# 

# **EMBEDDED SYSTEMS PROJECT**

## **INTERFACING TRAFFIC LIGHTS WITH LPC-2148**

SUBMITTED TO

**Department of Computer Science & Engineering**

Team Members(CSE B):

Ananya Paleti – 220905081(11)

Shubhangi Shree – 220905114(16)

Manjyot Saini – 220905135(20)

Yash Thandani – 220905203(28)

Tarushi Tiwary – 220905171(24)

## ARM7 is most successful and widely used processor family in embedded system applications. So we have decided to choose ARM7 TDMI based NXP controller LPC2148. Also, ARM7 is a balance between classic and new Cortex series. ARM7 is excellent to get start with in terms of resources available on internet and quality documentation provided by NXP. It suits perfectly for beginners to get in-depth idea about hardware and software implementation.

## LPC2148 is manufactured by NXP Semiconductor (Phillips) and it is preloaded with many in-built features and peripherals. This makes it more efficient and reliable choice for an high-end application developer.

A computer chip with many small letters

Description automatically generated with medium confidence

Hardware components :

Features of LPC2148

The main features of LPC2148 include the following:

* The LPC2148 is a 16 bit or 32 bit ARM7 family based microcontroller and available in a small LQFP64 package.
* ISP (in system programming) or IAP (in application programming) using on-chip boot loader software.
* On-chip static RAM is 8 kB-40 kB, on-chip flash memory is 32 kB-512 kB, the wide interface is 128 bit, or accelerator allows 60 MHz high-speed operation.
* It takes 400 milliseconds time for erasing the data in full chip and 1 millisecond time for 256 bytes of programming
* Embedded Trace interfaces and Embedded ICE RT offers real-time debugging with high-speed tracing of instruction execution and on-chip Real Monitor software.
* It has 2 kB of endpoint RAM and USB 2.0 full speed device controller. Furthermore, this microcontroller offers 8kB on-chip RAM nearby to USB with DMA.
* One or two 10-bit ADCs offer 6 or 14 analogs i/ps with low conversion time as 2.44 μs/ channel.
* Only 10 bit DAC offers changeable analog o/p.
* External event counter/32 bit timers-2, PWM unit, & watchdog.
* Low power RTC (real time clock) & 32 kHz clock input.
* Several serial interfaces like two 16C550 UARTs, two I2C-buses with 400 kbit/s speed.
* 5 volts tolerant quick general purpose Input/output pins in a small LQFP64 package.
* Outside interrupt pins-21.
* 60 MHz of utmost CPU CLK-clock obtainable from the programmable-on-chip phase locked loop by resolving time is 100 μs.
* The incorporated oscillator on the chip will work by an exterior crystal that ranges from 1 MHz-25 MHz
* The modes for power-conserving mainly comprise idle & power down.For extra power optimization, there are individual enable or disable of peripheral functions and peripheral CLK scaling.

Memory

The LPC2148 microcontroller has 512-kB on-chip FLASH memory as well as 32-kB on-chip SRAM. Also, this microcontroller includes inherent support up to 2kB finish point USB RAM. This memory is well matched for all the microcontroller applications.

Programming

The initial step toward lpc2148 programming is an arrangement of GPIO Pins. So here are the related concepts as well as registers. The general purpose I/O port pins in LPC2148 includes P0.0 to P0.31 and P1.16 to P1.31, and actually, these pins are available based on the alternate function utilization.

Port-0 and Port-1 are 32-bit Input/output ports, and every bit of these ports can be controlled by an individual direction. The operations of port-0 & port-1 depend upon the function of a pin that is selected using the pin connected block. In Port-0, pins like P0.24, P0.26 & P0.27 are not obtainable whereas, in Port-1, the Pins 0 to 15 are not obtainable. Here, both the pins like Port-0 & Port-1 are controlled by two groups of registers discussed below.

GPIO AND REGISTERS

General-purpose input/output (GPIO) is a pin on an IC (Integrated Circuit). It can be either input pin or output pin, whose behaviour can be controlled at the run time. A group of these pins is called a port (Example, Port 0 of LPC2148 has 32 pins).

LPC2148 has two 32-bit General Purpose I/O ports.

1. PORT0

2. PORT1

PORT0 is a 32-bit port

Out of these 32 pins, 28 pins can be configured as either general purpose input or output.

1 of these 32 pins (P0.31) can be configured as general-purpose output only.

3 of these 32 pins (P0.24, P0.26 and P0.27) are reserved. Hence, they are not available for use. Also, these pins are not mentioned in pin diagram.

PORT1 is also a 32-bit port. Only 16 of these 32 pins (P1.16 – P1.31) are available for use as general-purpose input or output.

Fast and Slow GPIO Registers:

There are 5 Fast (also called Enhanced GPIO Features Registers) GPIO Registers and 4 Slow (also called Legacy GPIO Registers) GPIO Registers available to control PORT0 and PORT1.

The Slow Registers allow backward compatibility with earlier family devices using the existing codes.

GPIO Related Registers In LPC2148

1. IOxPIN - This register can be used to read or write values directly to the pins. Regardless of the direction set for the particular pins. It gives the current state of GPIO pin when read.
2. IOXDIR - This is the GPIO direction control register. Setting a bit to 'O' in this register will configure the corresponding pin to be used as input while setting it to '1 will configure it as output.
3. IOSET- This register can be used to drive an 'output' configured pin to logic 1 i.e. HIGH. Writing zerodoes not have any effect and hence it can't be used to drive a pin to Logic 0 i.e. LOW. Fordriving pins LOW IOxCLR is used .
4. IOxCLR- This register can be used to drive an 'output' configured pin to logic 0 i.e. LOW. Writing zero does not have any effect and hence it can't be used to drive a pin to Logic 1.

CODE:

#include "lpc214x.h" // Header file for LPC2148

// Define the GPIO pins for the traffic lights

#define RED\_PIN (1 << 0) // P0.0 for Red LED

#define YELLOW\_PIN (1 << 1) // P0.1 for Yellow LED

#define GREEN\_PIN (1 << 2) // P0.2 for Green LED

// Define delay functions

void delay\_ms(unsigned int delay)

{

unsigned int i;

unsigned int j;

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

{

for (j = 0; j < 60000; j++)

{

// Do nothing, just wait

}

}

}

// Initialize GPIO pins for the traffic light

void traffic\_light\_init(void)

{

// Configure P0.0, P0.1, and P0.2 as output pins

IODIR0 |= RED\_PIN | YELLOW\_PIN | GREEN\_PIN;

// Turn off all lights initially

IOCLR0 = RED\_PIN | YELLOW\_PIN | GREEN\_PIN;

}// Control traffic lights according to the standard sequence

void traffic\_light\_sequence(void)

{

while (1)

{

// Turn on Green light for 5 seconds

IOCLR0 = RED\_PIN | YELLOW\_PIN; // Turn off Red and Yellow

IOSET0 = GREEN\_PIN; // Turn on Green

delay\_ms(5000); // Wait for 5 seconds

// Turn on Yellow light for 2 seconds

IOCLR0 = GREEN\_PIN; // Turn off Green

IOSET0 = YELLOW\_PIN; // Turn on Yellow

delay\_ms(2000); // Wait for 2 seconds

// Turn on Red light for 5 seconds

IOCLR0 = YELLOW\_PIN; // Turn off Yellow

IOSET0 = RED\_PIN; // Turn on Red

delay\_ms(5000); // Wait for 5 seconds

}

}

int main(void)

{

// Initialize the traffic light system

traffic\_light\_init();

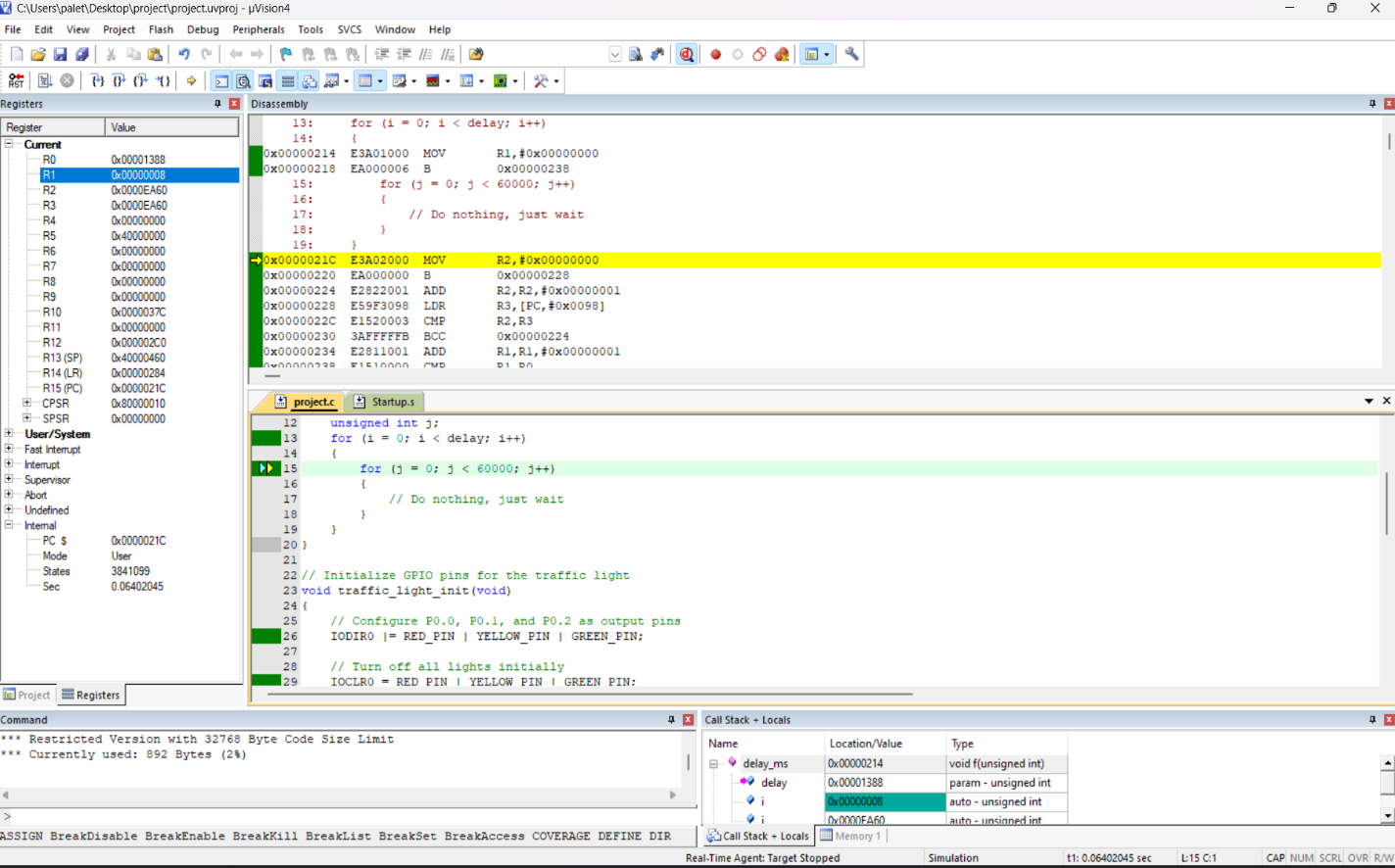
// Run the traffic light sequence in a loop

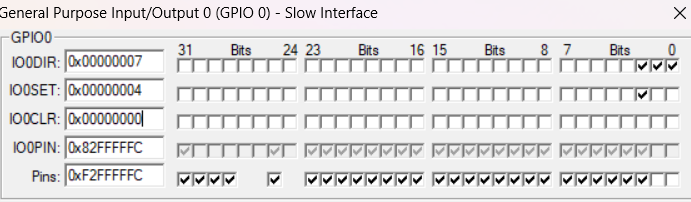
traffic\_light\_sequence();

// We should never reach here, as traffic\_light\_sequence runs indefinitely

return 0;

}





Code explanation:

1. **Pin Definitions**:
   * The code defines three macros for the GPIO pins corresponding to the red, yellow, and green LEDs of the traffic lights: **RED\_PIN**, **YELLOW\_PIN**, and **GREEN\_PIN**.
2. **Delay Function**:
   * **delay\_ms(unsigned int delay)**: This function is used to create a delay in milliseconds. It uses nested loops to create the delay.
3. **Traffic Light Initialization**:
   * **traffic\_light\_init()**: This function initializes the GPIO pins connected to the traffic lights.
   * It configures pins P0.0, P0.1, and P0.2 as output pins using the bitwise OR operation with **IODIR0**.
   * Initially, it turns off all the lights by setting the corresponding pins low using **IOCLR0**.
4. **Traffic Light Sequence**:
   * **traffic\_light\_sequence()**: This function controls the traffic lights according to a standard sequence: green light on for 5 seconds, yellow light on for 2 seconds, and red light on for 5 seconds.
   * It runs in an infinite loop (**while(1)**) to continuously cycle through the sequence.
   * Each step of the sequence involves turning off the previous light (if any) and turning on the next light using **IOCLR0** and **IOSET0**.
   * After turning on a light, it waits for the specified duration using the **delay\_ms()** function.
5. **Main Function**:
   * **main()**: This is the entry point of the program.
   * It first initializes the traffic light system by calling **traffic\_light\_init()**.
   * Then, it enters an infinite loop by calling **traffic\_light\_sequence()**.
   * Since the traffic light sequence runs indefinitely, the program will never reach the **return 0;** statement.

In summary, this code initializes and controls a traffic light system by configuring GPIO pins of the LPC2148 microcontroller to control the LEDs representing the traffic lights. It then cycles through a predefined sequence of turning on and off the lights, creating a simulated traffic light operation.