Experiment No.: 5

Learn and understand how to configure MSP-EXP430G2 digital I/O pins. Write a C program for configuration of GPIO ports for MSP430 (blinking LEDs, push buttons interface).

Roll No.: ________ *Batch:* _________

Date of Performance: ________________

Date of Assessment: __________________

Particulars	**Marks**
Regularity (05)	
Practical Conduction (10)	
Program Output (05)	
Understanding (Viva) (05)	
Total (25)	
Signature of Course Teacher	

Experiment: 5

AIM: Learn and understand how to configure MSP-EXP430G2 digital I/O pins. Write a C program for configuration of GPIO ports for MSP430 (blinking LEDs, push buttons interface).

PREREQUISITE:

Before starting this lab, students should be familiar with:

- Basics of MSP430 architecture and GPIO configuration
- LED and push-button interfacing techniques.
- Energia IDE.
- Basic C programming concepts.

OBJECTIVE:

By the end of this experiment, students will be able to:

- Configure MSP430 GPIO pins as input and output.
- Blink on-board LEDs using software delay.
- Interface push-button and control LEDs based on button input.
- Modify delay and LED blinking patterns.

HARDWARE REQUIREMENTS:

- MSP-EXP430G2 LaunchPad (MSP430G2553)
- USB Cable
- Computer or Laptop with USB Port

SOFTWARE REQUIREMENTS:

- Energia IDE
- MSP430 USB Drivers / MSPDebug / UniFlash Tool

THEORY:

1) Introduction to MSP430 and GPIO:

The MSP430 is a 16-bit, ultra-low-power microcontroller designed by Texas Instruments for embedded applications requiring low power and high performance. It features several peripherals, including timers, ADCs, communication interfaces, and general-purpose input/output (GPIO) ports. GPIOs are versatile pins that can function as either input or output depending on the configuration of their registers.

Each GPIO pin is controlled by specific registers that define its direction, output value, and function. The main registers used are PxDIR, PxOUT, PxIN, PxREN, PxSEL, and PxSEL2.

• PxDIR: Determines the direction of the pin (0 = Input, 1 = Output).

• PxOUT: Controls the output value when configured as output; also selects pull-up or pull-down when configured as input.

• PxIN: Reads the logic level on the pin (1 = High, 0 = Low).

• PxREN: Enables or disables the internal resistor network.

• PxSEL/PxSEL2: Selects between GPIO and peripheral functions.

2) LED Interfacing:

The MSP-EXP430G2 LaunchPad has two onboard LEDs connected to P1.0 (Red) and P1.6 (Green) through current-limiting resistors. To make these LEDs work, the pins must be configured as outputs by setting the corresponding bits in the P1DIR register. The P1OUT register controls whether the LED is ON (logic 1) or OFF (logic 0). LEDs can be made to blink by repeatedly toggling the output using an XOR operation (P1OUT ^= BIT0).

3) Push Button Interfacing:

The push button S2 on the LaunchPad is connected to pin P1.3. It is active-low, meaning it reads logic 0 when pressed and logic 1 when released. The pin is configured as input by clearing its bit in P1DIR and enabling the internal pull-up resistor by setting the respective bit in P1REN and P1OUT.
Mechanical buttons tend to bounce when pressed, causing false triggering. Software debouncing (introducing small delays) helps avoid this issue.

4) Delay Generation:

Delays are created in software using intrinsic functions such as __delay_cycles(n), where n represents the number of clock cycles to wait. For example, at 1 MHz clock frequency, __delay_cycles(1000000) provides an approximately 1-second delay. For accurate timing, hardware timers like Timer_A are preferred.

5) Working Principle:

In this experiment, P1.0 and P1.6 are configured as output pins for LEDs, and P1.3 is configured as an input pin for the push button. The LEDs are toggled in an infinite loop using a delay function to observe blinking. The button controls whether the LED is ON or OFF. The delay value can be adjusted to change the blink rate.

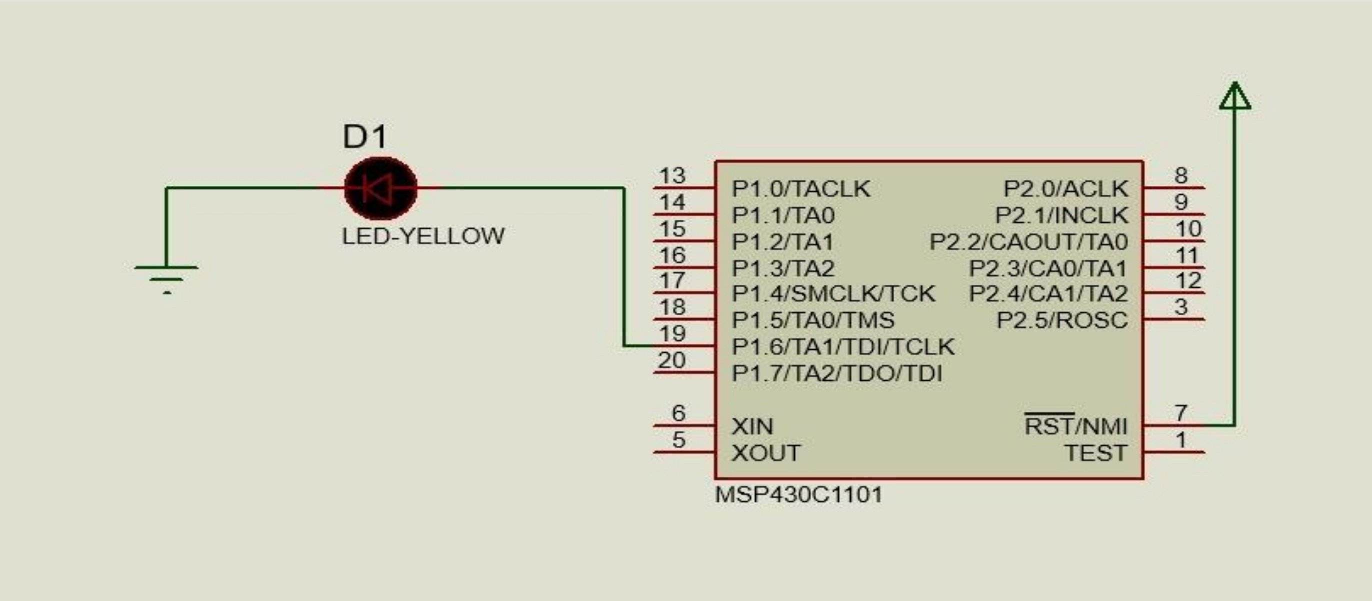
6) Pull-up and Pull-down Resistors:

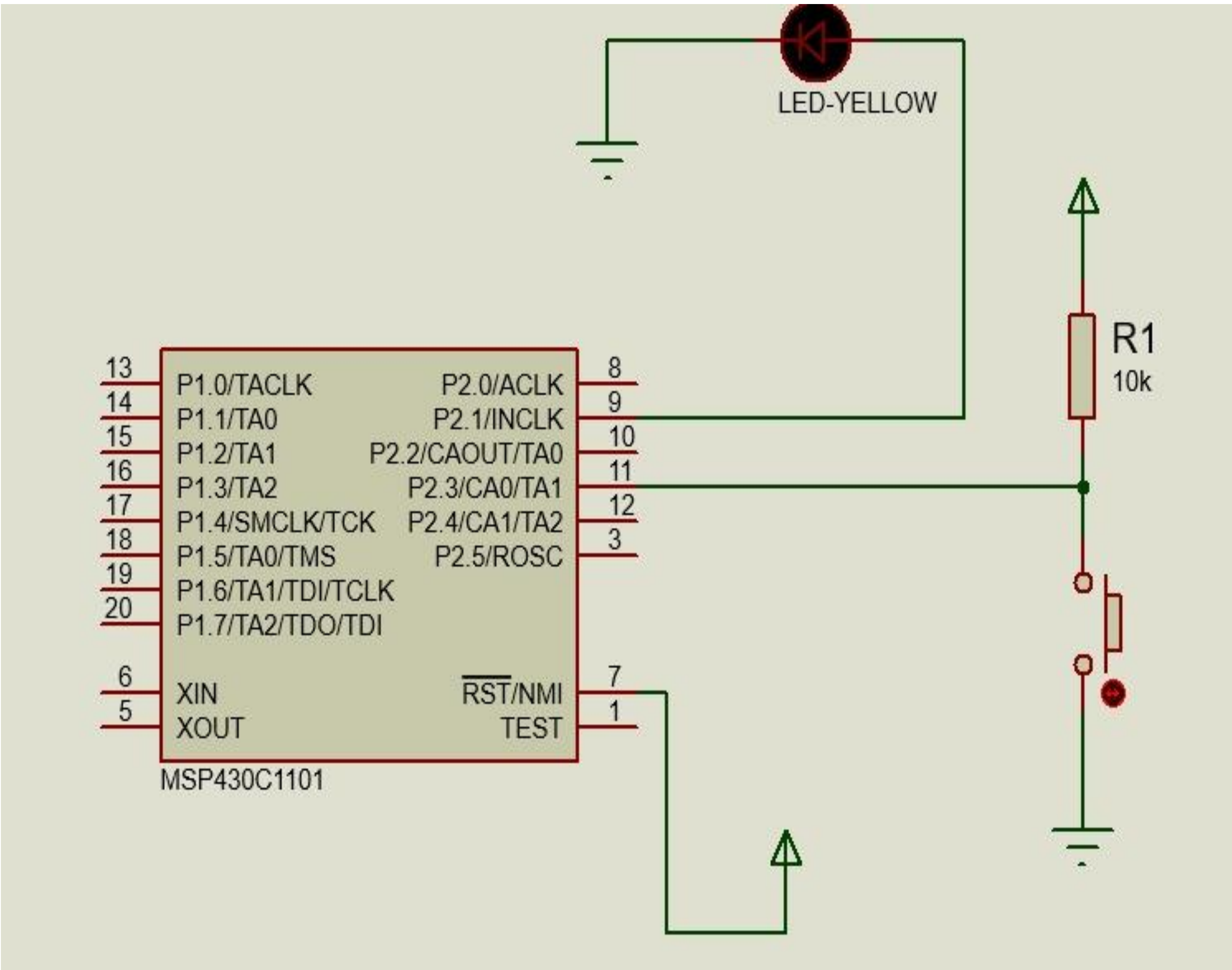
MSP430 GPIO pins include internal pull-up or pull-down resistors to ensure stable logic levels on input pins. These are enabled via PxREN and configured through PxOUT. Setting PxOUT to 1 enables pull-up; setting it to 0 enables pull-down.

7) Power and Low-Power Modes:

The MSP430 supports multiple low-power modes to reduce energy consumption. While blinking LEDs, the MCU can remain active, but in advanced applications, it should enter low-power states between operations to save power.

CIRCUIT DIAGRAM:





PROCEDURE:

- Open Code Composer Studio and create a new MSP430 project.
- Select MSP430G2553 device and set compiler settings.
- Write the GPIO configuration code.
- Build and debug the project to generate output.
- Flash the program into the LaunchPad.
- Observe LED blinking and button operation.

SAMPLE CODE :

Attach separate sheet for code

CONCLUSION:

Experiment No.: 6

PWM-based LED Brightness Control using MSP430G2553

Roll No.: _________ Batch: __________
Date of Performance: _________________
Date of Assessment: ___________________

Particulars	Marks
Regularity (05)	
Practical Conduction (10)	
Program Output (05)	
Understanding (Viva) (05)	
Total (25)	
Signature of Course Teacher	

Experiment: 6

AIM: Implement Pulse Width Modulation to control the brightness of the on-board, green LED. This experiment will help you to learn and understand the configuration of PWM and Timer peripherals of the MSP430G2553.
Exercises: a) Observe the PWM waveform on a particular pin using CRO.

PREREQUISITE:

Before starting this lab, students should be familiar with:

- Basics of MSP430 architecture and GPIO configuration
- Operation of Timer_ A module in MSP430.
- Concept of Pulse Width Modulation (PWM).
- Energia IDE.
- Basic C programming concepts.

OBJECTIVE:

By the end of this experiment, students will be able to:

- Understand the concept of PWM and its applications.
- Configure Timer_A module for PWM signal generation.
- Vary the duty cycle to control LED brightness.
- Observe PWM waveform on an oscilloscope (CRO).

HARDWARE REQUIREMENTS:

- MSP-EXP430G2 LaunchPad (MSP430G2553)
- CRO (Cathode Ray Oscilloscope)
- USB Cable, Connecting wires / jumper cables
- Computer or Laptop with USB Port

SOFTWARE REQUIREMENTS:

- Energia IDE / Code Composer Studio (CCS)
- MSP430 USB Drivers / MSPDebug / UniFlash Tool

THEORY:

1) Introduction to PWM:

Pulse Width Modulation (PWM) is a technique used to control analog quantities using digital signals. It involves varying the width of the pulse (ON time) while keeping the frequency constant. The ratio of ON time to total time (period) is called the duty cycle. By changing the duty cycle, the average voltage and thus the power delivered to a load can be controlled.

2) Duty Cycle:

Duty cycle determines the proportion of time the signal stays high during one period. For example, at a 50% duty cycle, the signal is high half of the time and low for the other half, resulting in half brightness in an LED.

3) PWM in MSP430:

MSP430G2553 includes Timer_A peripherals that can be configured for PWM generation. Each timer has Capture/Compare registers (CCR0, CCR1, CCR2) which define the PWM period and duty cycle. The PWM signal is generated automatically in hardware without CPU intervention.

4) Timer_A Configuration:

• Select clock source (SMCLK or ACLK).

• Choose Up mode for counting.

• Set CCR0 = PWM period value.

• Set CCR1 = duty cycle value.

• Configure output mode (OUTMOD_7) for Reset/Set PWM mode.

5) Working Principle:

The Timer counts from 0 to CCR0 repeatedly. When the counter matches CCR1, the output toggles, generating a PWM waveform. The green LED connected to P1.6 (TA0.1) changes brightness based on CCR1 value. A larger CCR1 value increases ON time and thus LED brightness.

6) Mathematical Relationship:

PWM Period = (CCR0 + 1) / Timer Clock Frequency

Duty Cycle (%) = (CCR1 / CCR0) × 100

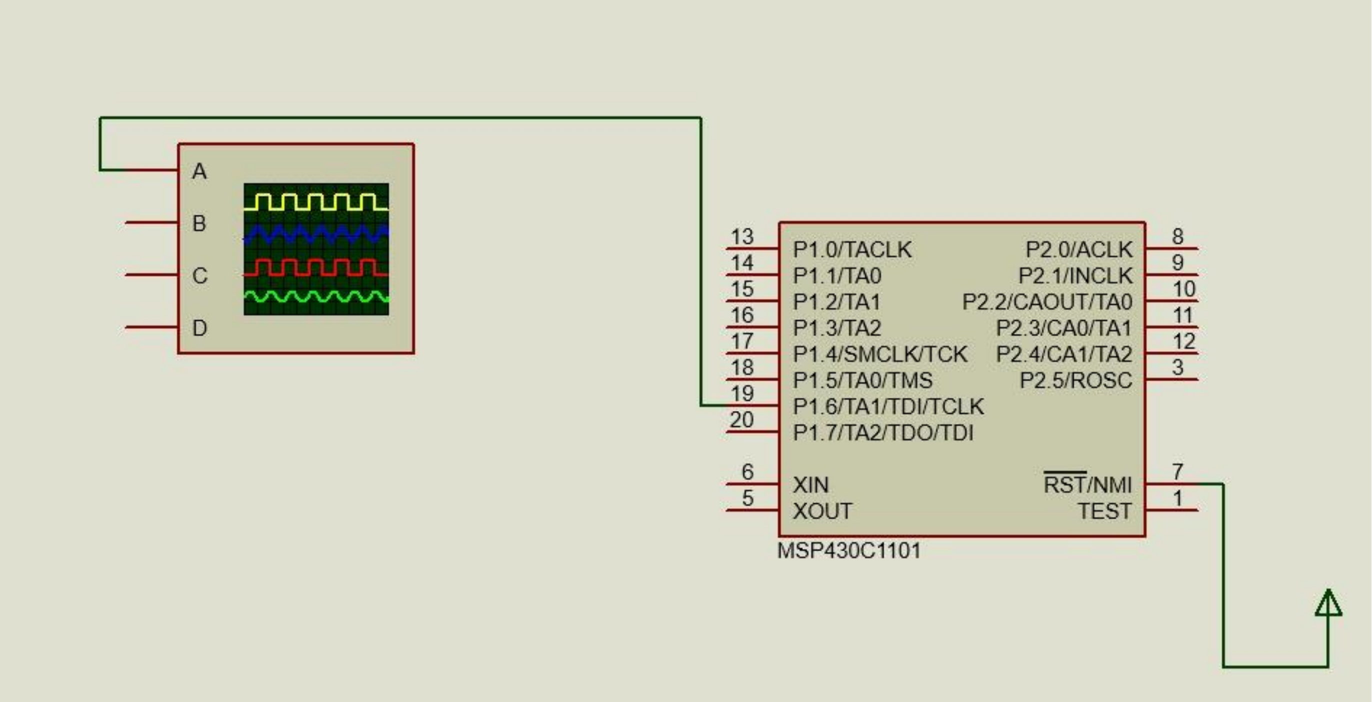
7) Observation:

By varying CCR1, the LED brightness can be increased or decreased. The PWM waveform can also be viewed on CRO by connecting the probe to P1.6.

8) Applications:

PWM is used for controlling LED brightness, DC motor speed, voltage regulation, and signal generation.

CIRCUIT DIAGRAM:



PROCEDURE:

- Open Code Composer Studio and create a new project for MSP430G2553.
- Configure device and include <msp430.h>.
- Write the code to configure Timer_A for PWM generation.
- Build and flash the code to the MSP430 LaunchPad.
- Observe the brightness variation of the green LED.
- Connect CRO probe to P1.6 to observe PWM waveform.

SAMPLE CODE :

Attach separate sheet for code

CONCLUSION:

Experiment No.: 7

Interface IR sensor with MSP430G2553 to detect intruder and turn on buzzer.

Roll No.: _________ Batch: __________

Date of Performance: _________________

Date of Assessment: ___________________

Particulars	Marks
Regularity (05)	
Practical Conduction (10)	
Program Output (05)	
Understanding (Viva) (05)	
Total (25)	
Signature of Course Teacher	

Experiment: 7

AIM: Interface IR sensor with MSP430G2553 to detect intruder and turn on buzzer.

PREREQUISITE:

Before starting this lab, students should be familiar with:

- Basics of MSP430 microcontroller architecture and GPIO configuration.
- Working principle of IR sensor modules.
- Digital input/output configuration in C.
- Interfacing of sensors and actuators (like buzzers) with microcontrollers.

OBJECTIVE:

By the end of this experiment, students will be able to:

- Understand the operation of IR sensor and its output characteristics.
- Configure MSP430 GPIO pins for sensor input and actuator output.
- Write a C program to detect the intruder using IR sensor.
- Activate the buzzer when the intruder is detected.

HARDWARE REQUIREMENTS:

- MSP-EXP430G2 LaunchPad (MSP430G2553)
- IR Sensor Module
- Active Buzzer
- Connecting wires / jumper cables
- Breadboard
- USB Cable
- Computer / Laptop

SOFTWARE REQUIREMENTS:

- Energia IDE / Code Composer Studio (CCS)
- MSP430 USB Drivers / MSPDebug / UniFlash Tool

THEORY:

1) Introduction to IR Sensor:

Infrared (IR) sensors are electronic devices that emit and/or detect infrared radiation to sense objects or measure distances. An IR sensor module typically consists of an IR LED (emitter) and a photodiode (receiver). The photodiode detects reflected IR light from nearby objects. When an object comes within the range, the reflected IR radiation falls on the photodiode, changing its output voltage.

There are two main types of IR sensors:

• Active IR Sensor: Has both emitter and detector in a single module (used for obstacle/intruder detection).

• Passive IR Sensor (PIR): Detects IR radiation from human bodies.

In this experiment, an active IR sensor is used. It gives a digital output — logic HIGH (1) when no obstacle is present and logic LOW (0) when an obstacle/intruder is detected.

2) Working Principle:

When an intruder or object passes in front of the sensor, IR rays emitted from the LED reflect back and are received by the photodiode. The sensor output changes state, which can be read by the MSP430 input pin. The microcontroller then triggers an output signal to drive a buzzer.

3) Interfacing with MSP430:

The IR sensor output pin is connected to one of the GPIO pins of the MSP430 configured as input (e.g., P1.3). The buzzer is connected to an output pin (e.g., P1.0). When the sensor detects an intruder (logic LOW), the microcontroller sets the output pin HIGH to turn on the buzzer.

Typical connections:

• IR Sensor Output → P1.3 (Input)

• Buzzer → P1.0 (Output)

• VCC → 5V, GND → Ground

4) MSP430 GPIO Operation:

To configure GPIO pins:

• PxDIR register: 0 for input, 1 for output.

• PxIN register: reads logic level from input pin.

• PxOUT register: sets or clears output pin.

Example:

```
P1DIR &= ~BIT3; // Configure P1.3 as input
P1DIR |= BIT0;  // Configure P1.0 as output
if ((P1IN & BIT3) == 0) P1OUT |= BIT0; // Turn ON buzzer when intruder detected
```

5) Buzzer Working:

A buzzer converts electrical energy into sound. When its control pin is set HIGH by the microcontroller, the buzzer produces an audible tone. In this experiment, the buzzer remains OFF when no intruder is detected and turns ON when the IR sensor output is LOW.

6) Applications:

• Intruder detection systems

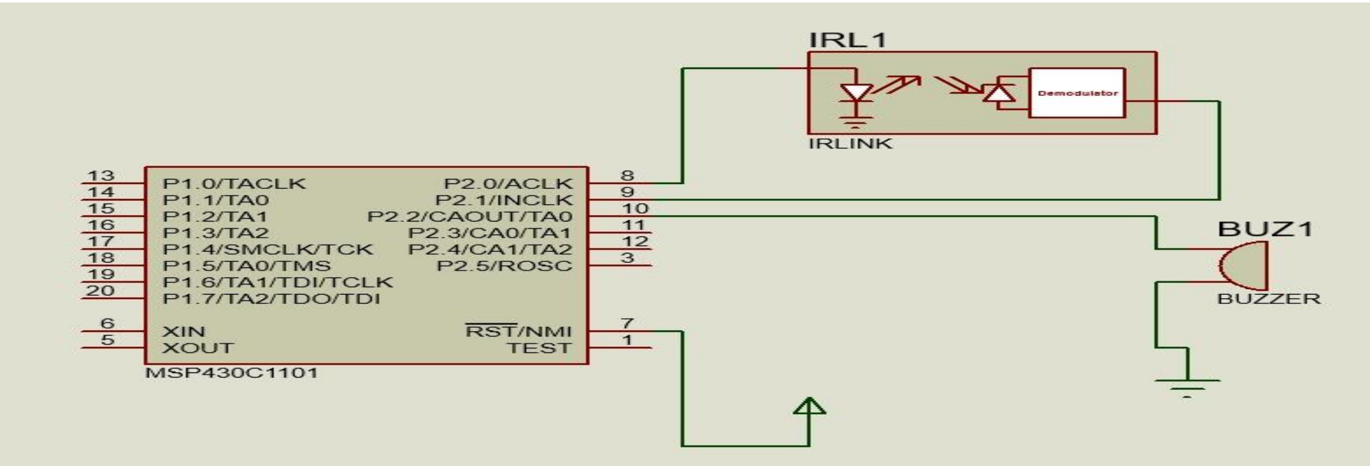
• Obstacle detection in robotics

• Automatic door control

• Line-following robots

• Object counting systems

CIRCUIT DIAGRAM:



PROCEDURE:

- Connect the IR sensor output to P1.3 and the buzzer to P1.0 on the MSP430 LaunchPad.
- Open Code Composer Studio and create a new project for MSP430G2553.
- Configure the project and include <msp430.h>.
- Write the C program for interfacing IR sensor and buzzer.
- Build and flash the program to the LaunchPad.
- Move an object in front of the IR sensor to simulate an intruder.
- Observe that the buzzer turns ON when an intruder is detected and turns OFF when there is no object.

SAMPLE CODE :

Attach separate sheet for code

CONCLUSION:

Experiment No.: 8

Interface a relay with MSP430G2553 and write an embedded C program to turn ON and OFF relay and DC motor.

Roll No.: __________ Batch: ___________

Date of Performance: ___________________

Date of Assessment: ______________________

Particulars	Marks
Regularity (05)	
Practical Conduction (10)	
Program Output (05)	
Understanding (Viva) (05)	
Total (25)	
Signature of Course Teacher	

Experiment: 8

AIM: Interface a relay with MSP430G2553 and write an embedded C program to turn ON and OFF relay and DC motor.

PREREQUISITE:

Before starting this lab, students should be familiar with:

- MSP430 GPIO programming and C code structure.
- Relay and transistor switch operation.
- Flyback diode protection for inductive loads.
- DC motor operation principles.
- Safety when interfacing external components with microcontrollers.

OBJECTIVE:

By the end of this experiment, students will be able to:

- Understand relay and transistor driver circuits.
- Interface relay and DC motor with MSP430.
- Write embedded C code for turning ON/OFF a relay and DC motor.
- Apply safe interfacing practices for inductive loads.

HARDWARE REQUIREMENTS:

- MSP-EXP430G2 LaunchPad (MSP430G2553)
- Relay Module (5V)
- DC Motor (5V/12V)
- USB Cable
- Computer / Laptop

SOFTWARE REQUIREMENTS:

- Energia IDE / Code Composer Studio (CCS)
- MSP430 USB Drivers / MSPDebug / UniFlash Tool

THEORY:

1) Introduction:

A relay is an electromagnetic switch used to control high-power devices with a low-power control signal. The MSP430 microcontroller cannot directly drive the relay coil, so a transistor is used as an interface. When the transistor is driven into saturation by the microcontroller output, it energizes the relay coil, which closes the contacts and powers the motor.

2) Relay Working Principle:

The relay coil is energized when a small current flows through it, creating a magnetic field that moves an armature to connect the Normally Open (NO) and Common (COM) contacts. When de-energized, the contacts return to their default position.

3) Transistor Driver Circuit:

The GPIO pin of the MSP430 provides a control signal to the transistor base through a resistor. When the GPIO output is HIGH, the transistor saturates, completing the relay coil circuit. When LOW, the transistor switches off, de-energizing the coil.

4) Flyback Diode Protection:

The relay coil is an inductive load that generates a high-voltage back EMF when turned off. A diode connected across the coil (cathode to +V, anode to transistor collector) provides a path for this current, protecting the transistor and MSP430 from voltage spikes.

5) DC Motor Operation:

The DC motor is connected to the relay contacts. When the relay energizes, the NO contact closes and power flows to the motor, turning it ON. When the relay de-energizes, the contact opens and motor stops.

6) GPIO Configuration:

The MSP430 uses its P1.0 pin as output for relay control. The PxDIR register sets the direction, and PxOUT controls the logic level. Example:

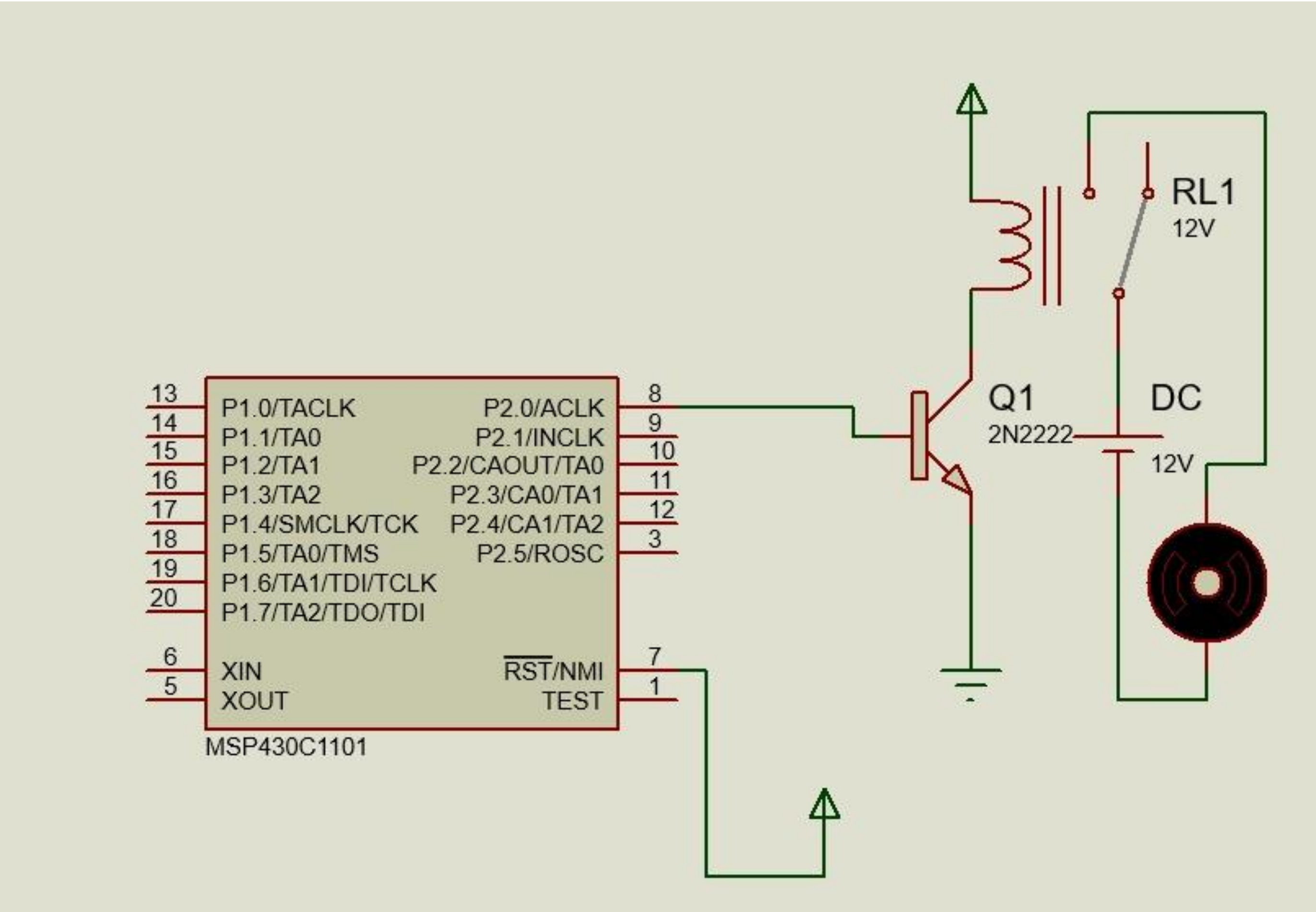
```
P1DIR |= BIT0;   // Configure P1.0 as output
```

```
P1OUT |= BIT0;   // Logic HIGH – Turn ON relay
P1OUT &= ~BIT0;  // Logic LOW – Turn OFF relay
```

7) Applications:

- Home automation
- Industrial control systems
- Motor ON/OFF control
- Safety interlocks
- Load switching

CIRCUIT DIAGRAM:



PROCEDURE:

- Assemble the transistor driver and relay circuit on the breadboard.
- Connect the relay contacts to switch the DC motor power.
- Connect the relay control pin to P1.0 of MSP430 through a 1kΩ resistor.
- Provide external 5V supply for the relay coil and motor.
- Open Code Composer Studio and create a new MSP430G2553 project.
- Write and build the code to toggle relay ON/OFF.
- Flash the code into MSP430.
- Observe relay click and motor ON/OFF operation.

SAMPLE CODE :

Attach separate sheet for code

CONCLUSION: