Introduction to Relays

- A relay is a device that responds to a small current or voltage change by activating a switch or other devices.
- Used to remotely switch signals or power.
- Input control usually electrically isolated from output.
- Input signal determines whether switch is open or closed.
Various Relay Configurations

Types of Relays

- Classic general-purpose relays have EM coils and can switch power.
- Solid-state relays (SSR) have input-triggered semiconductor switches.
- Reed relay has an EM coil and can switch low level DC signals.
- The bilateral switch uses CMOS, FET, or biFET transistors (technically not a relay but behaves similarly).
Types of Relays

Drawing of an EM Relay

Double Pole Double Throw (DPDT)

- Contact which is normally closed
- Contact which is normally open
- Leaf Spring Pole
- Armature Fulcrum
- Armature
- Armature Return Spring
- Electromagnetic Coil
- Frame
- Contact gap
Electromagnetic Relay Basics

- Input circuit is an EM coil with an Iron Core.
- Output switch includes two sets of silver or silver-alloy contacts (poles).
- One set is fixed to the relay frame, and other is located at end of leaf spring poles connected to the armature.
- Contacts held in “normally closed” position by the armature return spring.
- When input circuit energizes EM coil, a “pull-in” force is applied to the armature and “normally closed” contacts break while “normally open” contacts are made.

Solid State Relays

- Developed to solve limited life expectancy and contact bounce problems since they have no moving parts.
- Also, faster, insensitive to vibrations, reduced EMI, quieter, and no contact arcing.
- Optocoupler provides isolation between the input circuit (pseudocoil) and the triac (pseudocontact).
- Signal from phototransistor triggers the output triac so that it switches the load current.
- Zero-voltage detector triggers triac only when AC voltage is zero, reducing surge currents when triac is switched.
- Once triggered, triac conducts until next zero crossing.
Solid State Relays

Reed Relays

Single Pole Single Throw (SPST) Reed Relay
Solenoids

Pulse-Width Modulated DC Motors

- DC motor also has frame that remains motionless and an armature that moves in this case in a circular manner.
- When current flows through EM coil, magnetic force created that causes rotation of the shaft.
- Brushes positioned between frame and armature used to alternate the current direction through the coil so that a DC current generates a continuous rotation of the shaft.
- When current removed, shaft is free to rotate.
- Pulse-width modulated DC motor activated with fixed magnitude current but duty cycle varied to control speed.
Interfacing EM Relays, Solenoids, and DC Motors

- Interface circuit must provide sufficient current and voltage to activate the device.
- In off state, input current should be zero.
- Due to inductive nature of the coil, huge back electromotive force (EMF) when coil current is turned off.
- Due to high speed transistor switch, there is a large \( di/dt \) when the coil is deactivated (activation also but smaller).
- Voltages can range from 50 to 200V.
- To protect the driver electronics, a snubber diode is added to suppress the back EMF.

Relay and Motor Interfaces

[Diagram of a relay and motor interface circuit with a computer output port, transistor, and diode.]
Isolated Interfaces

H-Bridge

Motor coil

Current flow
Isolated H-Bridge with Direction Control

Stepper Motors

- Very popular due to inherent digital interface.
- Easy to control both position and velocity in an open-loop fashion.
- Though more expensive than ordinary DC motors, system cost is reduced as they require no feedback sensors.
- Can also be used as shaft encoders to measure both position and speed.
Stepper Motors

Simple Stepper Motor Interface

![Simple Stepper Motor Interface Diagram](image-url)
### Stepper Motor Sequence

<table>
<thead>
<tr>
<th>PortB</th>
<th>A</th>
<th>A’</th>
<th>B</th>
<th>B’</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Activate</td>
<td>deactivate</td>
<td>activate</td>
<td>deactivate</td>
</tr>
<tr>
<td>9</td>
<td>Activate</td>
<td>deactivate</td>
<td>deactivate</td>
<td>activate</td>
</tr>
<tr>
<td>5</td>
<td>Deactivate</td>
<td>activate</td>
<td>deactivate</td>
<td>activate</td>
</tr>
<tr>
<td>6</td>
<td>Deactivate</td>
<td>activate</td>
<td>activate</td>
<td>deactivate</td>
</tr>
</tbody>
</table>

### Stepper Motor Basic Operation

- **Stator**: The stationary part of the stepper motor, containing coils that produce magnetic fields.
- **Poles**: The magnetic poles of the stator that interact with the rotor's teeth.
- **Electromagnets**: The coils within the stator that generate magnetic fields.
- **Rotor**: The rotating part of the stepper motor, containing the teeth that interact with the stator's fields.
- **Shaft**: The central axis around which the rotor rotates.
- **Tooth pitch**: The distance between the teeth on the rotor.
- **North** and **South**: The north and south poles of the magnetic fields generated by the stator.
Stepper Motor Basic Operation (cont)
Stepper Motor Basic Operation (cont)

Bipolar Stepper Motor Interface
A *slip* is when computer issues a sequence change, but the motor does not move.

- Occurs if load on shaft exceeds available torque of motor.
- Can also occur if computer changes output too fast.
- If initial shaft angle known and motor never slips, computer can control shaft angle and speed without position sensor.
Data Structures to Control Stepper Motor

```c
const struct State{
    unsigned char Out; // Output
    const struct State *Next[2]; // CW/CCW
};
typedef struct State StateType;
typedef StateType *StatePtr;
#define clockwise 0     // Next index
#define counterclockwise 1 // Next index
StateType fsm[4]= {
    {10, &fsm[1], &fsm[3] },,
    { 9, &fsm[2], &fsm[0] },
    { 5, &fsm[3], &fsm[1] },
    { 6, &fsm[0], &fsm[2] }};
unsigned char Pos; // between 0 and 199
StatePtr Pt;      // Current State
```

Ritual to Control Stepper Motor

```c
void Init(void){
    Pos = 0;
    Pt = &fsm[0];
    DDRB = 0xFF;
}
```
**Helper Functions to Control Stepper Motor**

```c
void CW(void){
    Pt = Pt->Next[clockwise];  // circular
    PORTB = Pt->Out;          // step motor
    if(Pos==199){             // shaft angle
        Pos = 0;              // reset
    }else{
        Pos++;}
}  // CW

void CCW(void){
    Pt = Pt->Next[counterclockwise];
    PORTB = Pt->Out; // step motor
    if(Pos==0){ // shaft angle
        Pos = 199; // reset
    }else{
        Pos--;}
}  // CCW
```

**High-Level Control of Stepper Motor**

```c
void Seek(unsigned char desired){
    short CWsteps;
    if((CWsteps=desired-Pos)<0){
        CWsteps+=200;
    } // CW steps is 0 to 199
    if(CWsteps>100){
        while(desired!=Pos){
            CCW();
        }
    }else{
        while(desired!=Pos){
            CW();
        }
    }
}
```
Stepper Motor as Shaft Position Sensor

Timing of Stepper Motor as Shaft Position Sensor