**1.Serial transmission**

#include <lpc214x.h>

void UART0\_Init(void);

void UART0\_SendChar(char ch);

char UART0\_ReceiveChar(void);

void UART0\_SendString(char \*str);

int main(void) {

char receivedChar;

UART0\_Init(); // Initialize UART0

UART0\_SendString("UART Initialized\r\n");

while (1) {

receivedChar = UART0\_ReceiveChar(); // Wait for a character

UART0\_SendChar(receivedChar); // Echo back the received character

}

}

void UART0\_Init(void) {

PINSEL0 |= 0x00000005; // Enable UART0 Rx and Tx (P0.0, P0.1)

U0LCR = 0x83; // Enable DLAB and set 8-bit data, 1 stop bit, no parity

U0DLL = 97; // Set baud rate to 9600 (assuming 15MHz PCLK)

U0DLM = 0; // Set baud rate

U0LCR &= ~0x80; // Disable DLAB

U0FCR = 0x07; // Enable and clear FIFO

}

void UART0\_SendChar(char ch) {

while (!(U0LSR & 0x20)); // Wait until THR is empty

U0THR = ch;

}

char UART0\_ReceiveChar(void) {

while (!(U0LSR & 0x01)); // Wait until data is available

return U0RBR;

}

void UART0\_SendString(char \*str) {

while (\*str) {

UART0\_SendChar(\*str++);

}

}

**2.arithmtic**

#include <lpc214x.h>

// Function to introduce delay

void delay\_ms(unsigned int ms) {

unsigned int i, j;

for (i = 0; i < ms; i++)

for (j = 0; j < 2000; j++);

}

int main() {

unsigned int num1 = 5, num2 = 3, result;

// Configure GPIO pins (P0.0 to P0.7 as output for result)

IO0DIR = 0xFF; // Set P0.0 to P0.7 as output

while (1) {

// Perform addition

result = num1 + num2;

IO0PIN = result; // Output result to P0.0 to P0.7

delay\_ms(1000); // Delay for 1 second

// Perform subtraction

result = num1 - num2;

IO0PIN = result; // Output result to P0.0 to P0.7

delay\_ms(1000); // Delay for 1 second

}

return 0;

}

**3.triangular wave**

#include <lpc214x.h>

// Define constants for DAC

#define DACR (\*(volatile unsigned int \*)0xE006C000) // DAC register address

#define MAX\_DAC\_VALUE 1023 // 10-bit maximum value

#define STEP\_SIZE 10 // Increment/Decrement step size

#define DELAY\_US 500 // Delay in microseconds

// Function to introduce delay

void delay\_us(unsigned int us) {

unsigned int i;

for (i = 0; i < us \* 10; i++); // Approximate delay

}

int main() {

unsigned int dac\_value = 0; // DAC output value

int increment = STEP\_SIZE; // Direction of waveform

// Initialize DAC (P0.25 as DAC output)

PINSEL1 |= (1 << 19); // Select P0.25 as AOUT (DAC output)

PINSEL1 &= ~(1 << 18); // Clear the other bit for P0.25

while (1) {

// Write to DAC register

DACR = (dac\_value << 6); // Shift to align with 10-bit DAC register

// Update DAC value for triangle waveform

dac\_value += increment;

// Reverse direction at max or min value

if (dac\_value >= MAX\_DAC\_VALUE || dac\_value == 0) {

increment = -increment;

}

// Delay to control frequency

delay\_us(DELAY\_US);

}

return 0;

**}**

**4.square wave**

#include <lpc214x.h>

// Define constants for DAC

#define DACR (\*(volatile unsigned int \*)0xE006C000) // DAC register address

#define MAX\_DAC\_VALUE 1023 // Maximum DAC value for a 10-bit resolution

#define MIN\_DAC\_VALUE 0 // Minimum DAC value

#define DELAY\_MS 500 // Delay in milliseconds (controls square wave frequency)

// Function to introduce delay

void delay\_ms(unsigned int ms) {

unsigned int i, j;

for (i = 0; i < ms; i++)

for (j = 0; j < 2000; j++); // Approximate delay for LPC2148

}

int main() {

unsigned int dac\_value = MAX\_DAC\_VALUE; // Start with max value

// Configure P0.25 as DAC output

PINSEL1 |= (1 << 19); // Select P0.25 as AOUT (DAC output)

PINSEL1 &= ~(1 << 18); // Clear the other bit for P0.25

while (1) {

// Write current DAC value to the DAC register

DACR = (dac\_value << 6); // Align with 10-bit DAC register (left-shift by 6)

// Toggle between max and min DAC value

if (dac\_value == MAX\_DAC\_VALUE) {

dac\_value = MIN\_DAC\_VALUE;

} else {

dac\_value = MAX\_DAC\_VALUE;

}

// Delay to create the desired square wave frequency

delay\_ms(DELAY\_MS);

}

return 0;

}

**5.7 segment**

**Decimal**

#include <lpc214x.h>

// 7-segment display configuration for numbers (common cathode)

// Each element corresponds to segments a, b, c, d, e, f, g

// For example, 0 = 0b00111111 corresponds to segments a, b, c, d, e, f

unsigned char segment\_code[] = {

0x3F, // 0

0x06, // 1

0x5B, // 2

0x4F, // 3

0x66, // 4

0x6D, // 5

0x7D, // 6

0x07, // 7

0x7F, // 8

0x6F // 9

};

void delay\_ms(unsigned int ms) {

unsigned int i, j;

for (i = 0; i < ms; i++)

for (j = 0; j < 2000; j++); // Approximate delay

}

int main() {

unsigned int number = 5; // Decimal number to display (0-9)

// Configure P0.0 to P0.7 as output for 7-segment display

IO0DIR = 0xFF; // Set P0.0 to P0.7 as output

while (1) {

// Display the number on the 7-segment

IO0PIN = segment\_code[number]; // Write corresponding segments to P0.0-P0.7

delay\_ms(1000); // Refresh every 1 second

}

return 0;

}

**2.hexa decimal**

#include <lpc214x.h>

// 7-segment display configuration for hexadecimal numbers (common cathode)

// Each element corresponds to segments a, b, c, d, e, f, g

// For example, 0 = 0b00111111 corresponds to segments a, b, c, d, e, f

unsigned char segment\_code[] = {

0x3F, // 0

0x06, // 1

0x5B, // 2

0x4F, // 3

0x66, // 4

0x6D, // 5

0x7D, // 6

0x07, // 7

0x7F, // 8

0x6F, // 9

0x77, // A

0x7C, // B

0x39, // C

0x5E, // D

0x79, // E

0x71 // F

};

void delay\_ms(unsigned int ms) {

unsigned int i, j;

for (i = 0; i < ms; i++)

for (j = 0; j < 2000; j++); // Approximate delay

}

int main() {

unsigned int number = 0; // Hexadecimal number to display (0-F)

// Configure P0.0 to P0.7 as output for 7-segment display

IO0DIR = 0xFF; // Set P0.0 to P0.7 as output

while (1) {

// Display the number on the 7-segment

IO0PIN = segment\_code[number]; // Write corresponding segments to P0.0-P0.7

delay\_ms(1000); // Refresh every 1 second

// Increment the number and wrap around after F

number = (number + 1) % 16; // Loop through 0-F

}

return 0;

}

**3.alphabeta**

#include <lpc214x.h>

// 7-segment display configuration for alphabets (common cathode)

// Each element corresponds to segments a, b, c, d, e, f, g

unsigned char alphabet\_code[] = {

0x77, // A

0x7C, // b

0x39, // C

0x5E, // d

0x79, // E

0x71, // F

0x3D, // G

0x76, // H

0x06, // I

0x1E, // J

0x75, // K (approximated)

0x38, // L

0x37, // M (approximated)

0x54, // n

0x5C, // o

0x73, // P

0x67, // q

0x50, // r

0x6D, // S

0x78, // t

0x3E, // U

0x3E, // V (same as U)

0x2A, // W (approximated)

0x76, // X (same as H)

0x6E, // Y

0x5B // Z (approximated)

};

void delay\_ms(unsigned int ms) {

unsigned int i, j;

for (i = 0; i < ms; i++)

for (j = 0; j < 2000; j++); // Approximate delay

}

int main() {

unsigned int letter\_index = 0; // Index for alphabets (0 = A, 1 = b, ..., 25 = Z)

// Configure P0.0 to P0.7 as output for 7-segment display

IO0DIR = 0xFF; // Set P0.0 to P0.7 as output

while (1) {

// Display the current alphabet on the 7-segment

IO0PIN = alphabet\_code[letter\_index]; // Write corresponding segments to P0.0-P0.7

delay\_ms(1000); // Refresh every 1 second

// Increment the letter and wrap around after Z

letter\_index = (letter\_index + 1) % 26; // Loop through A-Z

}

return 0;

}

**6.Internal ADC**

#include <lpc214x.h>

// Define the ADC channel

#define ADC\_CHANNEL 1 // ADC Channel 1 (P0.28)

// Function to initialize the ADC

void ADC\_Init() {

PINSEL1 |= (1 << 24); // Configure P0.28 as AD0.1 (ADC input)

PINSEL1 &= ~(1 << 25); // Clear other bits for P0.28

}

// Function to read value from ADC

unsigned int ADC\_Read() {

unsigned int adc\_value;

// Select ADC channel 1 and start conversion

AD0CR = (1 << 1) | // Select AD0.1 (Channel 1)

(4 << 8) | // ADC clock = PCLK/5

(1 << 21) | // Enable ADC

(1 << 24); // Start conversion

// Wait for conversion to complete

while (!(AD0GDR & (1 << 31)));

// Read the result (bits 6-15 of AD0GDR)

adc\_value = (AD0GDR >> 6) & 0x3FF; // 10-bit result

return adc\_value;

}

// Function to initialize LEDs (connected to P0.0-P0.7)

void LED\_Init() {

IO0DIR |= 0xFF; // Set P0.0 to P0.7 as output for LEDs

}

// Function to display ADC result on LEDs

void LED\_Display(unsigned int value) {

IO0PIN = (IO0PIN & ~0xFF) | (value & 0xFF); // Display lower 8 bits of value

}

int main() {

unsigned int adc\_result;

// Initialize ADC and LEDs

ADC\_Init();

LED\_Init();

while (1) {

// Read ADC value

adc\_result = ADC\_Read();

// Display ADC result on LEDs

LED\_Display(adc\_result);

// Small delay

for (volatile int i = 0; i < 50000; i++);

}

return 0;

}

**8.LED CLASHING**

#include <lpc214x.h>

// Function to introduce delay

void delay\_ms(unsigned int ms) {

unsigned int i, j;

for (i = 0; i < ms; i++)

for (j = 0; j < 2000; j++); // Approximate delay for LPC2148

}

int main() {

// Configure P0.0 as output

IO0DIR |= (1 << 0); // Set P0.0 as output

while (1) {

// Turn ON the LED

IO0SET = (1 << 0); // Set P0.0 HIGH

delay\_ms(500); // Delay of 500ms

// Turn OFF the LED

IO0CLR = (1 << 0); // Set P0.0 LOW

delay\_ms(500); // Delay of 500ms

}

return 0;

}