

Ayush Goyal

190905522 CSE D 62

DAA Lab-6 (Week 6) – Interfacing LED to ARM Microcontroller

Solved Exercise: Write a program to turn on/off the LEDs serially.

CODE:

```
#include<LPC17xx.h>

unsigned int i,j;
unsigned long LED = 0x00000010;

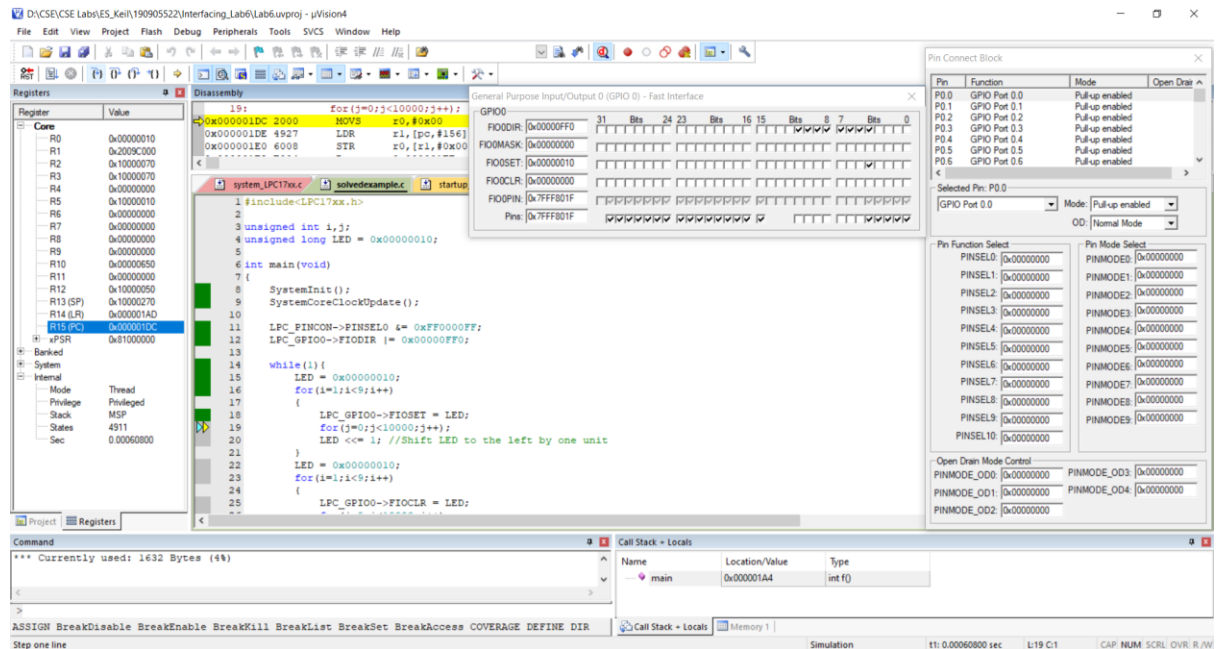
int main(void)
{
    SystemInit();
    SystemCoreClockUpdate();

    LPC_PINCON->PINSEL0 &= 0xFF0000FF;
    LPC_GPIO0->FIODIR |= 0x00000FF0;

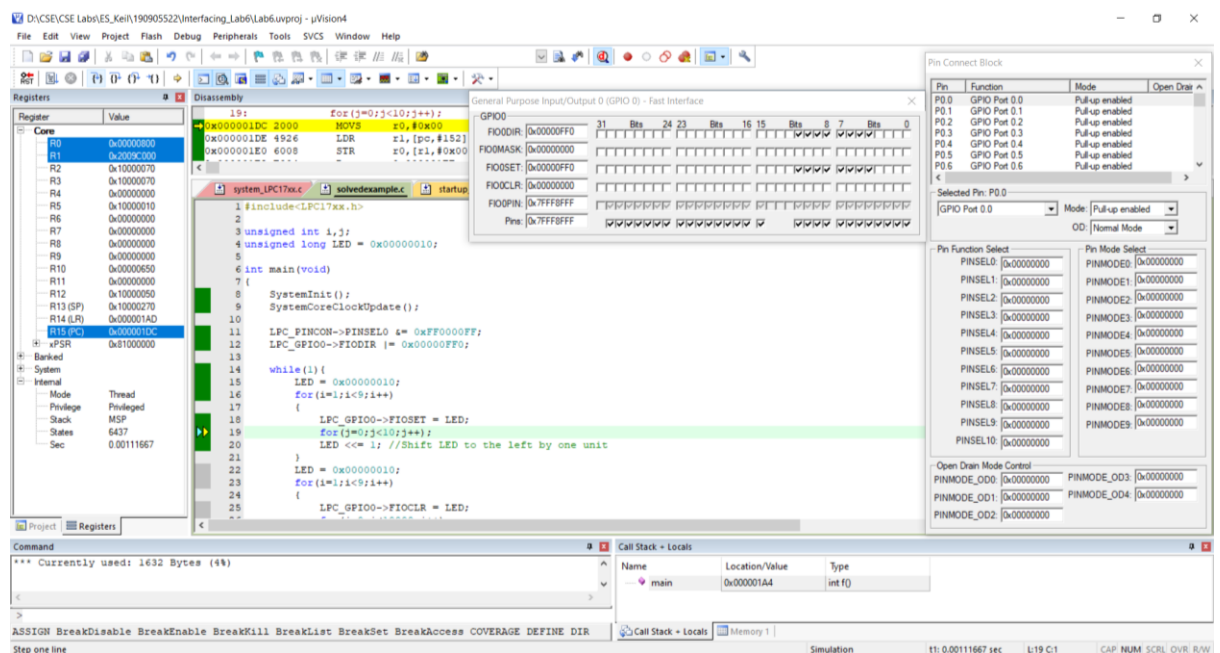
    while(1){
        LED = 0x00000010;
        for(i=1;i<9;i++){
            {
                LPC_GPIO0->FIOSET = LED;
                for(j=0;j<10000;j++);
                LED <<= 1; //Shift LED to the left by one unit
            }
            LED = 0x00000010;
            for(i=1;i<9;i++){
                {
                    LPC_GPIO0->FIOCLR = LED;
                    for(j=0;j<10000;j++);
                    LED<<=1;
                }
            }
        }
    }
}
```

OUTPUT:

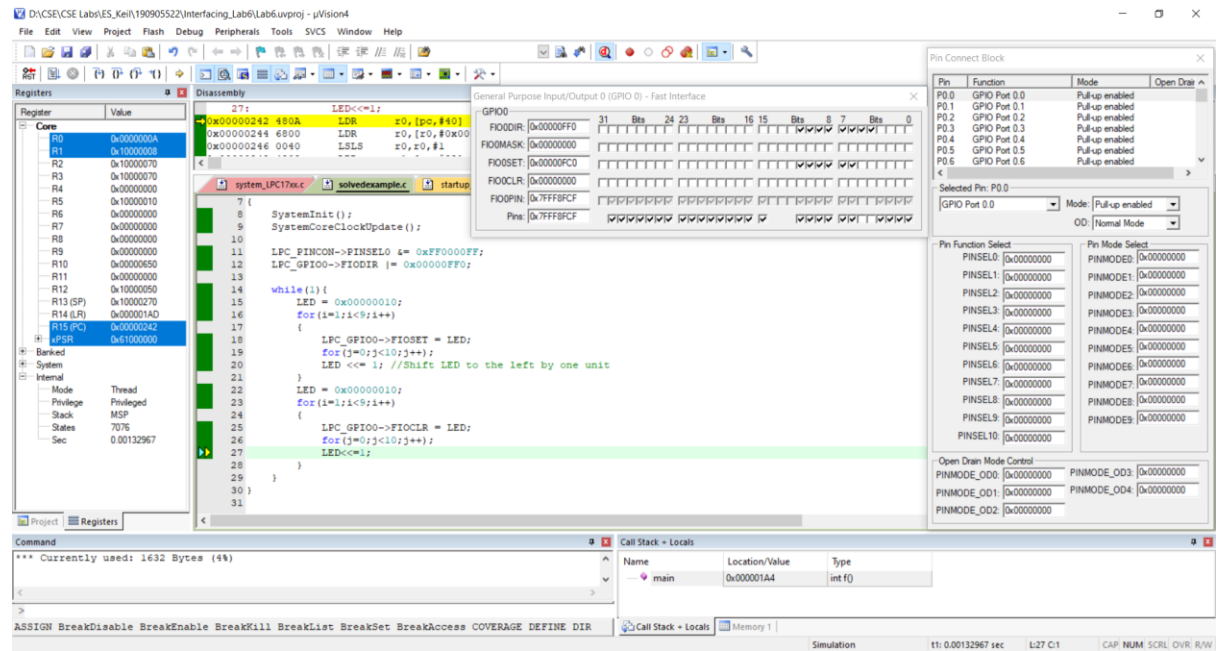
After the first LED is turned on:



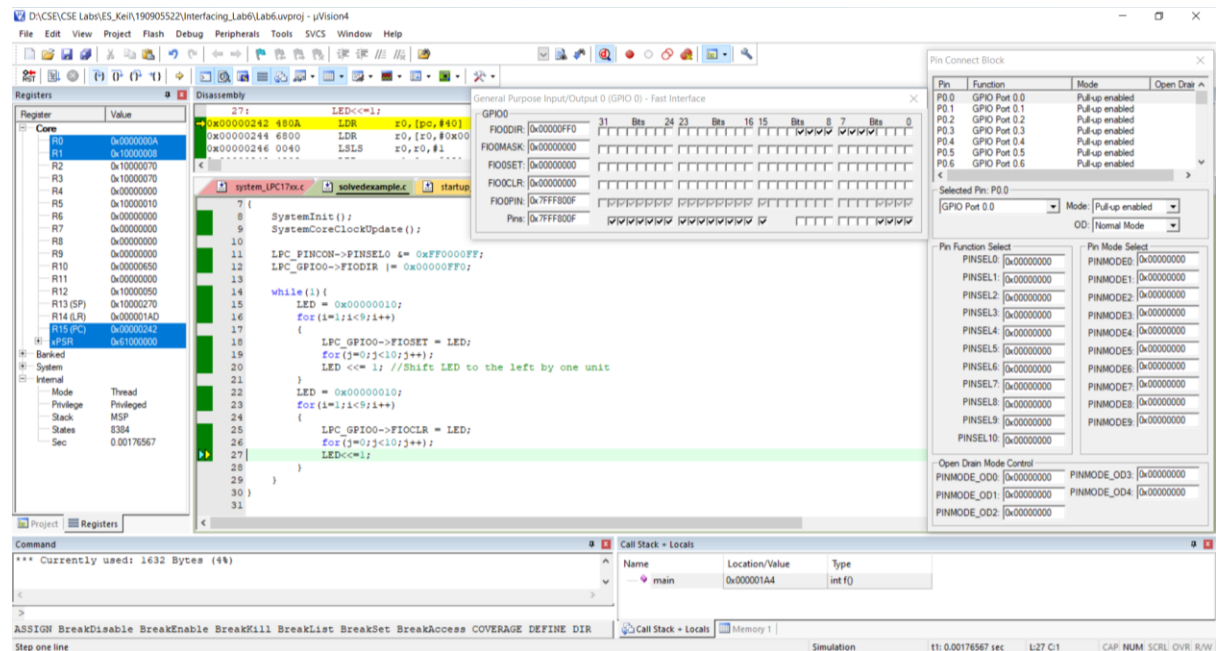
After the 8th LED is turned on:



After 1st and 2nd LED is turned off:



After 8 LEDs are turned off:



Thus, we can see that this while loop iterates continuously serially turning on and off the LEDs.

1) Write a C program to display an 8-bit binary up counter on the LEDs.

CODE:

```
#include<LPC17xx.h>

unsigned int i,j;

int main(void)
{
    SystemInit();
    SystemCoreClockUpdate();

    LPC_PINCON->PINSEL0 &= 0xFF0000FF;
    LPC_GPIO0->FIODIR |= 0x0000FF0; //configuring as output pins
    //using same pins P0.4 to P0.11 as GPIO functions

    while(1){
        for(i=0;i<256;i++)
        {
            LPC_GPIO0->FIOPIN = i<<4;
            for(j=0;j<10;j++);
        }
    }
}
```

OUTPUT:

After 1st count it should be just 4th bit on:

The screenshot displays the Keil uVision4 IDE interface. The main window shows the C program code. The Disassembly window is open, showing the assembly code for the first iteration of the while loop. The Pin Connect Block window is also open, showing the pin configuration for GPIO Port 0. The Command window at the bottom shows the current memory usage.

Disassembly Window:

Address	Disassembly	Comment
0x00000000	1: for(i=0;i<256;i++)	
0x00000001	MOV r0, #0x00	
0x00000002	LDR r1, [r0, #0]	
0x00000003	STR r0, [r1, #0x00]	
0x00000004	2: unsigned int i,j;	
0x00000005	3: int main(void)	
0x00000006	4: {	
0x00000007	5: SystemInit();	
0x00000008	6: SystemCoreClockUpdate();	
0x00000009	7: LPC_PINCON->PINSEL0 &= 0xFF0000FF;	
0x0000000A	8: LPC_GPIO0->FIODIR = 0x0000FF0;	//configuring as output pins
0x0000000B	9: //using same pins P0.4 to P0.11 as GPIO functions	
0x0000000C	10: while(1){	
0x0000000D	11: for(i=0;i<256;i++)	
0x0000000E	12: {	
0x0000000F	13: LPC_GPIO0->FIOPIN = i<<4;	
0x00000010	14: for(j=0;j<10;j++);	
0x00000011	15: }	
0x00000012	16: }	
0x00000013	17: }	
0x00000014	18: }	
0x00000015	19: }	
0x00000016	20: }	
0x00000017	21: }	
0x00000018	22: }	

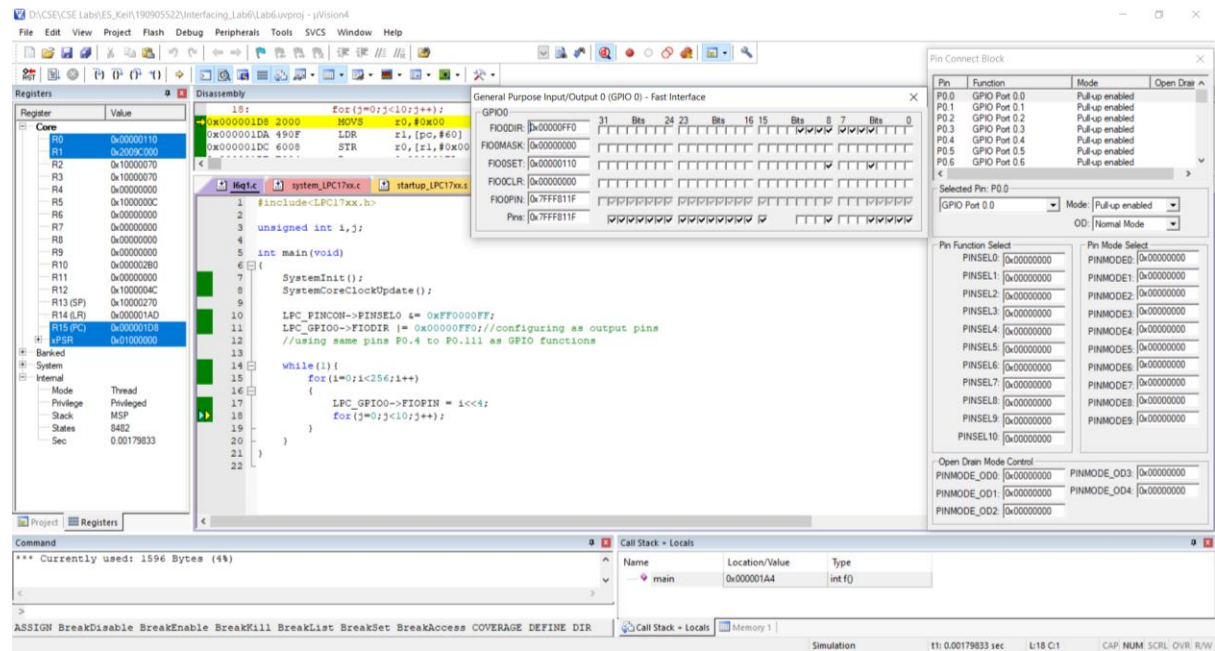
Pin Connect Block Window:

Pin	Function	Mode	Open Drain
P0.0	GPIO Port 0.0	Pull-up enabled	
P0.1	GPIO Port 0.1	Pull-up enabled	
P0.2	GPIO Port 0.2	Pull-up enabled	
P0.3	GPIO Port 0.3	Pull-up enabled	
P0.4	GPIO Port 0.4	Pull-up enabled	
P0.5	GPIO Port 0.5	Pull-up enabled	
P0.6	GPIO Port 0.6	Pull-up enabled	

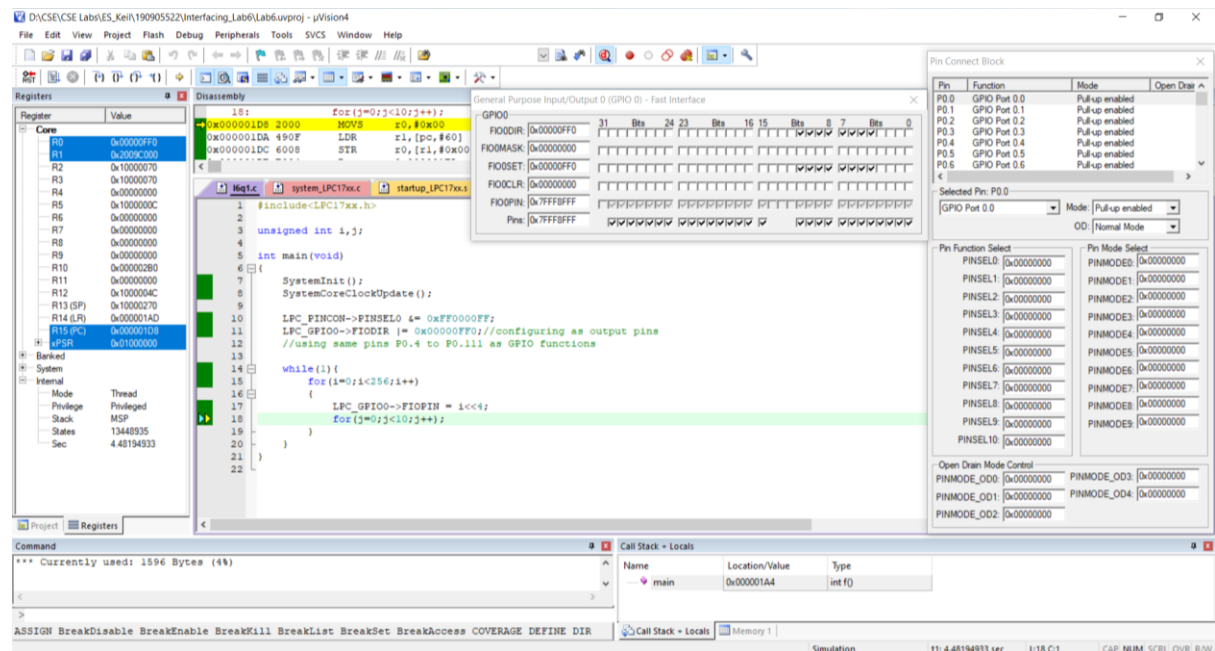
Command Window:

```
*** Currently used: 1596 Bytes (4%)
```

After 18 counts or iterations, the binary counter should be 10001:



After 255 counts, the binary up counter should be at value 11111111:



Thus, we can see the functioning of an 8-bit binary up counter on the LEDs.

- 2) Write a C program to read a key and display an 8-bit up/down counter on the LEDs.
Hint: Use key SW2(if SW2=1, up counter else down counter), which is available at CNB1 pin 7. Connect CNB1 to any controller connector like CNB, CNC, etc. Configure the corresponding port pin as GPIO using corresponding PINSEL register and as input pin using corresponding FIODIR register.

CODE:

```
#include <LPC17xx.h>

unsigned int i,j,k;
int c=0;

int main(void)
{
    //i am configuring port0 pins p0.4-p0.11 as gpio function
    LPC_PINCON->PINSEL0 &= 0xFF0000FF;
    LPC_PINCON->PINSEL4 &= 0xFCFFFFFF;
    //2.12 gpio needs to be configured as input pin
    LPC_GPIO0->FIODIR |= 0x00000FF0; //configuring as output pins
    LPC_GPIO2->FIODIR &= 0xFFFFEFFF; //pin 12 is made to be 0 since i/p

    while(1){
        k = LPC_GPIO2->FIOPIN >> 12; //We read input from 2.12
        k &= 0x00000001;
        if(k==1)
            c++;
        else
            c--;
        if(c== -1)
            c=255;
        if(c==256)
            c=0;
        //left shifting by four because fourth bit is the starting bit
        LPC_GPIO0->FIOPIN=c<<4;
        //Putting a random delay here
        for(j=0;j<5000;j++);
    }
}
```


OUTPUT:

12th bit of the GPIO2 pin is turned on at this moment therefore we can see in GPIO0 that it acts as an up counter and the output is shown after 18 iterations therefore 10001.

The screenshot shows the uVision4 IDE with the following components:

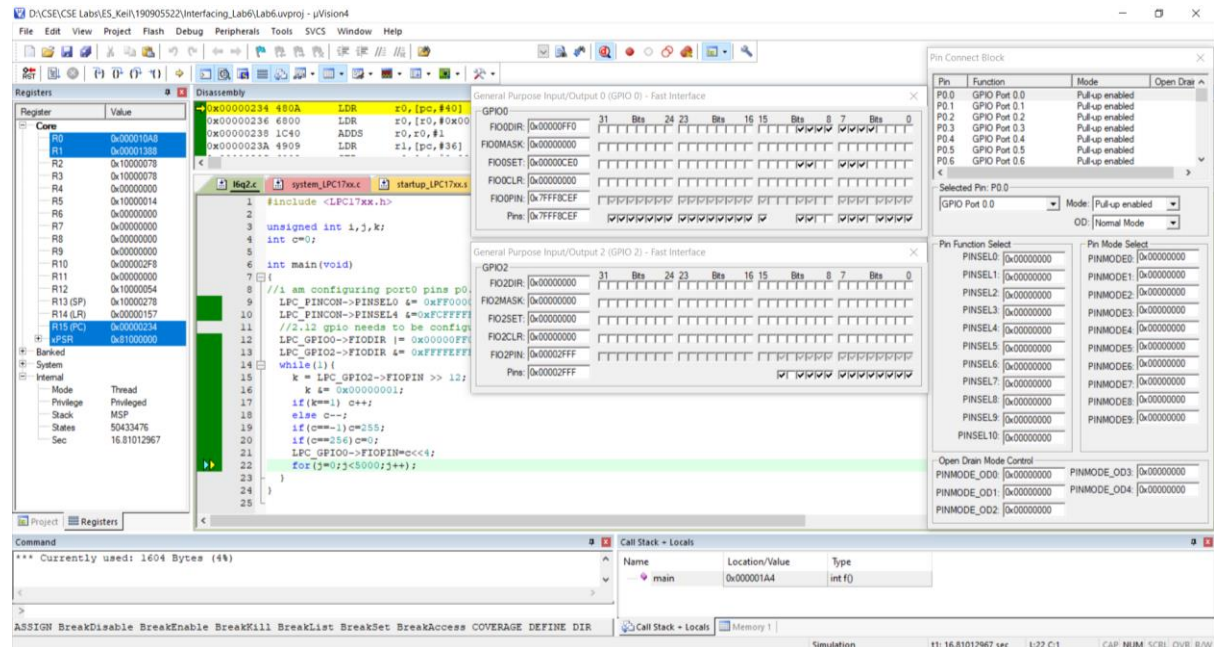
- Registers:** Shows the current state of the processor registers. R0 is 0x00001150, R1 is 0x00001150, R2 is 0x0000078, R3 is 0x0000078, R4 is 0x00000000, R5 is 0x00000000, R6 is 0x00000000, R7 is 0x00000000, R8 is 0x00000000, R9 is 0x00000000, R10 is 0x000002F8, R11 is 0x00000000, R12 is 0x00000054, R13 (SP) is 0x00000278, R14 (LR) is 0x00000157, R15 (PC) is 0x00000238, and xPSR is 0x01000000.
- Disassembly:** Shows the assembly code for the main function. The code is configuring GPIO0 and GPIO2. The current instruction is `0x00000238 IC40 ADDS r0,r0,#1`.
- General Purpose Input/Output 0 (GPIO 0) - Fast Interface:** Shows the configuration for GPIO0. The pins are P0.0 to P0.6, all with pull-up enabled. The pin P0.0 is selected.
- General Purpose Input/Output 2 (GPIO 2) - Fast Interface:** Shows the configuration for GPIO2. The pins are P2.0 to P2.7, all with pull-up enabled. The pin P2.0 is selected.
- Pin Connect Block:** Shows the pin configuration for GPIO0. The pins are P0.0 to P0.6, all with pull-up enabled. The pin P0.0 is selected.
- Call Stack - Locals:** Shows the call stack with the main function at location 0x00001A4.

The switch remaining on, output is shown after 256 iterations of the up counter therefore showing 1111 1111:

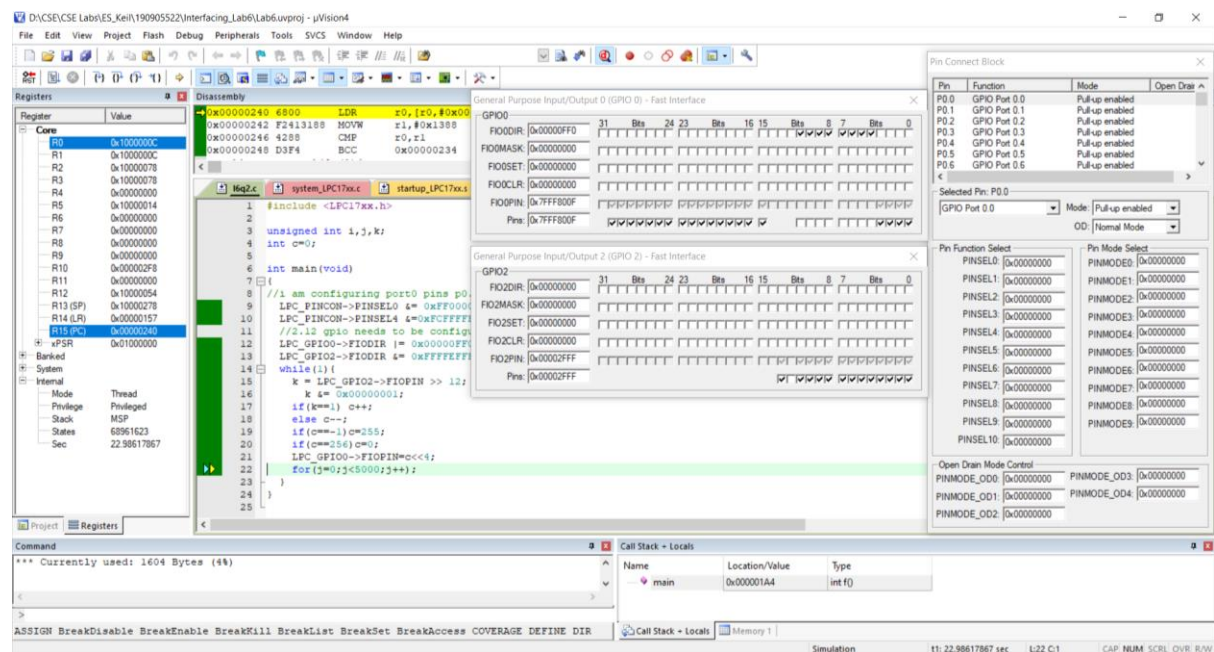
The screenshot shows the uVision4 IDE with the following components:

- Registers:** Shows the current state of the processor registers. R0 is 0x00000145, R1 is 0x0000000C, R2 is 0x0000078, R3 is 0x0000078, R4 is 0x00000000, R5 is 0x00000000, R6 is 0x00000000, R7 is 0x00000000, R8 is 0x00000000, R9 is 0x00000000, R10 is 0x000002F8, R11 is 0x00000000, R12 is 0x00000054, R13 (SP) is 0x00000278, R14 (LR) is 0x00000157, R15 (PC) is 0x00000242, and xPSR is 0x01000000.
- Disassembly:** Shows the assembly code for the main function. The code is incrementing the counter. The current instruction is `0x00000242 F24313B9 NOPW r1,r0,13B9`.
- General Purpose Input/Output 0 (GPIO 0) - Fast Interface:** Shows the configuration for GPIO0. The pins are P0.0 to P0.6, all with pull-up enabled. The pin P0.0 is selected.
- General Purpose Input/Output 2 (GPIO 2) - Fast Interface:** Shows the configuration for GPIO2. The pins are P2.0 to P2.7, all with pull-up enabled. The pin P2.0 is selected.
- Pin Connect Block:** Shows the pin configuration for GPIO0. The pins are P0.0 to P0.6, all with pull-up enabled. The pin P0.0 is selected.
- Call Stack - Locals:** Shows the call stack with the main function at location 0x00001A4.

After the 12th bit of the GPIO2 register is unselected, the switch is turned off and the pins from 0.4 to 0.11 in GPIO0 act as a down counter and it shows the value after a few iterations of counting down from 1111 1111:



The switch bit remaining off we can see the down counter goes all the way to 0000 0000 after 256 iterations starting from 1111 1111. It will loop back as well.



3) Write a program to simulate an 8-bit ring counter with key press (SW2).

CODE:

```
#include <LPC17xx.h>

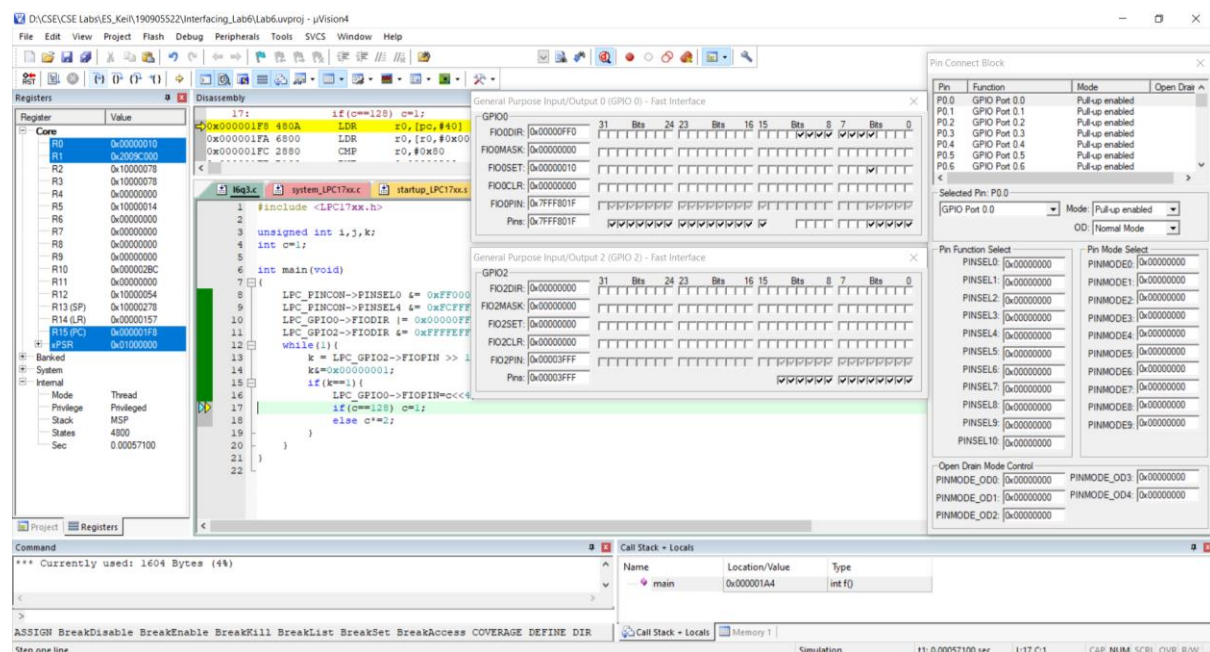
unsigned int i,k;
int c=1;

int main(void)
{
    LPC_PINCON->PINSEL0 &= 0xFF0000FF; //pin 0.4-0.11 as gpio function
    LPC_PINCON->PINSEL4 &= 0xFCFFFFFF; //for pin 2.12 gpio2 as switch
    LPC_GPIO0->FIODIR |= 0x00000FF0; //config as output pin
    LPC_GPIO2->FIODIR &= 0xFFFFEFFF; //pin 12 is made to be 0 since i/p

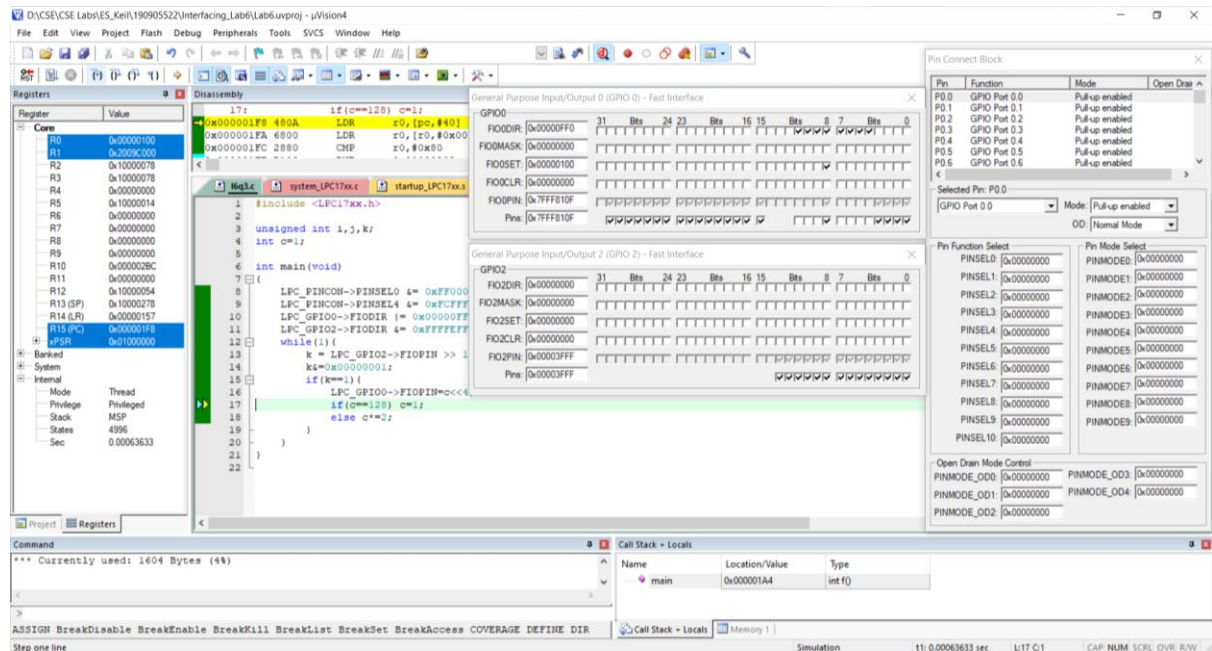
    while(1){
        k = LPC_GPIO2->FIOPIN >> 12;
        k&=0x00000001; //reads switch press
        if(k==1){
            LPC_GPIO0->FIOPIN=c<<4; //since starts from bit 4 to bit 11
            if(c==128) c=1; //to reset the ring counter
            else c*=2; //shifting by 1 bit
        }
        for(i=0;i<5000;i++); //random delay
    }
}
```

OUTPUT:

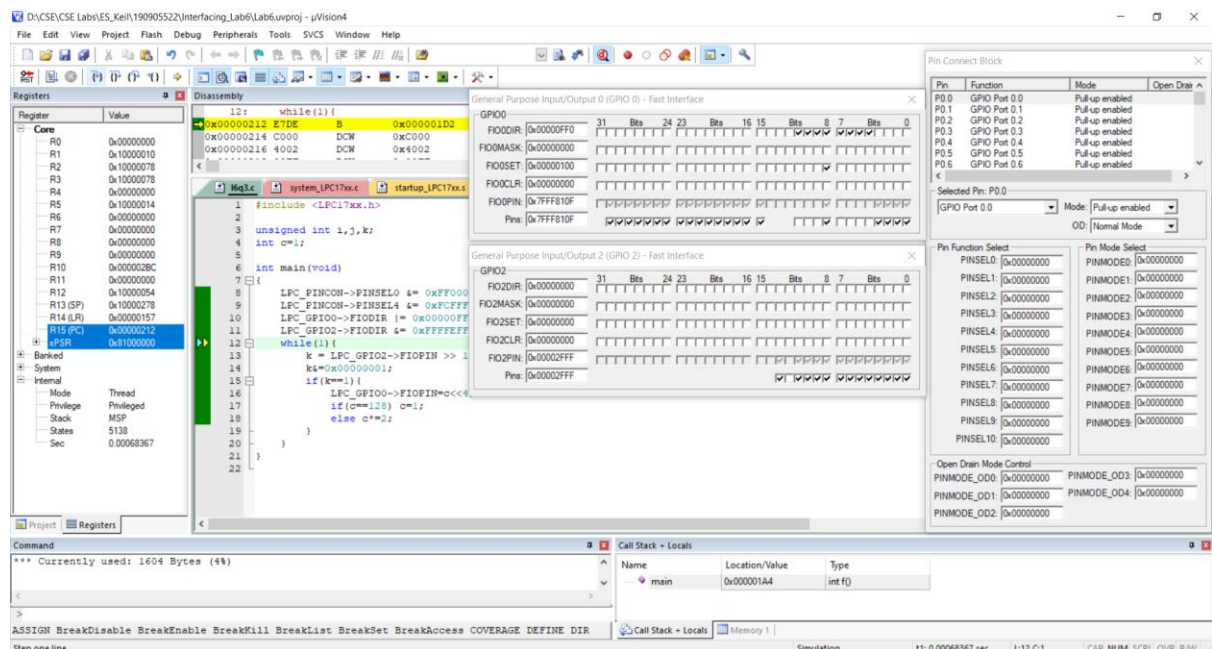
When the switch is pressed and the loop executes one time the output should be 0000 0001:



When the switch is pressed, and the loop is executed 5 times the output should be 0001 0000:



When the switch is not pressed, the 12th bit of the GPIO2 is unselected then the counter is stuck at the previous state 0001 0000 and will not start until the switch is pressed again:



The 8-bit ring counter goes till 1000 0000 and then back to 0000 0001 when the switch is pressed.

THE END