UNIT IV

INTERFACING WITH PIC MICROCONTROLLER

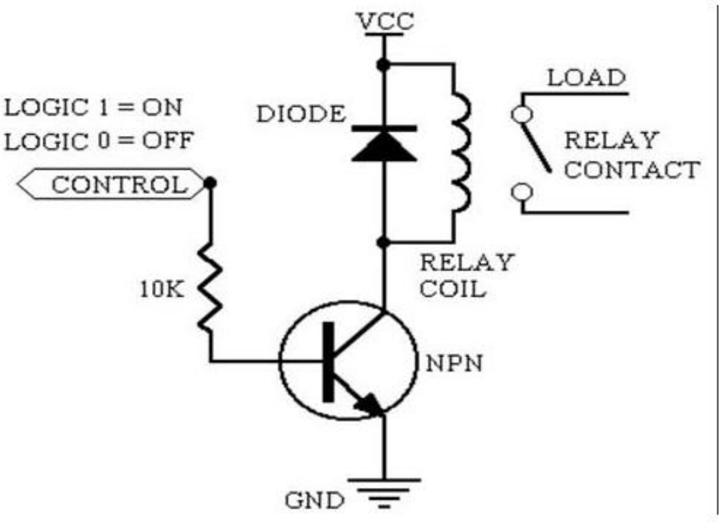
RELAY AND OPTOCOUPLER INTERFACING

RELAYS:

Relays are devices which allow low power circuits to switch a relatively high Current/Voltage ON/OFF. A relay circuit is typically a smaller switch or device which drives (opens/closes) an electric switch that is capable of carrying much larger current amounts.

Interfacing Relays

Fig. 1 shows how to interface the Relay to microcontroller. There are 2 input channels. Each input is connected to the triggering coil of the respective relay. There are 2 output channels that each correspond to an input. When the input is energized, the relay turns on and the '+' output is connected to +12v. When the relay is off, the '+' output is connected to Ground. The '-' output is permanently wired to Ground.



Interfacing Relay with PIC16F877A

We now want to control the relay operations by using PIC16f/18FAdvanced Development Board.Here we are using two Relays. The relay consists of a coil and a switch. When the coil is energized, the switch closes, connecting the two contacts together. ULN2803 is used as a driver for port I/O lines, drivers output connected to relay modules. Connector provided for external power supply if needed.
Relay Module: Port B pins (Relay1 – PORTB.0) and Relay2-PORTB.1) for relay module, make port pins to high, relay will activated.

```
#include
//Define PIC Registers
#include
//Define I/O Function
#define Relay1 RB0
//RB1 interfaced to Relay1
```

```
#define Relay2 RB1
//RB2 interfaced to Relay2 __CONFIG(0x3f72);
//Select HS oscillator, Enable (PWRTE,BOREN),
//Disable (CPD,CP,WDTEN,In-circuit Debugger)
#define FOSC 10000 //10Mhz==>10000Khz
#define BAUD_RATE 9.6 //9600 Baudrate
#define BAUD_VAL    char   FOSC/   16 * BAUD_RATE     – 1
//Calculation For 9600 Baudrate @10Mhz void Serial_init(void);
//Serial Communication Initialization void DelayMs(unsigned int);
void main

unsigned char ReceivChar
TRISB = 0x00
PORTB = 0x04
//PORTB configured as O/P Serial_init();
printf "\033[2J"
DelayMs 20
printf "\n\r\t3: Relay1 ON \n\r\t4: Relay1 OFF"
printf "\n\r\t5: Relay2 ON \n\r\t6: Relay2 OFF\n\r\n"
while 1

while RCIF==0
//Wait until the reception is over RCIF=0;
//Clear the flag for next reception ReceivChar=RCREG;
//Store the received char to a variable printf(" %c",ReceivChar);
//Dislpay the character received switch(ReceivChar)

case '3': //If '3' is received Relay1 turned ON Relay1=1;
break
case '4': //If '4' is received Relay1 turned OFF Relay1=0;
break  case '5': //If '5' is received Relay2 turned ON Relay2=1;
break  case '6': //If '6' is received Relay1 turned OFF Relay2=0;
break



void Serial_init

TRISC=0xc0
//RC7,RC6 set to usart mode(INPUT) TXSTA=0x24;
//Enable(Serial TXn,//Asynchronous, High Speed) SPBRG=BAUD_VAL;
//9600 baud at 10Mhz RCSTA=0x90;
//Usart Enable, Continuous receive enable TXIF=1;
//Start Transmit

void putch unsigned char data
```

```
while  TXIF==0
TXREG=data
//Transmit the data serially

void DelayMs  unsigned int Ms

int delay_cnst  while  Ms>0

Ms--
for  delay_cnst = 0 delay_cnst <220
delay_cnst++
```

OPTOCOUPLERS:

Optocouplers were discovered right after photo-transistors (like any other transistor, except it is stimulated by light), by combining a LED and photo-transistor in the same case. The purpose of an optocoupler is to separate two parts of a circuit.

This is done for a number of reasons:

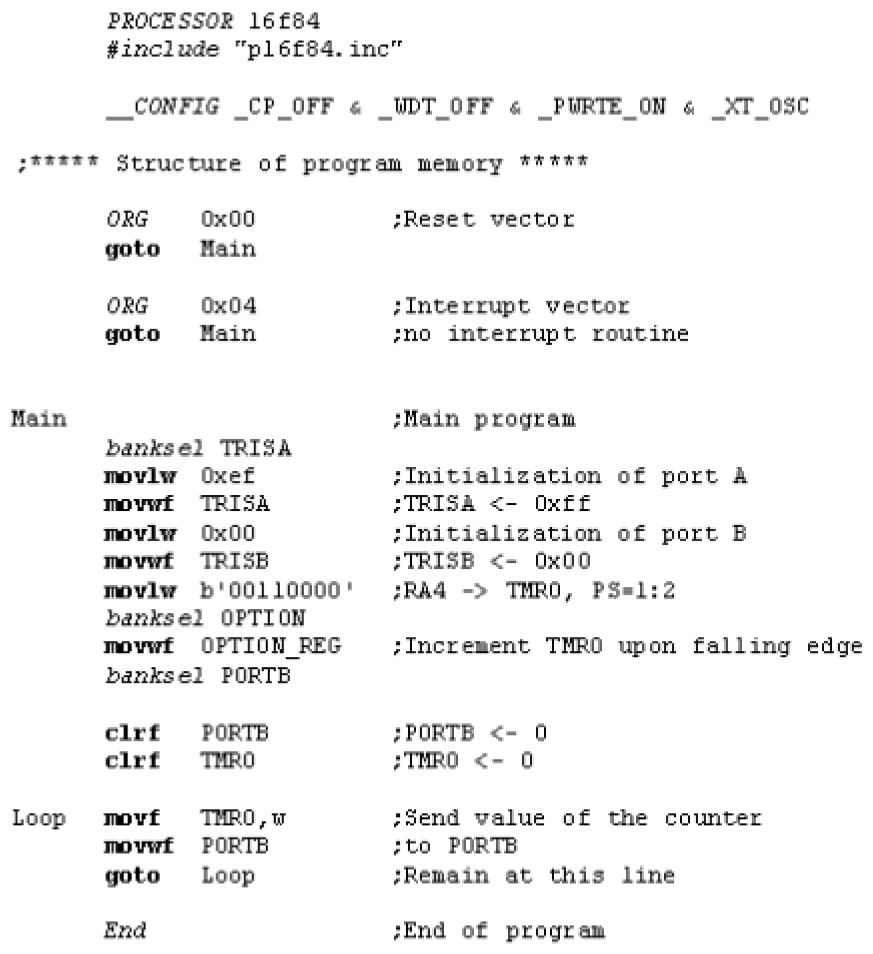
- Interference. Typical examples are industrial units with lots of interferences which affect signals in the wires. If these interferences affected the function of control section, errors would occur and the unit would stop working.
- Simultaneous separation and intensification of a signal. Typical examples are relays which require higher current than microcontroller pin can provide. Usually, optocoupler is used for separating microcontroller supply and relay supply.
- In case of a breakdown, optocoupled part of device stays safe in its casing, reducing the repair costs.

Optocouplers can be used as either input or output devices. They can have additional functions such as intensification of a signal or Schmitt triggering (the output of a Schmitt trigger is either 0 or 1 - it changes slow rising and falling waveforms into definite low or high values). Optocouplers come as a single unit or in groups of two or more in one casing. Each optocoupler needs two supplies in order to function. They can be used with one supply, but the voltage isolation feature, which is their primary purpose, is lost.

Optocoupler on an input line

The way it works is simple: when a signal arrives, the LED within the optocoupler is turned on, and it illuminates the base of a photo-transistor within the same case. When the transistor is activated, the voltage between collector and emitter falls to 0.7V or less and the microcontroller sees this as a logic zero on its RA4 pin.

The example below is a simplified model of a counter, element commonly utilized in industry (it is used for counting products on a production line, determining motor speed, counting the number of revolutions of an axis, etc). We will have sensor set off the LED every time axis makes a full revolution. LED in turn will 'send' a signal by means of photo-transistor to a microcontroller input RA4 (TOCKI). As prescaler is set to 1:2 in this example, every second signal will increment TMR0. Current status of the counter is displayed on PORTB LEDs.



```
        PROCESSOR 16f84
        #include "p16f84.inc"

        __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC

 ;***** Structure of program memory *****

        ORG     0x00            ;Reset vector
        goto    Main

        ORG     0x04            ;Interrupt vector
        goto    Main            ;no interrupt routine


Main                            ;Main program
        banksel TRISA
        movlw   0xef            ;Initialization of port A
        movwf   TRISA           ;TRISA <- 0xff
        movlw   0x00            ;Initialization of port B
        movwf   TRISB           ;TRISB <- 0x00
        movlw   b'00110000'     ;RA4 -> TMR0, PS=1:2
        banksel OPTION
        movwf   OPTION_REG      ;Increment TMR0 upon falling edge
        banksel PORTB

        clrf    PORTB           ;PORTB <- 0
        clrf    TMR0            ;TMR0 <- 0

Loop    movf    TMR0,w          ;Send value of the counter
        movwf   PORTB           ;to PORTB
        goto    Loop            ;Remain at this line

        End                     ;End of program
```