**EXPERMIMENT NO: 1**

**Subject: Embedded System Design**

**Subject code: ET 3271**

**Name:**

**Gr.No.:**

**Roll No.:**

**Class: TY Div : Batch:**

|  |  |
| --- | --- |
| **Title** | **:** LED and 7 segment Interfacing |
| **Aim** | **:** To interface 8 LEDs to LPC 2148 in common cathode mode. |
| **Objective** | : To Interface 8 LEDs to LPC 2148 and write an embedded C program to flash LEDs to produce different display pattern on it. |
| **Hardware** | : LPC2148 Development board, PC |
| **Software** | **:** Keil IDE Microvision, Philips LPC2000 flash utility. |

**Interfacing Diagram: ( draw and insert the diagram)**

**Theory:**

LPC 2148 features:

* 16/32-bit ARM7TDMI-S microcontroller in a 64 or 144 pin package.
* 32 kB on-chip Static RAM
* 500 kB on-chip Flash Program Memory
* On-chip crystal oscillator with an operating range of 1 MHz to 30 MHz.
* Two low power modes, Idle and Power-down.
* External 8, 16 or 32-bit bus (144 pin package only)
* In-System Programming (ISP) and In-Application Programming (IAP) via on-chip bootloader software.
* Two/four interconnected CAN interfaces with advanced acceptance filters.
* Four/eight channel (64/144 pin package) 10-bit A/D converter with conversion time as low as 2.44 ms.
* Two 32-bit timers/external event counters (with four capture and four compare channels each), PWM unit (six outputs) and watchdog
* Multiple serial interfaces including two UARTs (16C550), Fast I2C (400 kbits/s) and two

SPIs™.

* 60 MHz maximum CPU clock available from programmable on-chip Phase-Locked Loop.

**Registers Used:**

* **IODIR:**GPIO Port Direction control register.

This register is used to control the direction of the pins when they are configured as GPIO port pins. Direction bit for any pin must be set according to the pin functionality.

Direction control bits (0 = INPUT, 1 = OUTPUT).

* **IOSET:** This register is used to produce a HIGH level output at the port pins if they are configured as GPIO in an OUTPUT mode. Writing ‘1’produces a HIGH level at the corresponding port pins. Writing ‘0’has no effect. If any pin is configured as an input or a secondary function, writing to IOSET has no effect.

* **IOCLR:** This register is used to produce a LOW level at port pins if they are configured as GPIO in an OUTPUT mode. Writing 1 producesa LOW level at the corresponding port pins and clears the corresponding bits in the IOSET register. Writing 0 has no effect. If any pin is configured as an input or a secondary function, writing to IOCLR has no effect.

In ARM Microcontroller there are two 32-bit Timers, Timer0 and Timer1 which serves the purpose of generating time delays. Here are the registers used to program Timers in LPC2124 Microcontroller.

* **T0TCR:**This register is used to control the functions of the Timer0. Enable and reset operations of the Timer0 register can be controlled by this register.

* **T0PR:**This is a 32 bitPrescale register which holds the maximum value which the Prescale counter can take.

* **T0PC:**This is a 32 bitPrescale counter which specifies the division value of the processor clock before feeding it to the timers. The value in the register increments with every input pulse or Processor clock fed into it. Prescale register T0PR gives the max value this T0PC register can take. When reaching the max count T0PC register gets reset on the next Processor clock.

* **T0TC:**A 32 bit timer counter is incremented whenever Prescale counter T0PC value reaches its maximum level given in the Prescale register T0PR.

* **T0MR0:**These register stores the value which should be compared to the T0TC register. Operations specified based on the T0MCR will be performed whenever a match encounters.

* **T0MCR:**The Match control register is used to specify the operation whenever a match occurs between the value stored in T0MR0 and T0TC register.

* **T0IR:**Interrupt register which consists bits for match and capture interrupts. Writing will reset the interrupts in this register.

**Steps to program Timers:**

1. Reset timer0 initially to deactivate counting.
2. Load calculated values in the Prescaler register T0PR and Match register T0MR0.
3. Initialize T0PC and T0TC registers.
4. Select operations using match registers when match is encountered.
5. Start the Timer by enabling it through T0TCR register.
6. Wait till the interrupt and then clear the flag by writing T0IR register

**Algorithm: ( for LED interfacing)**

**Algorithm: ( for 7 segment interfacing)**

**Procedure:**

1. Create a new project, select LPC 2148 and write C program in an editor window
2. Compile and Link the program
3. Make .hex file
4. Download .hex file into the LPC2148 using LPC2000 flash utility. 5. Execute the program on the target board and observe the behavior.

**Program: ( insert C code)**

Code 1: LED Interfacing

#include <lpc21xx.h>

void delay(void)

{

unsigned int i;

unsigned int j;

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

{

for(j=0; j<100; j++);

}

return;

}

int main()

{

PINSEL0=0x00000000;

IODIR0 =0x007f8000;

while(1)

{

IOSET0=0x007f1000;

delay();

IOCLR0=0x007f1000;

delay();

}

}

Code 2: 7 segment display interfacing

#include <lpc21xx.h>

/\*

a= p0.16;

b= p0.17;

c= p0.18;

d= p0.19;

e= p0.20;

f= p0.21;

g= p0.22;

dot= p0.23

DIS1 p0.28

\*/

unsigned int delay, count=0, Switchcount=0;

unsigned int Disp[16] = {0x003f0000, 0x00060000, 0x005B0000, 0x004F0000, 0x00660000,

0x006D0000, 0x007D0000, 0x00070000, 0x007F0000, 0x006F0000,

0x00770000, 0x007C0000, 0x00390000, 0x005E0000, 0x00790000, 0x00710000};

int main(void)

{

PINSEL1 = 0x00000000;

IO0DIR = 0xF0FF0000;

while(1)

{

IO0SET = 0x10000000;

IO0CLR = 0x00FF0000;

for(delay =0; delay<100; delay++)

IO0SET = Disp[Switchcount];

for(delay =0; delay<1000000; delay++)

{}

Switchcount++;

if(Switchcount == 16)

{

Switchcount=0;

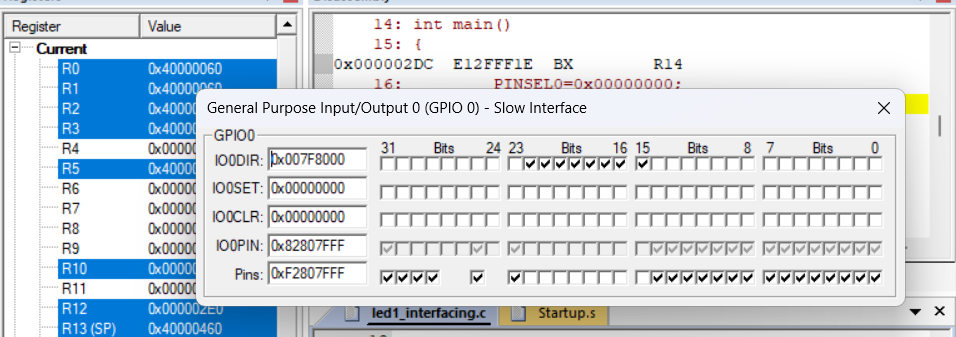
}

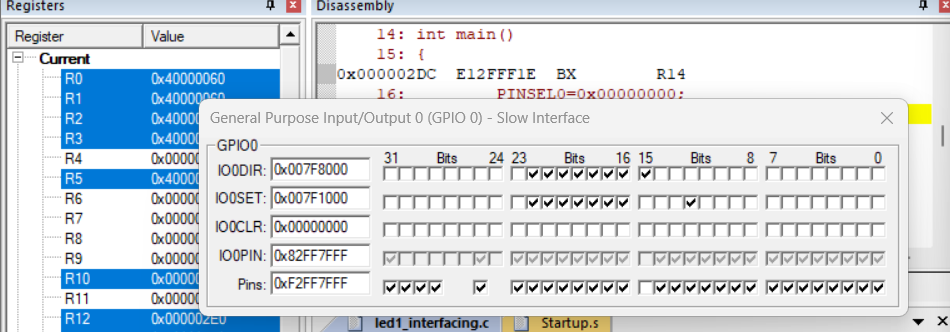
}

}

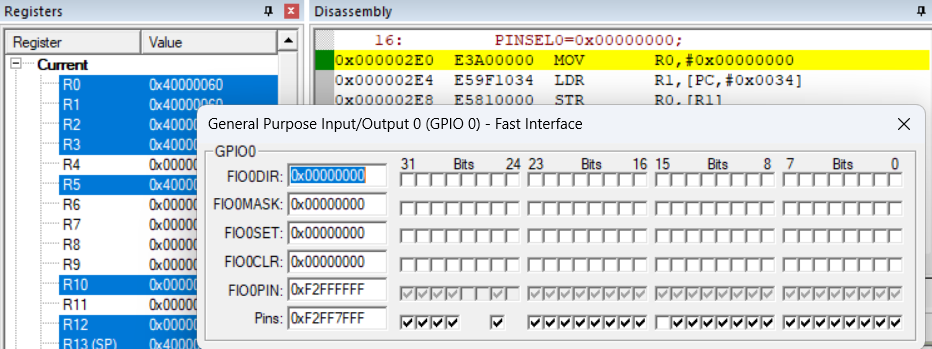
**Output :**

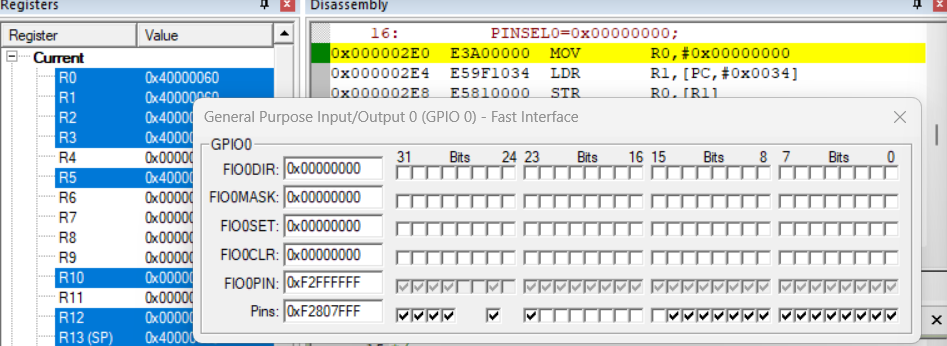
**led Interfacing**

****

****

****

****

****

**Conclusion:**

**As we upload the code for led interfacing all 8 led are start blinking at a time,. In case of 7 segment display all digits from 0 to 9 are visible after some delay.**