November 2024 Embedded System Design

Q_2

(a) Similarities

* Both are PIC microcontrollers made by Microchip
* Both use Harvard architecture, which means they have separate memory for program and data
* Both can be program with assembly lanage or C
* Both support digital I/O, timers, and interrupts

Difference

* 16F84A is simpler. It has less memory 13 I/O pins, and no ADC
* 16F877 is advanced. It has more memory 33 I/O pins, built in ADC, PWM and USART

(b)

Faster development
* Changes and test can be test immediately No need to remove the the chip
* Reduces risk of damaging pins or boards Can debya in real environment
* You can see how the microcontroller behaves with real sensors or motors

(c)

* Reprogrammable - You can erase and write it many times
* Non-volatile - It keeps data when power is off
* Faster and cheaper

(d)

* The CPU processes 8 bits at a time
* The registers and data bus are 8 bit wide

(e) "8k" means $8 \times 1024 = 8192$ words

In pic microcontrollers, each word is 14 bits

total bits $= 8192 \times 14$
$= 114,688$ bits

(f) The RC circuit connected to RESET is called a Power On Reset circuit

* when a power is turn on, the capacitor charges slowly. keeping the REGET pin LOW for a short time
* This gives time for the power supply to stabilize
* The RESET goes HIGH, and the microcontroller starts running normally

(a)

```
movlw   b'11110000'
```

Load binary to working register

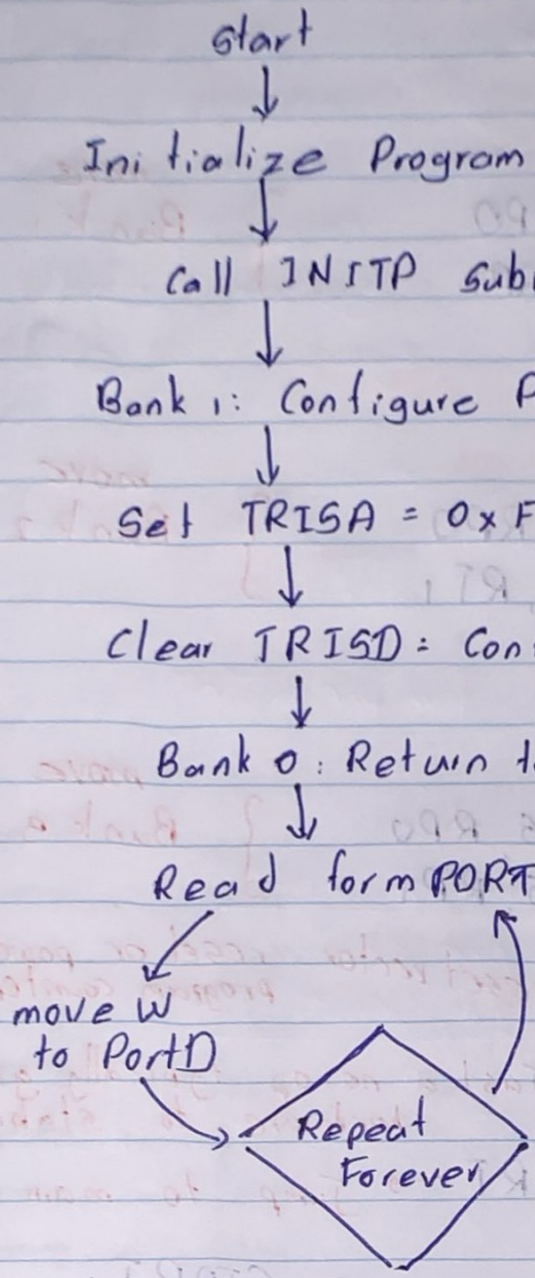
```
movwf   TRISB
```

Move the contents of WREG into the TRISB register

Q.

(a) Push Button LED Array Controller

(b)



(c)
```
Bank0  macro
    bcf STATUS,RP0
    bcf STATUS,RP1
endm

Bank1  macro
    bsf STATUS,RP0
    bcf STATUS,RP1
endm

Bank2  macro
    bcf STATUS,RP0
    bsf STATUS,RP1
endm

Bank3  macro
    bsf STATUS,RP0
    bsf STATUS,RP1
endm

ORG 0
NOP
GOTO START

ORG 0x20

START:
    CALL INITP

REPEAT:
    MOVF PORTA,W
    MOVWF PORTD
    GOTO REPEAT
```

move Bank0

move Bank1

move Bank2

move Bank3

reset vector

reset or power up, the program counter goes here

Just a no-op typically give time for hardware to stability

Jump to main program

main program in here

call INITP to configure PORTA and PORTD

Read button input form port A and put in to the working Register

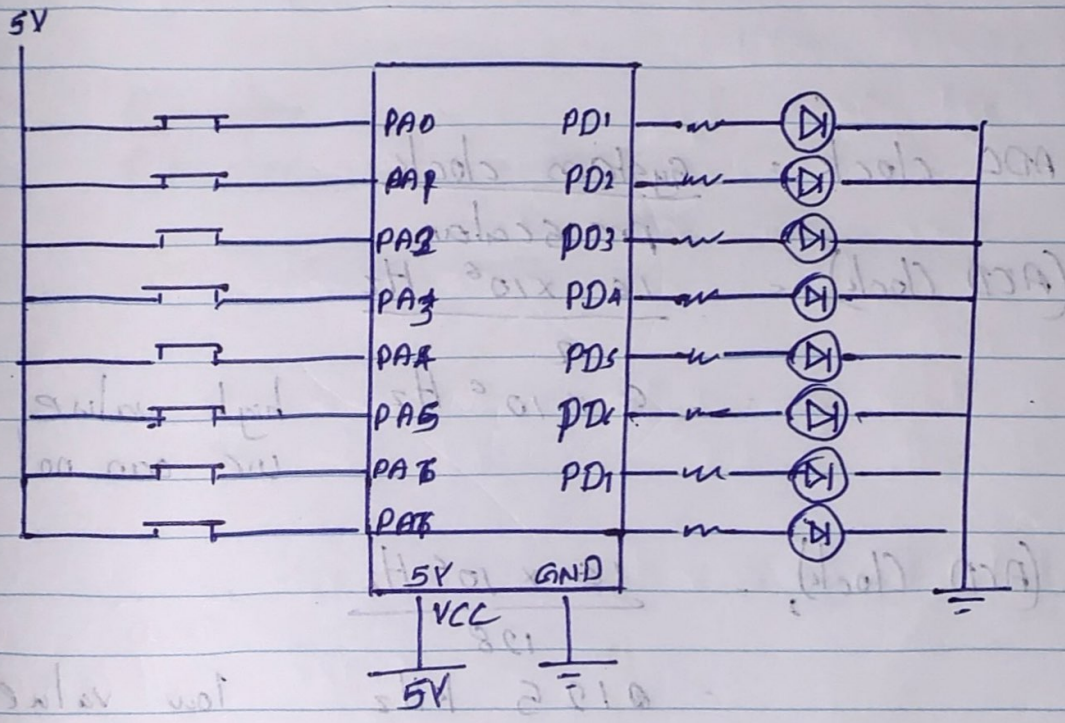
give value from working register and put into portD

Loop

~~INITP~~

```
INITP:
    Bank 1          → switch to Bank 1 to acces TRIS and ADCON1
    MOVMF 0x06
    MOVWF ADCON1    → Configure analog pins as digital (off A/D on PORTA)
    MOVLW 0xFF
    MOVWF TRISA     → Set all Port D pins as inputs
    CLRF TRISD      → set all Port D pins as output
    Bank 0          → Return to Bank 0 for normal operation
    RETURN
END
```

(d)

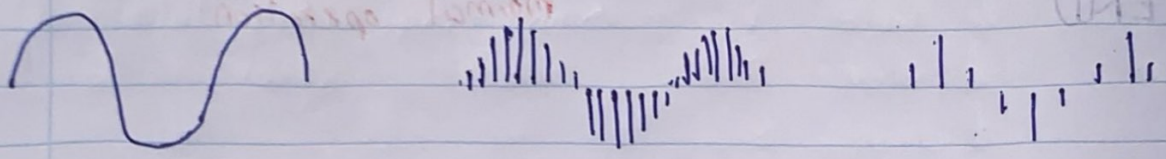


Q_3

(a)

(i) Void initHardware (void).
{ DDRA |= 0b 11111111;
DDRC |= 0b 11111111;
}

(ii) We use the system clock convert the analog signal to Digital



low pre scaler high pre scaler

ADC clock = System clock / pre scalar

(ADC Clock)$_1$ = $\frac{16 \times 10^6 \text{ Hz}}{2}$

= 8×10^6 Hz high value we can not identify

(ADC Clock)$_2$ = $\frac{16 \times 10^6 \text{ Hz}}{128}$

= 0125 kHz low value we can identify

(iii)
```
void initADC(void)
{
    ADCSRA |= (1<<ADEN);
    ADCSRA |= (1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
}
```

(iv) `ADCSRA |= (1<<ADSC);`

(v) `while (ADCSRA & (1<<ADSC));`

(vi)
```
PORTA |= ones | (tens << 4);
PORTC |= hundreds | (thousands << 4);
```

(vii) ~~Register~~
Resister : 1×10^3

b) ~~sto~~

(i) stop bit
* no of parity bit and it should be same even or odd
* baud rate of the devices
* Number of data carry bit (Data frame format)
* Voltage levels

(ii) Total bits per frame = 1 start bit
8 data bit
0 parity bit
1 stop bit
10 bit

At 9600 bps

time per bit = $\frac{1}{9600}$

Time per byte = 10 bits $\times$ $\frac{1}{9600}$

total transmission time = $4 \times 10 \times \frac{1}{9600}$
= 4.168 milliseconds

(c) whatch dog timer

Crucial points

01 System recovery mechanism that automatically reset the microcontroller if software becomes unresponsive

02 Acts as a failsafe against software bugs, unexpected conditions, or hardware issues that cause program execution to freeze

03. Requires periodic resets by properly functioning software to prevent timeout

04. Can be configured with different timeout periods depending on application requirements

Q.

(a) UART
Asynchronous serial communication (no. clock signal)
2 (Tx, Rx)
Full duplex
Longer range

SPI
Synchronous serial communication (with clock signal)
4 MOSI MISO SCLK SS
Full duplex
Short range

I2C
Synchronous Serial communicat. (with clock signal)
2 (SDA, SCL)
Half duplex
Short to medium range

(b) (i) Communication mechanism between ATmega 328P and LCD
Parallel communication
typically 4-bit or 8 bit interface

```
#include <avr/io.h>
#include <util/delay.h>
#include "LCD_Commands.h"

int main(void){

    LCD_Init();

    LCD_setCursor(0,0);
    LCD_WriteString("Embedded Systems");

    for(int i=0; i<16; i++){
        _delay_ms(300);
        LCD_ShiftLeft();
    }

    for(int i=0; i<16; i++){
        _delay_ms(300);
        LCD_ShiftRight();
    }

    LCD_Clear();
    LCD_SetCursor(1,0);
    LCD_WriteString("DEIE");

    while(1){
    }
```

(c) (i) Three SPI settings to match between transmitter and receiver

01. Clock polarity (CPOL)
02. Clock phase (CPHA)
03. Data order (MSB / LSB first).

(ii) Slave Select Pin

* The SS pin is used by the master to select a specific slave device for communication
* When SS is LOW (0) the slave is selected and responds to SPI signals
* If the master selects more than one slave at a time, multiple devices may try to send data simultaneously, causing bus contention and data corruption

(d) Burn in testing involves running hardware components under stressfull conditions for a prolonged period to identify early failures

01. Weed out defective units
02. Ensure long-term reliability by stabilizo components
03. Avoid field failures after deployment

SOCK_STEAM
Provides sequenced, reliable, two way, connection based byte stereams. It uses TCP

SOCK_DGRAM