Karan Vora (kv2154)
Real-Time Embedded Systems Assignment 3 (EL648HW3)

Problem 1):

Solution A):

0x40000000 to 0x60000000

Solution B):

0x4002 0C00 to 0x4002 0FFF

Solution C):

MODER
1) Written by the software to configuration I/O direction mode
2) 32 bits
3) Address offset: 0x00

OTYPER
1) Set by software to configure the output type of the I/O port
2) 32 bits
3) Address offset: 0x04

OSPEEDR
1) Are written by software to configure the I/O output speed
2) 32 bits
3) Address offset: 0x08

PUPDR
1) Are written by software to configure the I/O pull-up or pull-down
2) 32 bits
3) Address offset: 0x0C

IDR
1) These bits are read-only and can be accessed in word mode only. They contain the input value of the corresponding I/O port.
2) 32 bits
3) Address offset: 0x10

ODR
1) These bits can be read and written by software
2) 32 bits
3) Address offset: 0x14

BSRRL
1) These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits return the value 0x0000.

2) 16 bits for 1 individual pin
3) Address offset: 0x18

BSRRH
1) These bits are read or write only
2) 16 bits for 1 individual pin
3) Address offset: 0x18

==========

Problem 2):

Solution B):

```
typedef struct
{
        volatile uint32_t MODER
        volatile uint32_t OTYPER
        volatile uint32_t OSPEEDER
        volatile uint32_t PUPDR
        volatile uint32_t IDR
        volatile uint32_t ODR
        volatile uint16_t BSRRL
        volatile uint16_t BSRRH
} GPIO_TypeDef;

#define GPIOD_BASE (0x40020C00)
#define GPIOD ((GPIO_TypeDef *) GPIOD_BASE)
```

Solution C):

IDR might be declared as volatile const instead of just a volatile because these bits are read-only and can be accessed in word mode only. Rest all registers are either read-write or only write.

Solution D):

No we do not need to use malloc() in C to allocate memory as already GPIO has been allocated memory address and dynamically addressing would result in erratic behavior of the controller.

Solution E):

To set ‘01’ in the 2 bit register.

MODER has 2 bits per pin:

```
GPIOD->MODER &= ~(3 << 2*3);
GPIOD->MODER |= 1 << 2*3;
```

OTYPER has 1 bit per pin:

```
GPIOD->OTYPER |= 1 << 3;
```

OSPEEDR has 2 bits per pin:
GPIOD->OSPEEDR &= ~(3 << 2*3);
GPIOD->OSPEEDR |= 1 << 2*3;

PUPDR has 2 bits per pin:
GPIOD->PUPDR &= ~(3 << 2*3);
GPIOD->PUPDR |= 1 << 2*3;

ODR has 1 bit per pin:
GPIOD->ODR |= 1 << 3;

BSRRL has 1 bit per pin:
GPIOD->BSRRL |= 1 << 3;

BSRRH has one bit per pin:
GPIOD->BSRRH |= 1 << 3;

To set the value 1 to all pins in the GPIO
MODER has two bits per pin:
GPIOD->MODER |= 0x55555555;

OTYPER has one bit per pin and the upper 16 bits are reserved. So just set the lower 16 bits:
GPIOD->OTYPER |= 0xffff;

OSPEEDR has two bits per pin:
GPIOD->OSPEEDR |= 0x55555555;

PUPDR has two bits per pin:
GPIOD->PUPDR |= 0x55555555;

ODR has one bit per pin and the upper 16 bits are reserved.

set the lower 16 bits: GPIOD->ODR |= 0xffff;

BSRRL has one bit per pin: GPIOD->BSRRL |= 0xffff;

BSRRH has one bit per pin: GPIOD->BSRRH |= 0xffff;

Setting a value of 1 which result
MODER: Sets pin(s) to general purpose output mode.
OTYPER: Sets output pin(s) to open drain type.
OSPEEDR: Sets speed to medium.
PUPDR: Sets a pull-up resistor on pins.
ODR: Writes a high value of 1 on the output pins.
BSRRL: Sets a value of 1 on the output pins.
BSRRH: Resets the value of the output pins .Sets it to 0.