|  | **Final Lab Assessment** **CSE360: Computer Interfacing   Department of Computer Science and Engineering**  **Total Marks: 10** |
| --- | --- |

**[Complete all the blanks and commands with necessary information in the given C type file.**

**Also answer ALL 4 short questions.]**

**Name: Section:**

**ID:**

### i) .C file **0.5 x 12 = 6**

| #include "i2c.h"  void I2C\_GPIO\_Init() {  *// Enable clock from 2 different Ports to configure I2C3*   RCC->AHB1ENR |= **(a) 1<<0**;  RCC-> **(b)AHB1ENR** |=**(b) 1<<2**;  *// Configure pins for I2C3 SCL and SDA*  GPIO**(c)C**->**(c)MODER** &= **(c) ~((3 << (9 \* 2))**; *// Clear MODER bit for the 1st port (SDA)*  **(d)**GPIOA->**MODER** &= **~((3 << (8 \* 2))**; *// Clear MODER bit for the 2nd port (SCL)*  GPIO**(e)C**-> MODER |= **(e)(2 << (9 \* 2))**; *// Set alternate function mode*  GPIO**(f)A**-> MODER |= **(f)(2 << (8 \* 2))**; *// Set alternate function mode*   GPIO->OTYPER |= ;  GPIO->PUPDR |= ;   GPIO->OSPEEDR |= ;   *// Set alternate function to I2C3*  GPIO**(g)C**->**(g)AFR[1]** |= (**(g) 4 << ((9 - 8) \* 4)**);  GPIO**(h)A**->**(h)AFR[1]** |= (**(h) 4 << ((8 - 8) \* 4)**); }  void **I2C\_Init**() {  *// Enabling Clock for I2C1*  RCC->APB1ENR |= *// RCC\_APB1ENR\_I2C1EN*   *// Reset I2C3*  I2C3->CR1 |= (1 << 15);  I2C3->CR1 &= **(i)~(1 << 15)**;   I2C3->CR2 = 16; // Set APB1 clock frequency in 16 MHz (i)  I2C3->CCR = 80; //Set the clock control register ccr = fclk / (2 \* i2c\_freq) i2c\_freq is 100khz here and fclk is 16mhz(ii)  I2C3->TRISE = 17; //max rise time (iii)   *// Enable I2C3*  I2C3->CR1 |= (1 << 0); *// I2C\_CR1\_PE* }  void I2C\_Start() {  I2C3->CR1 |= (1 << 8); *// I2C\_CR1\_START*  while (!(I2C3->SR1 & (1 << 0)));  }  void I2C\_Stop() {  I2C3->CR1 |= (1 << 9); }  void I2C\_ACK\_Enable() {  I2C3->CR1 |= (1 << 10); }  void I2C\_ACK\_Disable() {  I2C3->CR1 &= ~(1 << 10);  }  void **I2C\_SendAddress**(uint8\_t address, uint8\_t rw) {  **(j) I2C3->DR = (address << 1) | rw**;  **(k)\_\_\_\_\_\_I2C3->CR1 |= (1 << 10);\_\_\_\_\_\_\_\_**;  while (!(I2C3->SR1 & (1 << 1)));  (void)I2C3->SR1;*// Clear ADDR flag*  (void)I2C3->SR2; *// Clear ADDR flag* }  void **I2C\_WriteByte**(uint8\_t data) {  while (!(I2C3->SR1 & (1 << 7))); *// I2C\_SR1\_TXE*   I2C3->DR = data;  while (!(I2C3->SR1 & (1 << 2))); *// I2C\_SR1\_BTF* }   uint8\_t I2C\_ReadByte() {  while (!(I2C3->SR1 & (1 << 6))); *// I2C\_SR1\_RXNE*  return I2C3->DR; }  void **I2C\_Write\_Buffer**(uint8\_t address, uint8\_t buffer[], uint8\_t len) {  I2C\_Start();  I2C\_SendAddress(address, 0);  for (int i = 0; i < len; i++)  {  I2C\_WriteByte(buffer[i]);  }  I2C\_Stop(); }  void **I2C\_Read\_Buffer**(uint8\_t address, uint8\_t buffer[], uint8\_t len) {  I2C\_Start();  **(l)**I2C\_SendAddress(address, 1);  for (int i = 0; i < len; i++)  {  I2C\_ReadByte(buffer[i]);  }  I2C\_Stop(); } |
| --- |

### 

### ii) Short Questions **1 x 4 = 4**

| 1. **Write the alternative port(s) and pin(s) to configuration I2C3.**   **Ans.** Port H: pin 7 (PH7) for SCL and pin 8 for SDL (PH8)   1. **Write down the mode of the PUPDR register.**   **Ans.**  00: No pull-up, pull-down  01: Pull-up  10: Pull-down  11: Reserved  However, the mode of our PUPDR register in the given configuration is not set. If no config was set from earlier, it will be 00 or No pull-up, pull down. However, for our configuration of I2C3, since I2C3 uses an open drain mechanism we need to set the PUPDR register at mode 01 or pull up.   1. **State the correlation of Command (i), (ii), (iii) from the .C file**   **Ans.**  (i) I2C3->CR2 = 16; sets the APB1 bus clock frequency for the STM32 to drive the i2c peripherals at 16 MHZ. This determines how fast can I2c3 bus can operate.  After that, the second line (ii)I2C3->CCR = 80; uses the frequency with a formula CCR= fclk/2\*freq\_i2c= 16Mhz/(2\*100khz)=80. Here the CCR is the clock frequency for the I2c3 protocol for data transfer and communication through configured pins.  Lastly, the rise time is set at 17 ns using I2C3->TRISE = 17; which ensures that the maax time for a signal to go from low to high. It is important to keep sufficient time based on this TRISE time as the microprocessor allows the SCL line to go from low to high allowing devices to properly interpret the signals.    Hence, these commands ensure that the stm32 operates at 16Mhz with an i2c3 data transfer rate set at 100khz frequency. Lastly, rise time ensures that all devices can understand the low-high signals which is too needed for communication.   1. while (!(I2C1->SR1 & (1 << 0))) - **Explain the use of this command with all the necessary information.**   **Ans.**  SR1 register holds many flags that indicate the status of I2c devices. The 0th bit of the SR1 register of I2C is of SB (start bit). The following command is used to wait until the start bit has been set on the I2C1 bus. The while loop checks if the 0th bit of the I2C1 SR1 registers is set to 1 by (1<<0), which sets the start condition, and when it is set the while loop breaks unless it waits. It is needed to set the start bit in order to start data communication between master and slave. |
| --- |

### 