**1.Program to toggle the state of an LED connected to a GPIO pin**

**PROGRAM:**

**led\_blink**

**#include "stm32f4xx.h" //// void delayMs(int n); ///// void led\_blink() //// {**

**/////// RCC->AHB1ENR |= 1; /\* enable GPIOA clock \*/ /////// GPIOA->MODER |= (0<<11);**

**GPIOA->MODER |= (1<<10); /\* set pin to output mode \*/ ////// while(1) //////// {**

**///////// GPIOA->ODR |= 0x00000020; /\* turn on LED \*/ //////// delayMs(500);**

**/////// GPIOA->ODR &= ~0x00000020; /\* turn off LED \*/ ///////////// delayMs(500);**

**///// } //////////// } /////////// void delayMs(int n) //////// { ///////// int i;**

**///// for (; n > 0; n--) /////// for (i = 0; i < 3195; i++) ; ////// }**

**led\_blink.h**

**void led\_blink();**

**main.c**

**#include "led\_blink.h" ////// void main() ////// { ////// led\_blink(); /// }**

**2.PROGRAM TO READING A SWITCH AND DISPLAYING IT ON AN LED**

**PROGRAM:**

**gpio.h ///////////// #ifndef GPIO\_H ////////////// #define GPIO\_H**

**#include "stm32f4xx.h" ///////////// void GPIO\_Init(void);**

**#endif /\* GPIO\_H \*/**

**gpio.c**

**#include "gpio.h" //////////void GPIO\_Init(void) {**

**RCC->AHB1ENR |= RCC\_AHB1ENR\_GPIOCEN; /\* enable GPIOC clock \*/**

**RCC->AHB1ENR |= RCC\_AHB1ENR\_GPIOAEN; /\* enable GPIOA clock \*/**

**GPIOA->MODER &= ~GPIO\_MODER\_MODER5\_Msk; /\* clear pin mode \*/**

**GPIOA->MODER |= GPIO\_MODER\_MODER5\_0; /\* set pin to output mode \*/**

**GPIOC->MODER &= ~GPIO\_MODER\_MODER13\_Msk; /\* clear pin mode to input mode \*/ ////////// }**

**main.c**

**#include "stm32f4xx.h" ///////////// #include "gpio.h"**

**int main(void) { /////////GPIO\_Init(); /////////// while(1) {**

**if (GPIOC->IDR & GPIO\_IDR\_IDR\_13) /\* if PC13 is high \*/**

**GPIOA->BSRR = GPIO\_BSRR\_BR\_5; /\* turn off green LED \*/**

**else**

**GPIOA->BSRR = GPIO\_BSRR\_BS\_5; /\* turn on green LED \*/**

**3. Write a Simple Program to transmit the data between Microcontroler using UART**

**PROGRAM:**

**uart.c**

**#include "uart.h"**

**////// #define GPIOAEN (1U<<0) /////////// #define PA2MOD (2U<<4)**

**#define PA2AF (7U<<8) /////////// #define APB1EN (1U<<17)**

**#define USART2\_TX (1U<<3) //////////#define USART2EN (1U<<13)**

**#define SR\_TXE (1U<<7) ///////// #define SYS\_FREQ 16000000**

**#define APB1\_CLK SYS\_FREQ //////// #define UART\_BAUDRATE 115200**

**static void uart\_set\_baudrate(USART\_TypeDef \*USARTx, uint32\_t PeriphClk, uint32\_t BaudRate);**

**static uint16\_t compute\_uart\_bd(uint32\_t PeriphClk, uint32\_t BaudRate);**

**void delayMs(int n)**

**{**

**for (; n > 0; n--)**

**for (int i = 0; i < 5000; i++);**

**}**

**void uart2\_init(void) /////////{ /////// RCC->AHB1ENR |= GPIOAEN;**

**///////// GPIOA->MODER |= PA2MOD; /////// GPIOA->AFR[0] |= PA2AF;**

**////// RCC->APB1ENR |= APB1EN;**

**uart\_set\_baudrate(USART2, APB1\_CLK, UART\_BAUDRATE);**

**USART2->CR1 = USART2\_TX; /////// USART2->CR1 |= USART2EN; ////////// }**

**static void uart\_set\_baudrate(USART\_TypeDef \*USARTx, uint32\_t PeriphClk, uint32\_t BaudRate)**

**{**

**USARTx->BRR = compute\_uart\_bd(PeriphClk, BaudRate);**

**}**

**static uint16\_t compute\_uart\_bd(uint32\_t PeriphClk, uint32\_t BaudRate)**

**{**

**return ((PeriphClk + (BaudRate / 2U)) / BaudRate);**

**}**

**void uart2\_write(int ch)**

**{**

**// Make sure the transmit data register is empty**

**while(!(USART2->SR & SR\_TXE));**

**// Write to transmit data register**

**USART2->DR = (ch & 0xFF);**

**}**

**uart.h**

**#ifndef INC\_UART\_H\_ /////////// #define INC\_UART\_H\_**

**#include <stdint.h> ////////#include "stm32f4xx.h"**

**void uart2\_init(void); //////// void uart2\_write(int ch);**

**void delayMs(int n); //////// #endif /\* INC\_UART\_H\_ \*/**

**main.c**

**#include "stm32f4xx.h" /////////// #include <stdint.h>**

**#include "uart.h" //////////////int main()**

**{ ///////// uart2\_init(); //////// while(1) /////// {**

**/// char name[] = "Transmited\n\r"; ///// for(int i = 0; name[i] != '\0'; i++)**

**//////uart2\_write(name[i]); //////////delayMs(50);**

**} ///////// return 0; //////// }**

**5.IMPLEMENT A TIMER – BASED DELAY FUNCTION**

**PROGRAM:**

**Main.c**

**#include "stm32f4xx.h" /////////// #include "systick.h" //// int main(void) {**

**systick(); //// }**

**Systick.c**

**#include"stm32f4xx.h" /////////// #define SYSTICK\_LOAD\_VAL 16000**

**///////// #define CTRL\_ENABLE (1U<<0)**

**#define CTRL\_CLKSRC (1U<<2)**

**#define CTRL\_COUNTFLAG (1U<<16)**

**void systickDelayMs(int delay); //////// void systick(void){**

**RCC->AHB1ENR |= 1; /\* enable GPIOA clock \*/**

**GPIOA->MODER &= ~0x00000C00; /\* clear pin mode \*/**

**GPIOA->MODER |= 0x00000400; /\* set pin to output mode \*/**

**while(1) //// { //////// GPIOA->ODR |= 0x00000020; /\* turn on LED \*/**

**systickDelayMs(500); ////// GPIOA->ODR &= ~0x00000020; /\* turn off LED \*/**

**systickDelayMs(500); ////// } ////// }**

**/\* 16 MHz SYSCLK \*/ ////// void systickDelayMs(int delay) ////{**

**SysTick->LOAD = SYSTICK\_LOAD\_VAL;**

**SysTick->VAL = 0;**

**SysTick->CTRL = CTRL\_ENABLE | CTRL\_CLKSRC;**

**for(int i=0; i<delay ; i++){**

**while((SysTick->CTRL & CTRL\_COUNTFLAG) == 0){}**

**} //// SysTick->CTRL = 0; /////}**

**Systick.h**

**void systick(void);**

**6.CONFIGURE A TIMER TO GENERATE PERIODIC INTERRUPTS**

**PROGRAM:**

**Main.c**

**#include "stm32f4xx.h"**

**#define PERIOD 1000**

**#include"periodictimer.h"**

**void delayMs(int n);**

**int main(void)**

**{**

**periodictimer(); ///////////// }**

**Periodictimer.c**

**#include "stm32f4xx.h"**

**#define PERIOD 1000**

**void TIM2\_IRQHandler(void);**

**void periodictimer(void)**

**{ ///////////// \_\_disable\_irq();**

**RCC->AHB1ENR |= 1;**

**GPIOA->MODER &= ~0x00000C00;**

**GPIOA->MODER |= 0x00000800;**

**GPIOA->AFR[0] = 0x00100000;**

**RCC->APB1ENR |= 1; /\* enable TIM2 clock \*/**

**TIM2->PSC = 16000 - 1; /\* divided by 16000 \*/**

**TIM2->ARR = 0xFFFF; /\* max count \*/**

**TIM2->CCR1 = PERIOD;**

**TIM2->CNT = 0;**

**TIM2->CCMR1 = 0x30; /\* Output compare toggle \*/**

**TIM2->CCER = 1; /\* CC1 enable \*/**

**TIM2->CR1 = 1; /\* enable counter \*/**

**TIM2->DIER |= 2; /\* enable CC1IE \*/**

**NVIC\_EnableIRQ(TIM2\_IRQn); /\* enable interrupt in NVIC \*/**

**\_\_enable\_irq(); /\* global enable IRQs \*/**

**while(1) { }**

**} /////////// void TIM2\_IRQHandler(void)**

**{**

**TIM2->SR = 0; /\* clear UIF \*/**

**TIM2->CCR1 = (TIM2->CCR1 + PERIOD) & 0xFFFF; /\* update CCR1 \*/**

**}**

**Periodictimer.h**

**void periodictimer(void);**

**7.Write C program to configure a timer for an input capture using STM32F446RE**

**PROGRAM:**

**main.c**

**#include <stdio.h> /////// #include <stdint.h>**

**#include "stm32f4xx.h" /////////#include "tim.h" /////////////// int timestamp = 0 ;**

**/Set up : Connect a jumper wire from PA5 to PA6/ ///////////////// int main(void)**

**{ ///////// tim2\_pa5\_output\_compare(); /////////tim3\_pa6\_input\_capture();**

**while(1) //////// { /////////// /Wait until edge is captured/**

**while(!(TIM3->SR & SR\_CC1IF)){} ////////// /Read captured value/**

**timestamp = TIM3->CCR1; /////// } ////////}**

**tim.c**

**#include <stdio.h> //////// #include <stdint.h>**

**#include "stm32f4xx.h" //////////#include "tim.h"**

**int timestamp = 0 ;**

**/Set up : Connect a jumper wire from PA5 to PA6/ ////////// int main(void)**

**{ //////// tim2\_pa5\_output\_compare(); ////tim3\_pa6\_input\_capture();**

**while(1) /////////// {**

**/Wait until edge is captured/**

**while(!(TIM3->SR & SR\_CC1IF)){} ////////// /Read captured value/**

**timestamp = TIM3->CCR1; /////////// } /////////// }**

**tim.h**

**#ifndef TIM\_H\_ ////////#define TIM\_H\_ //////// void tim2\_1hz\_init(void);**

**void tim2\_pa5\_output\_compare(void); ////////// #define SR\_UIF (1U<<0)**

**#endif /\* TIM\_H\_ \*/**

**8.Write C program to configure a timer for an output capture using STM32F446RE**

**PROGRAM:**

**main.c /////// #include <stdio.h> ////////////////#include <stdint.h>**

**#include "stm32f4xx.h" ///////////#include "tim.h" ///// int main(void)**

**{ ///// tim2\_pa5\_output\_compare(); ////////// while(1)**

**{ //// } //// } //// tim.c ////// #include "stm32f4xx.h"**

**#define TIM2EN (1U<<0) /////#define CR1\_CEN (1U<<0)**

**#define OC\_TOGGLE ((1U<<4) | (1U<<5))**

**#define CCER\_CC1E (1U<<0)**

**#define GPIOAEN (1U<<0) ///#define AFR5\_TIM (1U<<20)**

**void tim2\_1hz\_init(void) ///{ ////// RCC->APB1ENR |=TIM2EN;**

**////// TIM2->PSC = 1600 - 1 ; // 16 000 000 / 1 600 = 10 000**

**TIM2->ARR = 10000 - 1; // 10 000 / 10 000 = 1 ///TIM2->CNT = 0;**

**TIM2->CR1 = CR1\_CEN; //// } /////void tim2\_pa5\_output\_compare(void)**

**{ //// RCC->AHB1ENR |=GPIOAEN;**

**GPIOA->MODER &=~(1U<<10);**

**GPIOA->MODER |=(1U<<11);**

**GPIOA->AFR[0] |=AFR5\_TIM;**

**RCC->APB1ENR |=TIM2EN;**

**TIM2->PSC = 1600 - 1 ; // 16 000 000 / 1 600 = 10 000**

**/Set auto-reload value/**

**TIM2->ARR = 10000 - 1; // 10 000 / 10 000 = 1**

**/Set output compare toggle mode/**

**TIM2->CCMR1 = OC\_TOGGLE;**

**/Enable tim2 ch1 in compare mode/**

**TIM2->CCER |=CCER\_CC1E;**

**/Clear counter/**

**TIM2->CNT = 0;**

**/Enable timer/**

**TIM2->CR1 = CR1\_CEN;**

**}**

**tim.h**

**/\* //////////// \* tim.h**

**#ifndef TIM\_H\_**

**#define TIM\_H\_**

**void tim2\_1hz\_init(void);**

**void tim2\_pa5\_output\_compare(void);**

**#define SR\_UIF (1U<<0)**

**#endif /\* TIM\_H\_ \***