LED Blinking

#include "lpc214x.h"

void delay (unsigned int k);

void main(void)

{

IODIR0 = 0xFFFFFFFF; //Configure Port0 as output Port

PINSEL0 = 0; //Configure Port0 as General Purpose IO

while(1)

{

IOSET0 = 0x0000ff00; //Set P0.15-P0.8 to '1'

delay(1000); //1 sec Delay

IOCLR0 = 0x0000ff00; //Set P0.15-P0.8 to '0'

delay(1000); //1 Sec Delay

}

}

//Delay Program

//Input - delay value in milli seconds

void delay(unsigned int k)

{

unsigned int i,j;

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

for(i = 0; i<=800; i++);

}

Display Alpha

#include <LPC214X.H>

#define DS3 1<<13 // P0.13

#define DS4 1<<12 // P0.12

#define SEG\_CODE 0xFF<<16 // Segment Data from P0.16 to P0.23

unsigned char const seg\_alphabet[] = {

0x77, // 'A'

0x7C, // 'b'

0x39, // 'C'

0x5E, // 'd'

0x79, // 'E'

0x71 // 'F'

};

void delayms(int n) {

int i, j;

for(i = 0; i < n; i++) {

for(j = 0; j < 5035; j++) {;} // Delay for 60 MHz clock

}

}

int main(void) {

int count;

PINSEL0 = 0; // Configure Port0 as General Purpose IO => P0.0 to P0.15

PINSEL1 = 0; // Configure Port0 as General Purpose IO => P0.16 to P0.31

IODIR0 = SEG\_CODE | DS3 | DS4; // Configure Segment data & Select signal as output

IOSET0 = SEG\_CODE | DS3; // Disable DS3 display

IOCLR0 = DS4; // Enable DS4 Display

while (1) {

for (count = 0; count < 6; count++) {

IOCLR0 = SEG\_CODE;

IOSET0 = seg\_alphabet[count] << 16;

delayms(1000); // 1 sec delay

}

}

}

Display Numbers

#include <LPC214X.H>

#define DS3 1<<13 // P0.13

#define DS4 1<<12 // P0.12

#define SEG\_CODE 0xFF<<16 // Segment Data from P0.16 to P0.23

unsigned char const seg\_decimal[] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F};

void delayms(int n) {

int i, j;

for(i = 0; i < n; i++) {

for(j = 0; j < 5035; j++) {;} // Delay for 60 MHz clock

}

}

int main(void) {

int count;

PINSEL0 = 0; // Configure Port0 as General Purpose IO => P0.0 to P0.15

PINSEL1 = 0; // Configure Port0 as General Purpose IO => P0.16 to P0.31

IODIR0 = SEG\_CODE | DS3 | DS4; // Configure Segment data & Select signal as output

IOSET0 = SEG\_CODE | DS3; // Disable DS3 display

IOCLR0 = DS4; // Enable DS4 Display

while (1) {

for (count = 0; count < 10; count++) {

IOCLR0 = SEG\_CODE;

IOSET0 = seg\_decimal[count] << 16;

delayms(1000); // 1 sec delay

}

}

}

Display Hexa Decimal

#include <LPC214X.H>

#define DS3 1<<13 // P0.13

#define DS4 1<<12 // P0.12

#define SEG\_CODE 0xFF<<16 // Segment Data from P0.16 to P0.23

unsigned char const seg\_hexadecimal[] = {

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 delayms(int n) {

int i, j;

for(i = 0; i < n; i++) {

for(j = 0; j < 5035; j++) {;} // Delay for 60 MHz clock

}

}

int main(void) {

int count;

PINSEL0 = 0; // Configure Port0 as General Purpose IO => P0.0 to P0.15

PINSEL1 = 0; // Configure Port0 as General Purpose IO => P0.16 to P0.31

IODIR0 = SEG\_CODE | DS3 | DS4; // Configure Segment data & Select signal as output

IOSET0 = SEG\_CODE | DS3; // Disable DS3 display

IOCLR0 = DS4; // Enable DS4 Display

while (1) {

for (count = 0; count < 16; count++) {

IOCLR0 = SEG\_CODE;

IOSET0 = seg\_hexadecimal[count] << 16;

delayms(1000); // 1 sec delay

}

}

}

Square Wave

#include <lpc214x.h>

void delay(unsigned int count); // Function for generating a delay

void generate\_square\_wave(void); // Function to generate square waveform

int main(void) {

// Initialize DAC on P0.25

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

while (1) {

generate\_square\_wave(); // Generate square wave

delay(50000); // Small delay between waveform switching

}

}

void delay(unsigned int count) {

unsigned int i, j;

for (i = 0; i < count; i++) {

for (j = 0; j < 6000; j++); // Approximate delay

}

}

// Function to generate square waveform using DAC

void generate\_square\_wave(void) {

unsigned int high = 1023 << 6; // DAC value for maximum output

unsigned int low = 0 << 6; // DAC value for minimum output

for (int i = 0; i < 100; i++) {

DACR = high; // Set DAC to maximum (High)

delay(10000); // Hold for some time to create the high part of the square wave

DACR = low; // Set DAC to minimum (Low)

delay(10000); // Hold for some time to create the low part of the square wave

}

}

Triangular

#include <lpc214x.h>

void delay(unsigned int count); // Function for generating a delay

void generate\_triangle\_wave(void); // Function to generate triangular waveform

int main(void) {

// Initialize DAC on P0.25

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

while (1) {

generate\_triangle\_wave(); // Generate triangular wave

delay(50000); // Small delay between waveform switching

}

}

void delay(unsigned int count) {

unsigned int i, j;

for (i = 0; i < count; i++) {

for (j = 0; j < 6000; j++); // Approximate delay

}

}

// Function to generate triangular waveform using DAC

void generate\_triangle\_wave(void) {

unsigned int i;

// Incrementing part of the triangle

for (i = 0; i < 1023; i++) {

DACR = (i << 6); // Write to DAC (10-bit left justified)

delay(100); // Delay to control waveform frequency

}

// Decrementing part of the triangle

for (i = 1023; i > 0; i--) {

DACR = (i << 6); // Write to DAC

delay(100); // Delay to control waveform frequency

}

}