1) LED Blinking using software delay.

```
#include <lpc214X.h>
void delay(unsigned int);

int main(){
PINSEL2=0X00000000;
IODIR1=0xFFFFFFFF;
while(1)
{
IOSET1=0xFFFFFFFF;
delay(500);
IOCLR1=0xFFFFFFFF;
delay(500);
}
}

void delay(unsigned int i)
{
int j,k;
for(j=0;j<i;j++)
{
for(k=0;k<1275;k++);
}
}
```

2)LED Blinking using TIMER With Match Register

```
#include<lpc214x.h>
void delay(unsigned int msec);
void PLL_Init(void);
int main()
{
PINSEL2=0x00000000;
IODIR1=0xffffffff;
PLL_Init();
while(1)
{
IOSET1=0xffffffff;
delay(1000);
IOCLR1=0xffffffff;
delay(1000);
}
}
void delay(unsigned int msec)
{
T0CTCR=0x0;
T0TCR=0x00;
T0PR=59999;
```

```
T0MR0=1000;
T0MCR=0x0002;
T0TCR=0x02;
T0TCR=0x01;
while(T0TC!=T0MR0);
T0TCR=0x00;
}
void PLL_Init(void)
{
PLL0CON=0x01;
PLL0CFG=0x24;
PLL0FEED=0xaa;
PLL0FEED=0x55;
while(!(PLL0STAT&0x0400));
PLL0CON=0x03;
PLL0FEED=0xaa;
PLL0FEED=0x55;
VPBDIV=0x01;
}
```

2)LED Blinking using TIMER Without Match Register

```
#include<lpc214x.h>
void delay(unsigned int msec);
void PLL_Init(void);
int main()
{
PINSEL2=0x00000000;
IODIR1=0xffffffff;
PLL_Init();
while(1)
{
IOSET1=0xffffffff;
delay(1000);
IOCLR1=0xffffffff;
delay(1000);
}
}
void delay(unsigned int msec)
{
T0CTCR=0x0;
T0TCR=0x00;
T0PR=59999;
T0TCR=0x02;
T0TCR=0x01;
while(T0TC<msec);
T0TCR=0x00;
T0TC=0;
}
```

```
void PLL_Init(void)
{
PLL0CON=0x01;
PLL0CFG=0x24;
PLL0FEED=0xaa;
PLL0FEED=0x55;
while(!(PLL0STAT&0x0400));
PLL0CON=0x03;
PLL0FEED=0xaa;
PLL0FEED=0x55;
VPBDIV=0x01;
}
```

3) Interfacing with 16x2 LCD

```
#include <lpc214x.h>

#define LCD_PORT 0x00FF0000
#define EN 1<<10
#define RS 1<<11
#define RW 1<<20
#define LCD_SHIFT 16

void LCD_init(void);
void LCD_data(unsigned char);
void LCD_cmd(unsigned char);

void LCD_delay(unsigned int time){
int i,j;
for(i=0;i<time;i++)
  for(j=0;j<200;j++);
}

void LCD_data(unsigned char ch)
{
   IOCLR1 = LCD_PORT;
   IOSET1 = ch << LCD_SHIFT;
   IOSET0 = RS;
   IOCLR0 = RW;
   IOSET0 = EN;
   LCD_delay(100);
   IOCLR0 = EN;
}

void LCD_cmd(unsigned char ch)
{
   IOCLR1 = LCD_PORT;
   IOSET1 = ch << LCD_SHIFT;
   IOCLR0 = RS;
```

```
   IOCLR0 = RW;
   IOSET0 = EN;
   LCD_delay(100);
   IOCLR0 = EN;
}

void LCD_init(void){
PINSEL0 &= 0xFF0FFFFF;
PINSEL1 &= 0xFFFFFCFF;
PINSEL2 &= 0xFFFFFFF3;
IODIR0=RS|EN|RW;
IODIR1= LCD_PORT;
LCD_cmd(0x38);
LCD_cmd(0x06);
LCD_cmd(0x0C);
LCD_cmd(0x01);
LCD_cmd(0x80);
}


void LCD_display(int row , int position , char *ch){
unsigned char temp;
if(row==1){
temp = 0x80 |(position-1);
}
else{
temp= 0xC0 | (position -1);
}
LCD_cmd(temp);
while(*ch){
LCD_data(*ch++);
}
}

int main(){
unsigned int temp;
LCD_init();
while(1){
LCD_display(1,4,"PCCOE");
LCD_display(2,4,"AMCL");
LCD_delay(20);
}
}
```

4) ADC in HEX code:

```
#include <lpc214x.h>

#define LCD_PORT 0x00FF0000
```

```
#define EN 1<<10
#define RS 1<<11
#define RW 1<<20
#define LCD_SHIFT 16
void LCD_init(void);
void LCD_data(unsigned char);
void LCD_cmd(unsigned char);
void LCD_delay(unsigned int time){
int i,j;
for(i=0;i<time;i++)
  for(j=0;j<200;j++);
}
void LCD_data(unsigned char ch)
{
   IOCLR1 = LCD_PORT;
   IOSET1 = ch << LCD_SHIFT;
   IOSET0 = RS;
   IOCLR0 = RW;
   IOSET0 = EN;
   LCD_delay(100);
   IOCLR0 = EN;
}
void LCD_cmd(unsigned char ch)
{
   IOCLR1 = LCD_PORT;
   IOSET1 = ch << LCD_SHIFT;
   IOCLR0 = RS;
   IOCLR0 = RW;
   IOSET0 = EN;
   LCD_delay(100);
   IOCLR0 = EN;
```

```
}
void LCD_init(void)
{
PINSEL0 &= 0xFF0FFFFF;
PINSEL1 &= 0xFFFFFCFF;
PINSEL2 &= 0xFFFFFFF3;
IODIR0=RS|EN|RW;
IODIR1= LCD_PORT;
LCD_cmd(0x38);
LCD_cmd(0x06);
LCD_cmd(0x0C);
LCD_cmd(0x01);
LCD_cmd(0x80);
}
void LCD_display(int row , int position , char *ch)
{
unsigned char temp;
if(row==1){
temp = 0x80 |(position-1);
}
else{
temp= 0xC0 | (position -1);
}
LCD_cmd(temp);
while(*ch){
LCD_data(*ch++);
}
}

void ADCInit(void)
{
```

```
PINSEL1 |= 0x05000000;
}

unsigned int ADC_Read(unsigned char channel)
{
 static unsigned int ad1_data;
 AD0CR=0x00200300 |(1<<channel);
 AD0CR |= 1<<24;
 while(!(AD0GDR & 0x80000000));
 ad1_data=(AD0GDR & 0x0000FFC0)>>6;
 return ad1_data;
}
int main()
{
 unsigned int temp;
 char buf[16];
 LCD_init();
 ADCInit();
 while(1)
 {
  temp=ADC_Read(1);
  sprintf(buf,"ADC result: 0x%03X", temp);
  LCD_display(1,1,buf);
  LCD_delay(20);
 }
}
```

4B) ADC voltage Display code:

```
#include <lpc214x.h>
#include <stdint.h>
#include <stdio.h>
#include <string.h>
#define LCD_PORT 0x00FF0000
#define EN 1<<10
```

```
#define RS 1<<11
#define RW 1<<20
#define LCD_SHIFT 16
void LCD_init(void);
void LCD_data(char);
void LCD_cmd(char);
void LCD_delay(unsigned int time){
int i,j;
for(i=0;i<time;i++)
  for(j=0;j<200;j++);
}
void LCD_data(char ch)
{
   IOCLR1 = LCD_PORT;
   IOSET1 = ch << LCD_SHIFT;
   IOSET0 = RS;
   IOCLR0 = RW;
   IOSET0 = EN;
   LCD_delay(100);
   IOCLR0 = EN;
}
void LCD_cmd(char ch)
{
   IOCLR1 = LCD_PORT;
   IOSET1 = ch << LCD_SHIFT;
   IOCLR0 = RS;
   IOCLR0 = RW;
   IOSET0 = EN;
   LCD_delay(100);
   IOCLR0 = EN;
}

void LCD_init(void)
{
PINSEL0 &= 0xFF0FFFFF;
PINSEL1 &= 0xFFFFFCFF;
PINSEL2 &= 0xFFFFFFF3;
IODIR0=RS|EN|RW;
IODIR1= LCD_PORT;
LCD_cmd(0x38);
LCD_cmd(0x06);
LCD_cmd(0x0C);
LCD_cmd(0x01);
LCD_cmd(0x80);
}

void LCD_display(int row , int position , char *ch)
{
```

```
unsigned char temp;
if(row==1){
temp = 0x80 |(position-1);
}
else{
temp= 0xC0 | (position -1);
}
LCD_cmd(temp);
while(*ch){
LCD_data(*ch++);
}
}
void ADCInit(void)
{
PINSEL1 |= 0x05000000;
}

unsigned int ADC_Read(unsigned char channel)
{
 static unsigned int ad1_data;
 AD0CR=0x00200300 |(1<<channel);
 AD0CR |= 1<<24;
 while ( !(AD0DR1 & 0x80000000) ); /* Wait till DONE */
 ad1_data=(AD0GDR & 0x0000FFC0)>>6;
 return ad1_data;
}
int main(void)
{
static unsigned int temp;
float voltage;
char volt[18];
LCD_init();
ADCInit();
//PINSEL1 = 0xFFFFFCFF; /* P0.28 as AD0.1 */
//AD0CR=0x00200300; /* ADC operational, 10-bits, 11 clocks for conversion */
while(1)
{
temp=ADC_Read(1);
voltage = ( (temp/1023.0) * 3.3 ); /* Convert ADC value to equivalent voltage */
LCD_cmd(0xC0);
sprintf(volt, "Voltage=%.3fV", voltage);
LCD_display(1,2,volt);
LCD_delay(20);
}
}
```

5) EEPROM.C

```
#include "I2C.h"
#define EEPROM_Addr 0xA0      // Device address
void EEPROM_write(unsigned int add, unsigned char *data, unsigned char len)
{
   unsigned char i;
   I2CStart();                  // Assert START
   I2Csend(EEPROM_Addr | I2Cwrite); // Device address with LSB bit 0
   I2Csend(add >> 8);            // Address higher byte
   I2Csend(add & 0xFF);          // Address lower byte

   for(i = 0; i < len; i++)
      I2Csend(*data++);        // Write the array to EEPROM

   I2CStop();
}

void EEPROM_read(unsigned int add, unsigned char *data, unsigned char len)
{
   unsigned char i;
   I2CStart();                   // Assert START
   I2Csend(EEPROM_Addr | I2Cwrite); // Device address with LSB bit 0 (Dummy Write)
   I2Csend(add >> 8);             // Address higher byte
   I2Csend(add & 0xFF);           // Address lower byte
   I2CStart();                   // Assert Restart
   I2Csend(EEPROM_Addr | I2Cread);   // Device address with LSB bit 1

   for(i = 0; i < len; i++)
      *data++ = I2Cget();         // Read EEPROM

   I2CStop();
}
```

STM32 LED BLINKING

```
/* USER CODE BEGIN WHILE */
while (1)
 {
  /* USER CODE END WHILE */
HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET);
HAL_Delay(5000);
HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
 HAL_Delay(1000);
 /* USER CODE BE
}
```

8A) SERIAL COMMUNICATION

ASYNCHRONUS-USART2-TXPA2-RXPA3

```
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
```

```
/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */
uint8_t tx1[] = "welcome\r\n";
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);

/* USER CODE BEGIN PFP */
/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/
```

```
  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */
  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();

  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    HAL_UART_Transmit(&huart2, tx1, sizeof(tx1), 1000);
    HAL_Delay(1000);
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

8B) SERIAL COMMUNICATION TRANSMISSION AND RECEPTION

```
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

uint8_t tx_data[] = "Hello from USART2\r\n";  // MANUAL ADDITION
uint8_t rx_data[20];                          // MANUAL ADDITION

/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
CRC_HandleTypeDef hcrc;
UART_HandleTypeDef huart2;
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_CRC_Init(void);
/* USER CODE BEGIN PFP */
```

```
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */

/**
 * @brief  The application entry point.
 * @retval int
 */
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  MX_CRC_Init();

  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    // MANUAL ADDITIONS START
    HAL_UART_Transmit(&huart2, tx_data, sizeof(tx_data), 1000);  // send data
    HAL_UART_Receive(&huart2, rx_data, sizeof(rx_data), 1000);   // receive data
```

```
    HAL_UART_Transmit(&huart2, rx_data, sizeof(rx_data), 1000);  // echo back received data
    // MANUAL ADDITIONS END
  }
  /* USER CODE END WHILE */
}
```

9)INTERRUPT DRIVEN

CONFIG USART2 ASYNC, CONFIG TIM6, ENABLE TIM6 UPDATE INTERRUPT

```
#include "main.h"
#include <string.h>   // MANUAL ADDITION

/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef htim6;
UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_TIM6_Init(void);
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim); // MANUAL ADDITION

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  MX_TIM6_Init();
```

```
  HAL_TIM_Base_Start_IT(&htim6); // MANUAL ADDITION: Start timer interrupt

  while (1)
  {
    // Infinite loop left empty intentionally, main logic handled by timer interrupt
  }
}

// MANUAL ADDITION: Timer interrupt callback
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  if (htim->Instance == TIM6)
  {
    char msg[] = "Hello from Timer Interrupt\r\n";
    HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
  }
}
```