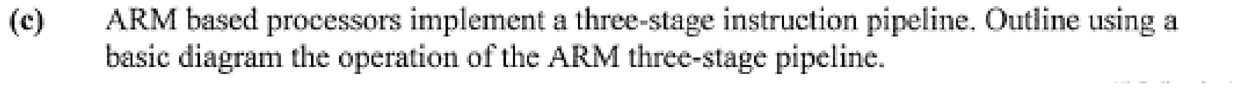
(iii) List **two** benefits of TDD in embedded systems development.



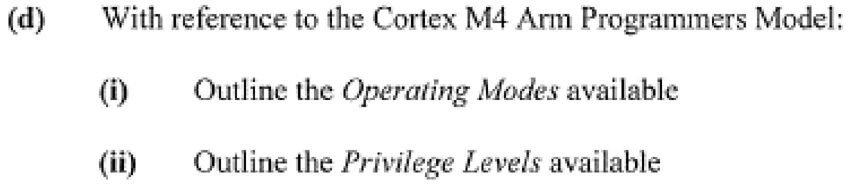
(b) Development of a 16 LED array to be used as indicators on a piece of networking equipment is required. You are tasked with writing the software for the driver, the current design requirements are as follows:

- The LED driver controls 16 two-state LEDs, A 1 in a bit position lights the corresponding LED; 0 turns it off.
- LEDs are memory-mapped to a 16-bit word (at an address to be determined), The least significant bit corresponds to LED 1; the most significant bit corresponds to LED 16.
- At power-on, the hardware default is for LEDs to be latched on. They must be turned off by the software.
- The LedDriverCreate() function is passed a pointer to the memory mapped address for the I/O port data register.

Write a C based test case *LedsOffAfterCreate* to check that all LEDs are off once the *LedDriver_Create()* function in your driver (LedDriver) has completed. Use the TEST_ASSERT_EQUAL(X,Y) testing framework method for the test case.



(c) ARM based processors implement a three-stage instruction pipeline. Outline using a basic diagram the operation of the ARM three-stage pipeline.



(d) With reference to the Cortex M4 Arm Programmers Model:

(i) Outline the *Operating Modes* available

(ii) Outline the *Privilege Levels* available

Question 2

(a)

Development of a 16 LED array to be used as indicators on a piece of networking equipment is required. You are tasked with writing the software for the driver, the current design requirements are as follows:

- The LED driver controls 16 two-state LEDs, A 1 in a bit position lights the corresponding LED; 0 turns it off.
- LEDs are memory-mapped to a 16-bit word (at an address to be determined), The least significant bit corresponds to LED 1; the most significant bit corresponds to LED 16.
- Multiple LEDs can be turned on seperatly.
- The LedDriverCreate() function is passed a pointer to the memory mapped address for the I/O port data register.

Write a C based test case *TurnOnMultipleLEDs* to check multiple LEDs have been turned on after using multiple calls to *LedDriver_TurnOn(unint16_t LedNumber)* function in your driver (*LedDriver*) have completed. Use the *TEST_ASSERT_EQUAL(X,Y)* testing framework method for the test case.

(b)

What is the ARM AMBA, outline its components, their features and where they are used.

(c)

With reference to the Cortex M4 Instruction Set Architecture (ISA):

(i) Explain what an ISA is, including the name of the ISA implemented on the Cortex M4.

(ii) The ARM Cortex M4 ISA supports some SIMD instructions, explain what the purpose of these instructions are.

(iii) Outline the purpose of the ARM CMSIS framework.

Question 3

(a) **(i)** How does a Memory Mapped I/O approach allow us to manipulate the GPIO in an Embedded System?

(ii) When declaring memory mapped register addresses in C a certain keyword qualifier should be used, what is it and why is it needed?

(b) With reference to memory mapped addressing methods and register manipulation:

(i) What are the disadvantages of the Read Modify Write method of register manipulation.

(ii) List the advantages provided by using Bit Specific and Bit Banded addressing methods.

(c) Considering the following diagram (Figure 1) of a switch and an LED connected to a TM4C123GH6PM microcontroller. The purpose of the software to be developed is to light LED L1 when switch S1 is pushed.

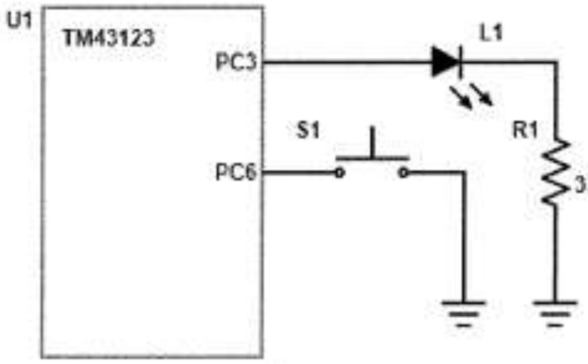


Figure 1

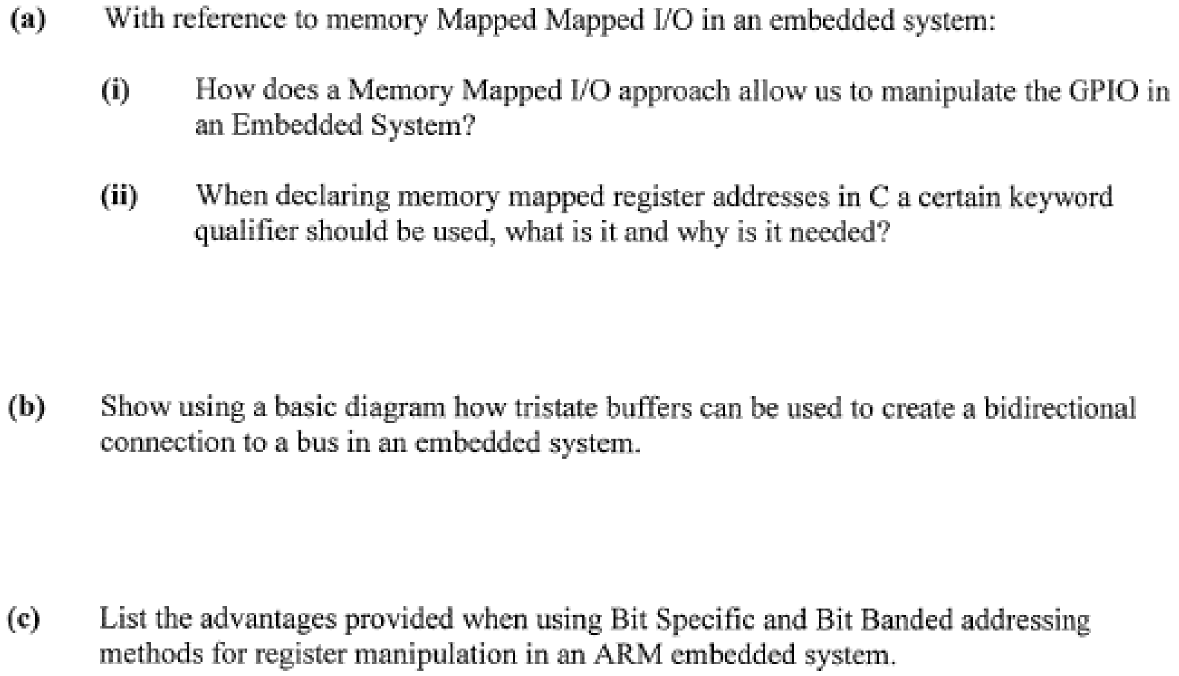
(i) Provide the pre-processor directives needed to access the registers using a memory mapped addressing approach. Your answer should adhere to the following requirements:

- Use Bit Specific addressing for the data register on pins PC3 and PC6.
- Use Bit Banded addressing to access the pull up resistor register.
- Show how Bit Specific and Bit Banded addresses were calculated.
- Use standard memory mapped addressing for all other registers.

(ii) Provide a *initPortC()* function to setup the relevant registers for the circuit to operate as required. Use comments to show what the initialisation steps are.

(iii) Develop a *main()* function that polls the input every 500ms using a *delay_ms(500)* function and sets the output accordingly.

Question 4



(a) With reference to memory Mapped Mapped I/O in an embedded system:

(i) How does a Memory Mapped I/O approach allow us to manipulate the GPIO in an Embedded System?

(ii) When declaring memory mapped register addresses in C a certain keyword qualifier should be used, what is it and why is it needed?

(b) Show using a basic diagram how tristate buffers can be used to create a bidirectional connection to a bus in an embedded system.

(c) List the advantages provided when using Bit Specific and Bit Banded addressing methods for register manipulation in an ARM embedded system.

(d) Considering the following diagram (Figure 1) of a switch and an LED connected to a TM4C123GH6PM microcontroller. The purpose of the software to be developed is to light LED L1 when switch S1 is pushed.

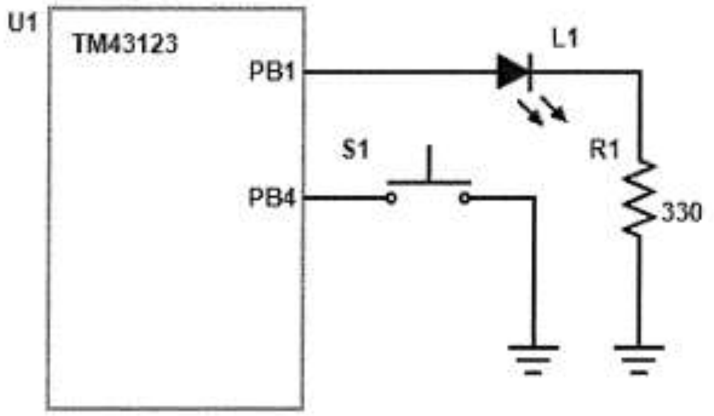


Figure 1

(i) Provide the pre-processor directives needed to access the registers using a memory mapped addressing approach. Your code should adhere to the following requirements:

- Use bit specific addressing for the data register on pins PB1 and PB4.
- Use bit banded addressing to access any required pull up/down resistor register entries.
- Show how Bit Specific and Bit Banded addresses were calculated.
- Use standard memory mapped addressing for all other registers.

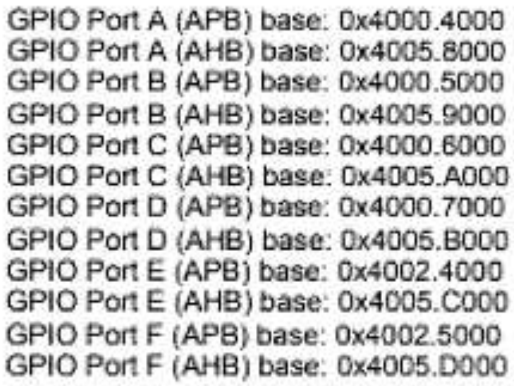
(ii) Provide a *initPortB()* function to setup the relevant registers for the circuit to operate as required.

(iii) Develop a *main()* function that polls the input every 200ms using a *delay_ms(200)* function and sets the output accordingly.

Appendix

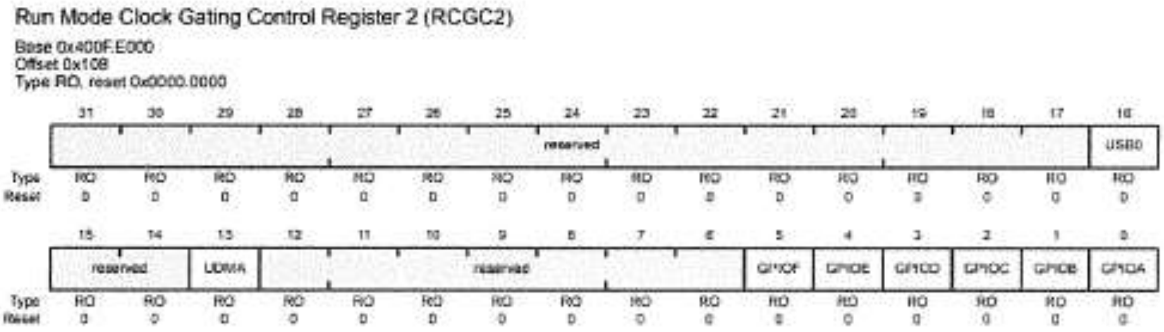
Question 3, 4

Base Addresses



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

Register 136: Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108



Run Mode Clock Gating Control Register 2 (RCGC2)
Base 0x400F.E000
Offset 0x108
Type RO, reset 0x0000.0000

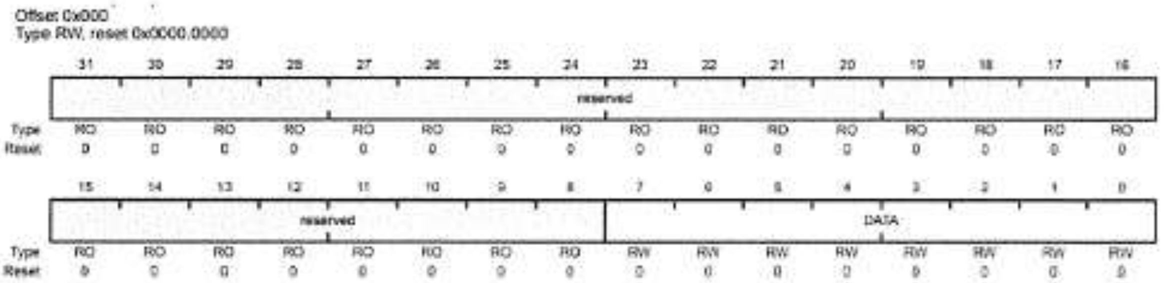
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															USB0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		UDMA	reserved							GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
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

0 = Disconnect clock, 1 = Connect clock for relevant port GPIO functions

---

GPIO Data (GPIODATA), offset 0x000



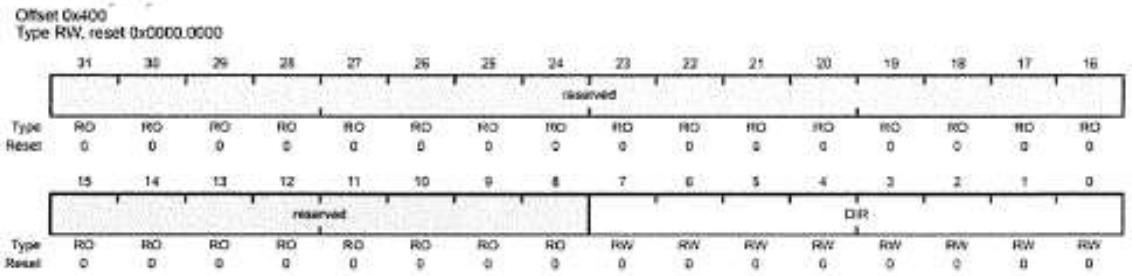
Offset 0x000
Type RW, reset 0x0000.0000

	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								DATA							
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

DATA 0-7 Corresponds to Pins 0-7, 0 = Digital low, 1 = Digital High

GPIO Direction (GPIODIR), offset 0x400

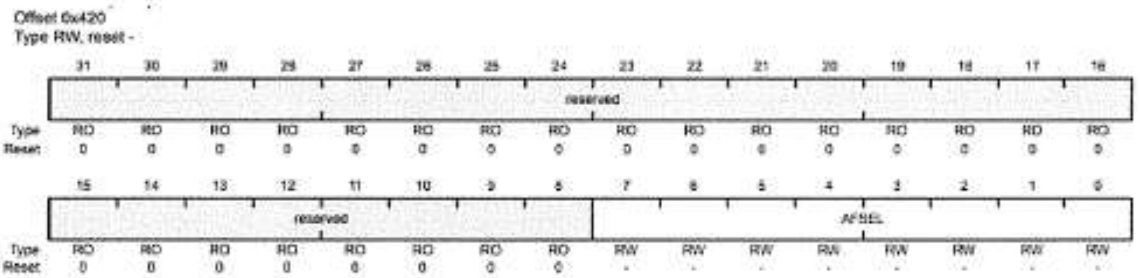


Offset 0x400
Type RW, reset 0x0000.0000

	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

DIR 0-7 Corresponds to Pins 0-7, 0 = Input, 1 = Output

GPIO Alternate Function Select (GPIOAFSEL), offset 0x420



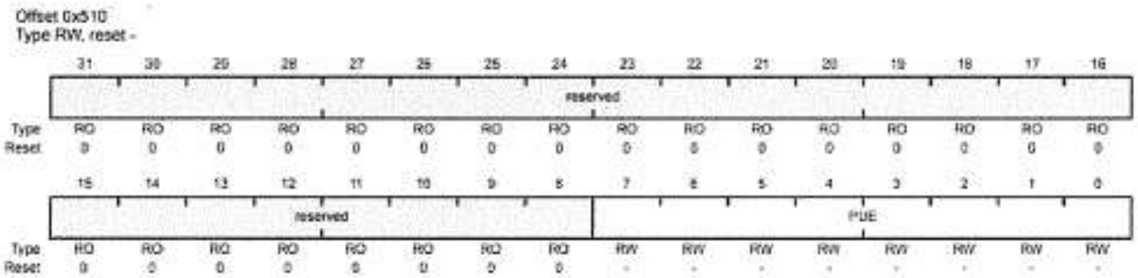
Offset 0x420
Type RW, reset -

	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								AFSEL							
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	-	-	-	-	-	-	-	-

AFSEL 0-7 corresponds to alternate function select for Pins 0-7, 0 = Disable, 1 = Enable

---

GPIO Pull-Up Select (GPIOPUR), offset 0x510



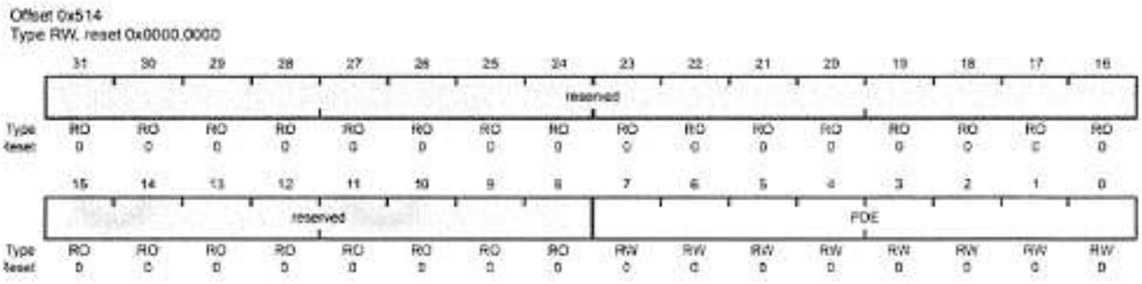
Offset 0x510
Type RW, reset -

	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								PUE							
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	-	-	-	-	-	-	-	-

PUE 0-7 Corresponds to Pull Up Enable for Pins 0-7, 0 = Disabled, 1 = Enabled

---

GPIO Pull-Down Select (GPIOPDR), offset 0x514

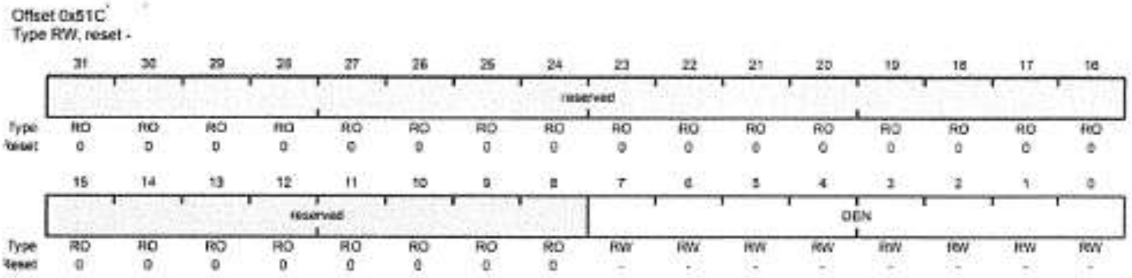


Offset 0x514
Type RW, reset 0x0000.0000

	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								PDE							
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

PDE 0-7 Corresponds to Pull Down Enable for Pins 0-7, 0 = Disabled, 1 = Enabled

GPIO Digital Enable (GPIODEN), offset 0x51C



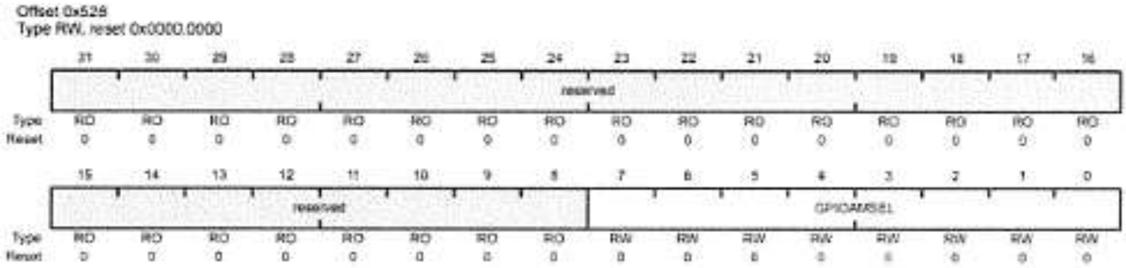
Offset 0x51C
Type RW, reset -

	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								DEN							
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	-	-	-	-	-	-	-	-

DEN 0-7 Corresponds to Pins 0-7, 0 = Digital function disabled, 1 = Digital function enabled

GPIO Analog Mode Select (GPIOAMSEL), offset 0x528



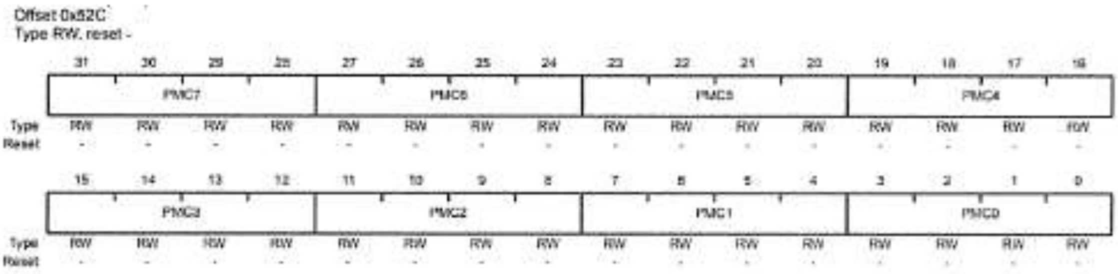
Offset 0x528
Type RW, reset 0x0000.0000

	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								GPIOAMSEL							
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

GPIOAMSEL 0-7 corresponds to the Analog Mode Select for Pins 0-7, 0= Disable, 1= Enable

GPIO Port Control (GPIOPCTL), offset 0x52C



Offset 0x52C
Type RW, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PMC7				PMC6				PMC5				PMC4			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PMC3				PMC2				PMC1				PMC0			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

PMC 0-7 corresponds to Port Mux Control for Pins 0-7, set the 4 bits to the alternate function.

Bit Specific Addressing:

If we wish to access bit	*Constant*
7	0x0200
6	0x0100
5	0x0080
4	0x0040
3	0x0020
2	0x0010
1	0x0008
0	0x0004

Port	Base address
PortA	0x40004000
PortB	0x40005000
PortC	0x40006000
PortD	0x40007000
PortE	0x40024000
PortF	0x40025000

Bit Banded Addressing:

Address Range		Memory Region	Instruction and Data Accesses
Start	End		
0x4000.0000	0x400F.FFFF	Peripheral bit-band region	Direct accesses to this memory range behave as peripheral memory accesses, but this region is also bit addressable through bit-band alias.
0x4200.0000	0x43FF.FFFF	Peripheral bit-band alias	Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not permitted.

```
bit_word_offset = (byte_offset x 32) + (bit_number x 4)

bit_word_addr = bit_band_base + bit_word_offset
```
