10 Novembre 2023

**4 Lezione Embedded**

Sistema microcontrollore -> economico -> tanti pulsanti devo mettere tanti pulsanti nel microcontrollore

Ingressi -> punti dell’integrato (piedini) -> ogni posto più c’è più costa -> diversi metodi -> con 7 fil, 12 segmenti, rileva 20 pulsanti diversi

Pulsante -> sistema a matrice -> con 4 righe -> dipende pulsante da rilevare, incroci che voglio generare -> metto pulsante che genera un cortocircuito tra riga nessima e bottone mesimo

Pulsante non collegato -> determinata tensione -> PULL DOWN -> resistenza collegata a 5V -> tensione stabile e sicura -> sistema che commuta in base allo stato del bottone H / L (alto o basso)

Decido io valori -> input e output

Qualsiasi pressione -> non comporta nessun cambiamento -> pulsante collegato a 5V -> non c’è corrente che passa -> non rilevo nessuna modifica parte D -> stato di riposo tastierino

Con tutto 1 -> non sta accendo nulla

Problema -> 3 punti

Portando a zero prima dei punti -> garantendo altre linee a 1 -> cortocircuitando diversi valori. Il dato che porto a 0 lo collego a massa e quindi a 0 -> controllare diversi dati una alla volta e posso decidere il suo valore di quello che sto decidendo di leggere

Quando premo pulsante diventa -> 0 -> controllo se bit è andato a zero -> loop annidato che controlla se bi è andato a zero e tengo traccia -> indice che mi serve per tenere in memoria i dati dove mi trovo

Un ciclo con 12 controlli -> tutti bit 3 per RD0 con i relativi RB0, RB1 e RB2 cosi per RD1 e RD2 -> controllo per tutte le colonne

Codice corrispondente alla funzione che capisce i controlli delle intersezioni dove è a 0

For (RB0 RB1 RB2)

{

For(RD)

{

If(RD == 0)

{

Nriga + ncol \* 4; // combinazione lineare spazio di due

coordinate su due dimensioni,

numero univoco ciascun pulsante

}

}

}

1 for -> abbassare 1 bit alla volta -> decido ultimi 3 bit in base a cosa leggo

2 for -> leggo la colonna del bit 3 finali corrispondenti secondo quindi leggo il secondo

PORTB & = ~ (1 << col);

<< -> shift logico -> byte (8 bit) -> colonne tutte a zero tranne quello che mi interessa per creare il 1

Shift a sx -> spostarlo a sx -> in questo modo va avanti -> numero dopo -> va via -> numero dopo diventa a 0 -> dipende macchina di solito 0

Numero di volte che mi sposto shift -> sposto numero di quanto mi serve

~ -> 0 diventa 1

1 diventa 0

& = te stesso e quello che viene dopo e lo aggiungi

PORTB = PORTB & ~ (1 << col);

io posso decidere -> cosa forzare a 1, abbasso una colonna alla volta -> controllo bit -> dopo controllo valore della resistenza del valore -> ~ inverto valore

PULL UP -> premo per accendere, READ -> lettura per accendere

Imposto le colonne e controllo le righe ed è una matrice 3 \* 4

Ultimo for -> controllo ciascun bit delle righe -> 1 shiftato a sx e in questo modo trovo RD0, RD1 oppure gli altri -> infine calcoli pulsante indicato

Posso generare -> diversi interrupt in un programma -> per ogni porta and -> un interrupt -> da eeprom, comparatore e n periferiche -> dal timer (TMR0) -> rilevare quell’interrupt filo T0IE -> attivare GIE -> attivare verso interrupt e T0IE -> interrupt per TMR0

Registro INTCON -> 8 bit ognuno significato -> più significato GIE, sensibile interrupt

T0IE -> sensibile interrupt per TMR0

Interrupt -> scatena al punto che mi interessa dopo il massimo -> non sappiamo che la chiamato

Programmatore -> scrivere codice perché funzioni -> interrupt necessari per fare l’opzioni interessata, passa tempo definito nel timer

Seriale -> lavora con una certa velocità

Errore soft -> interrupt TMR0 -> va bene in base requisiti, altrimenti devi trovare soluzione che diventi numero intero -> non trovo soluzioni chiami progettista che cambi dati -> per trovare opzione -> trovi devi fare bilanciamento

Display LCD -> display intelligenti (memoria interna posso stampare caratteri e simboli tabella ascii standard) -> certo numero di righe e certo numero di colonne

Fattore intelligente -> macchina HD44780 -> tutti combattibili -> quel chip -> stessa procedura

Chip diverso -> procedura diversa -> pettine diverso -> collegamento diverso con collegamento ai vari fili diversi

Microcontrollore collegato a pc -> comunicazione parallela sincrona -> numero di fili collegati -> dati (fili un bit del mio dato)

Diversi fili -> fili di enable -> alzo o abbasso con i vari dati -> dati inserito o no -> se si memorizza dati -> scrivo su display -> configurazione in una maniera

Filo RS -> register select -> due valori 0 o 1

Valore 1 -> stampa su display memorizza su memoria del display

Valore 0 -> no memoria visualizzazione ma altra -> registri di memoria dove posiziono il dato -> con verso di scrittura sx o dx -> dati decido l’ordine -> varie configurazioni quando RS = 0, se RS = 1 no

Bit passano in modo parallelo -> tutti assieme

Sincroni -> decise da un clock che tramite enable te lo fa passare

Asincrona -> velocità prestabilita, non esiste filo

Dati dove vogliono che vengono caricati -> piedini dove vogliono che vengono caricati

Piedini led cosa fanno

1 -> VSS e GND

2 -> VCC

3 -> contrasto immagine

4 -> scrivere

5 -> leggere e scrivere RW

6 -> E enable

7 -> DO

8 -> DI

15 -> A anodo led

16 -> K catodo led

Display -> con n caselle (visualizza carattere in base a codice ascii) -> memoria tiene in memoria dato inserito -> memoria a indirizzi -> tante caselle con n posti in base alla memoria

Casella del display -> come struttura dell’array

Chip -> dalla 16 in poi scorrono -> esce fuori spazio visivo display quindi non vedi come hai scritto -> scrivi posto sbagliato non vedi, posto giusto si

Struttura codice display

Prima linea -> da 1 a 39

Seconda linea -> terza linea da 80 a 119

Terza linea -> seconda linea da 40 a 79

Quarta linea -> quarta linea da 120 a 179

Codice più facile da scrivere

Tabella dove vedo -> configurazione del dispositivo che posso vedere dei punti dove interessa vedere le configurazioni ed in questo modo possiamo capire come scrivere nel display

Tabella cosa scrivere -> vedo cosa posso scrivere tabella in modo diagonale, capire che comando sto utilizzando -> far diventare facile qualcosa di complesso solo che non è troppo semplice

Tabella -> vedo in che riga sono e dove mi posso collegare -> quale riga mi rovo -> bit assumono valori e significati diversi

D1 -> incrementa valore da una parte o l’altra in base a quello che mi serve per spostarmi da dx o sx -> direzione scrivere, bit più significativo

D0 -> bit meno significativo S

Da meno significato a più significativo -> mi sposto di 1 alla volta o a dx o a sx

Display -> gestione parte estetica o parte normale in base a come voglio scrivere



Funzionamento display -> configurato in base a come utilizzarlo inizializzarlo -> calcoli tabelle che fai vedere



0x01 -> cancelli il display

L1 e l2 -> scrivo prima linea e ultima linea

Scrivi funzione che succede per delle sequenze -> per pilotaggio scheda -> comandare i dati -> comando i dati (invia dati e stampa carattere) -> più facile da utilizzare, per i comandi manda comandi o dati -> valore RS 0 o 1

2 Funzioni -> comando o dati, 1 solo funzione -> perché cosa passano i dati in cosa decidono