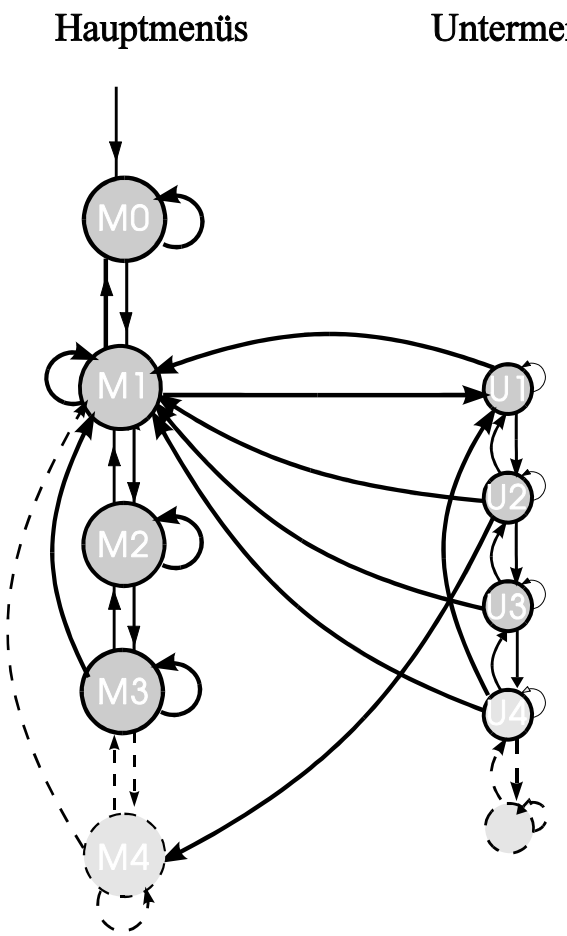


MODUL 5

Anwendung von "STATE MACHINES" in der µC - Programmierung

V1.1 Modul mit anderer Tastaturbeschaltung J. Humer 1995

Anwendung von "STATE MACHINES" in der µC - Programmierung

1 STATE MACHINES

1.1 Allgemeines :

STATE MACHINES (SM) sind zustandsgesteuerte Maschinen. SM werden in der Digitaltechnik zum Entwurf sequentieller Schaltungen verwendet. Die Bezeichnung STATE MASCHINE kommt aus der Automatentheorie und es wird prinzipiell zwischen zwei SM Modellen unterschieden. Man entwirft entweder nach dem

- MOORE Modell
  oder dem
- MEALY Modell

Kurzbeschreibung beider Modelle:

MEALY MODELL :

Der Zustand am Ausgang der Schaltung hängt vom **momentanen Zustand und vom Eingangszustand** ab.

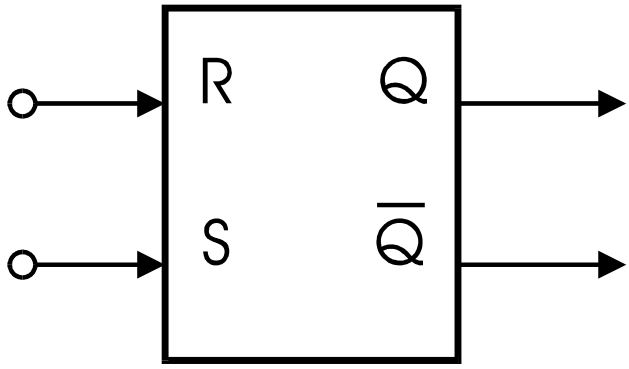
MOORE MODELL :

Der Ausgangszustand ist nur vom (Ereignis am) Eingang abhängig !
Änderungen des Ausgangs sind nur bei Taktflanke am Eingang möglich.

Beispiele für die Verwendung von SM :

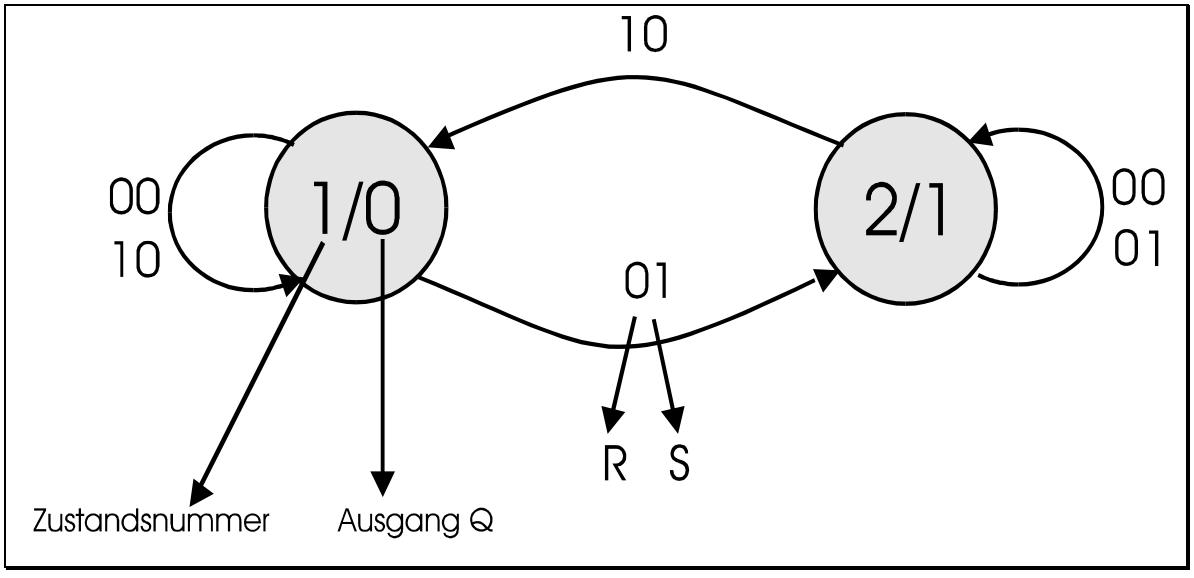
- Zählerschaltungen die in eigenem Kode zählen
- Sequenzerkennung im seriellen Bitstrom
- Serielles Interface
- Adressdekodierung
- Menüführung .....

2 Erklärung einer SM anhand eines RS-FF



SR	Q+
00	Q
01	1
10	0
11	X

2.1 ZUSTANDSDIAGRAMM (nach Moore) :



☞ Tragen Sie die verbotenen Zustände 11 Ausgang XX in das Zustandsdiagramm ein

2.2 Programmierung der Flip-Flop Steuerung mit State Machine

2.2.1 AUFGABENSTELLUNG:

Es ist nun mittels 2er Tasten Ihres Mikrokontrollerboards ein Modell eines RS Flip-Flops
zu programmieren und zwar derart, daß die beiden Tasten den Eingängen R beziehungsweise S entsprechen.
Die jeweiligen momentanen Zustände 1 und 2 sowie der Ausgangszustand (0 oder 1) ist mit den ebenfalls am Mikrokontrollerboard befindlichen LEDs anzuzeigen.
Diese leichte Aufgabe mit nur 2 Zuständen könnte man auch relativ einfach ohne State-Machine herkömmlich etwa folgendermaßen programmieren :

```
IF State = 1
DO CASE
      CASE  INPUT = SET
            State=2
      CASE INPUT = RESET
            State=1
ENDCASE
IF State = 2
DO CASE
     CASE  .....usw.
```

Um derart unübersichtliche Konstruktionen zu vermeiden, bietet sich die Programmierung mit Hilfe von State Machines an.

1. SCHRITT: Kodierung der Tastatur

Eingang INPUT	Aufgabe	Ausgang Out
SET	Zifferntasten	***KODE 0***
RESET	Buchstaben	***KODE 1***

unbenutzt	**----------**	***KODE 2***

2. SCHRITT: Definition der Ausgangsfunktionen (out)

Aktion Nr.:		Aufgabe :	Funktion Name:
0		Led 0 ansteuern	LED0()
1		Led 1 ansteuern	LED1()
2	z.B.	Led 8 ansteuern	LED8()

3. SCHRITT: Aufstellen der Zustands und Funktionstabellen

Hier werden die eingangs-, und zustandsabhängigen Folgezustände festgelegt

Eingänge		Momentaner Zustand	nächster Zustand
		0	1
SET	0	1	1
RESET	1	0	0
nicht benutzt	2	0	1

Hier werden die eingangs-, und zustandsabhängigen Folgeaktionen festgelegt

Eingänge		Momentaner Zustand	Aktion (out)
		0	1
SET	0	1	1
RESET	1	0	0
nicht benutzt	2	2	2

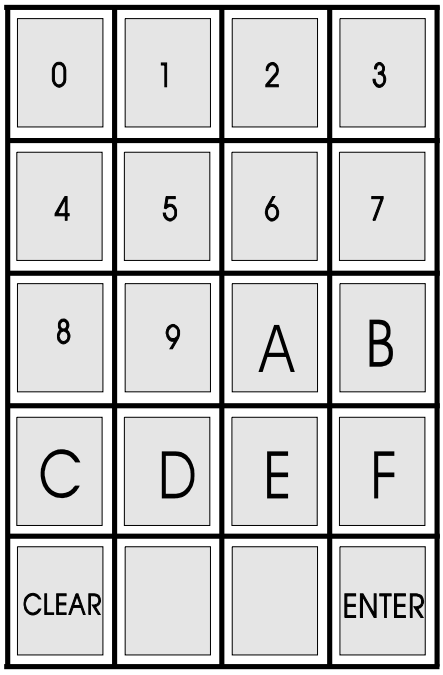
3 State Machine Steuerung für Tastatureingaben

3.1 Definition der STATE MACHINE :

AUFGABE: Folgende Sequenz soll erkannt und verarbeitet werden :

- *Buchstabe - ZIFFER - Buchstabe - ZIFFER* - ENTER
  z.B.: **B 3 D 5 ENTER**

3.2 Kodierung der Tastatur :



Eingang	Aufgabe	Ausgang
INPUT		Out
0...9	Zifferntasten	***KODE 0***
A-F	Buchstaben	***KODE 1***
Löschen	Eingabetaste	***KODE 2***
ENTER	Eingabetaste	***KODE 3***
unbenutzt	----------	***KODE 4***

3.3 Aufstellen der Zustands,- und Funktionstabellen:

Eingangsvariable **momentaner Zustand** **nächster Zustand**

INPUT present state next state

		0	1	2	3	4
0..9	0	0	2	2	4	4
A..F	1	1	1	3	3	4
CLEAR	2	0	0	1	2	3
ENTER	3	0	1	2	3	0
nicht benutzt	4					

Eingangsvariable **momentaner Zustand** **Aktion**

INPUT present state output

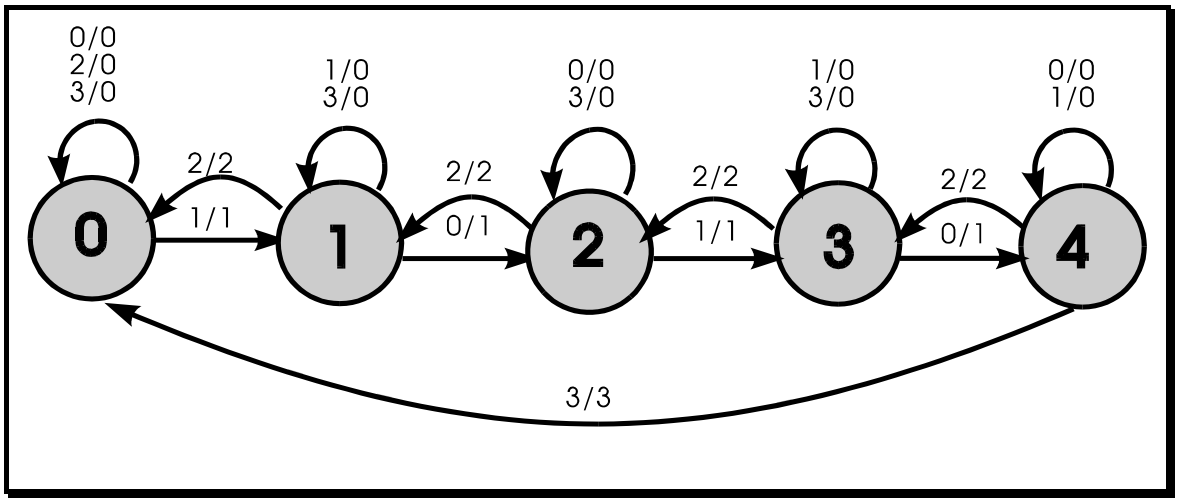
		0	1	2	3	4
0..9	0	0	1	0	1	0
A..F	1	1	0	1	0	0
CLEAR	2	0	2	2	2	2
ENTER	3	0	0	0	0	3
nicht benutzt	4					

☞ Ergänzen Sie die Kodierung für die nicht verwendeten Tasten

3.3 Definition der Ausgangsfunktionen OUT:

Aktion Nr.:	Aufgabe :	Funktion Name:
0	Keine Aktion	NO_OP()
1	Eingabe auf LCD schreiben	Write_LCD()
2	letzte Eingabe löschen	Clear_INPUT()
3	LCD Inhalt weitergeben,löschen	Save_LCD()
4	LCD Inhalt löschen	Clear_LCD()
5	xxxxx	xxx_xx()
6	.....	...._....()
usw.		

3.4 Graphische Problemlösung :



Übung 16: Tragen sie sowohl in den Tabellen als auch in der Grafik die fehlenden Zeilen für die unbenutzten Tasten ein.

4 Programmlisting Tastatureingabe:

```
/****   Steueranweisungen für den Compiler       ****/
#pragma noiv
#pragma mod517
#pragma code debug pl(61)

/****   Angabe der Include Dateien              ****/
#include <reg517.h>
#include <stdio.h>
#include <LCD.h>

bit gedrueckt,lcd_ser;
char x,y,text[25];
unsigned char taste,menu,input,state;



/*************************************/
/*         NO Operation  NO_OP()       */
/*************************************/
NO_OP()
{
   print_lcd(1,1,"Reihenfolge falsch");
   pos(2,1);printf("Taste %2bu",taste);
   warte(2000);
   blank_LCD();
   return;

}



/*************************************/
/*      Schreibt Eingabe  auf LCD      */
/*************************************/
Write_LCD()
{
   print_lcd(1,1,"Write_INPUT");
   pos(2,1);printf("Taste %2bu",taste);
   return;

}
```

```
/*************************************/
/*          Löscht Eingabe           */
/*************************************/
  Clear_INPUT()
  {
     print_lcd(1,1,"CLEAR_INPUT");
     return;
  }


 /*************************************/
 /*          Speichert Eingabe(n)     */
 /*************************************/
  Save_LCD()
  {
     print_lcd(1,1,"SAVE LCD");
     return;
  }



/*******************************************************/
/*        TASTENLAYOUT + CODIERUNG          */
/*******************************************************/

  code unsigned char decoder_tab[20]=
                                {   0,0,0,0,
                                    0,0,0,0,
                                    0,0,1,1,
                                    1,1,1,1,
                                    2,4,4,3   };



/*************************************************************/
/*    TABELLE  für Stateabh. Zifferneingabefunktionen        */
/*************************************************************/

 code (*fpnt[4][5])()=
{
{ NO_OP,    Write_LCD,  NO_OP,        Write_LCD,    NO_OP},
{ Write_LCD,NO_OP,      Write_LCD,    NO_OP,        NO_OP  },
{ NO_OP,    Clear_INPUT,Clear_INPUT,  Clear_INPUT,  Clear_INPUT },
{ NO_OP,    NO_OP,      NO_OP,        NO_OP,        Save_LCD    }
};
```

```
/*****************************************************************/
/*   Tabelle für die nächsten Zustände bei Zifferneingabe        */
/*****************************************************************/


   code unsigned char nextstate_tab [4][5]=
      {
      { 0,2,2,4,4 },
      { 1,1,3,3,4 },
      { 0,0,1,2,3 },
      { 0,1,2,3,0 }
      };



/**** Externer Interrupt 1 wird von der  Tastatur aufgerufen     ****/
/**** Tastencode wird von Port 6 eingelesen   ****/

void EXTINT1 (void) interrupt 2
{
  taste = P6;
  taste >>= 3;
  gedrueckt = 1;
}




/*****************************************************************/
/*   Funktion: MAIN(void)                                        */
/*                                                               */
/*   Parameter:keine                                             */
/*   Returnval: keiner                                           */
/*   Kommentar: Hauptprogramm                                 */
/*****************************************************************/

main ()
{

  EAL=0;
  input=0;
  state=0;  /* Anfangszustände Menü STATE MACHINE       */
             /*definieren */
  taste=0;
  gedrueckt=0;     /* Tastenflag wird von INT1 Routine gesetzt*/

  EX1=1;        /*  EXT1 Tastaturinterrupt erlauben */
  IT1=1;        /*  EXT1 Interrupt1  auf fallende Flanke getriggert */
  init_lcd();
  blank_lcd);
  lcd_ser = 1;  /* printf schreibt auf LCD */
```

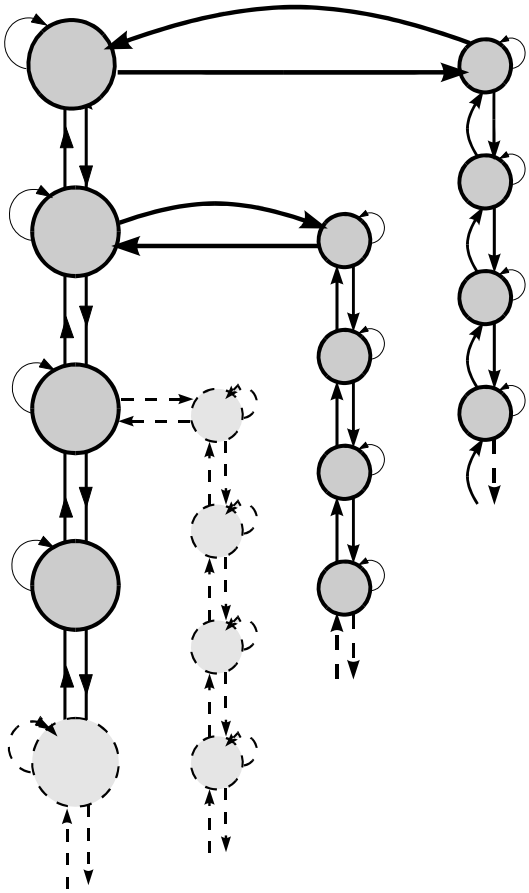
```
   pos(1,1);
   printf("State Machines");

   warte(3000);
   blank_LCD();

   EAL=1;


/* Beginn der Hauptschleife */


   while(1) {

      if(gedrueckt)
         {

         EAL=0;          /* alle Interrupts sperren */
         gedrueckt=0;    /* Bit rücksetzen */
         input=decoder_tab[taste];  /*Eingangskodierung abfragen */
                                    /* Buchstabe , Enter...*/
         (*fpnt[input][state])();   /* Suche in Tabellen nach */
                                    /*auszuführender Funktion */
         state=nextstate_tab[input][state];   /* und nächstem Zustand
*/


         EAL=1;
     }
   }
}
```

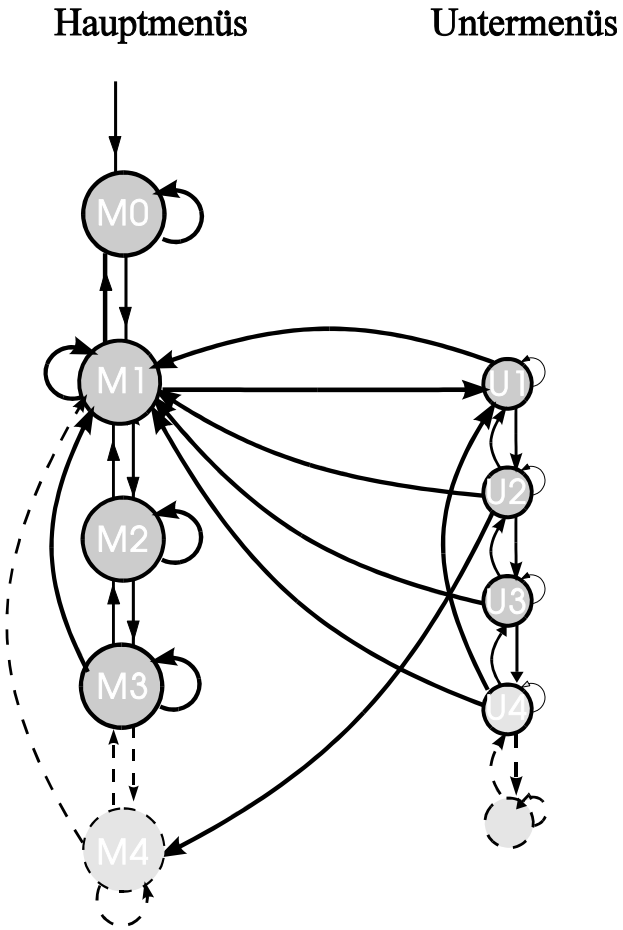
Anregung :

Ändern sie die Funktionsinhalte der Ausgangsfunktionen (Aktionen) ihren Anforderungen entsprechend.

Prinzip der Menüsteuerung mit STATE MACHINES

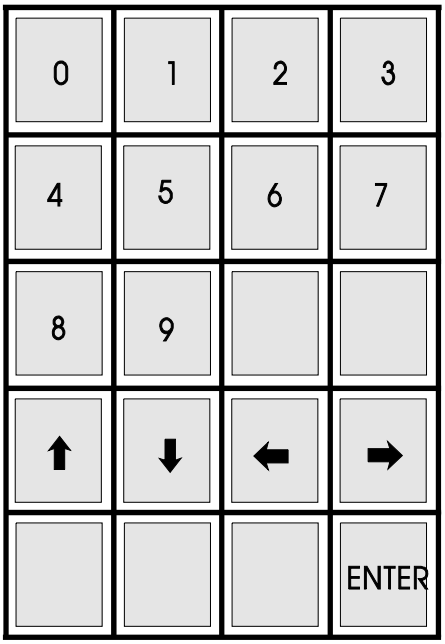
Hauptmenüs Untermenüs





5 Tastaturkodierung für eine Menüführung

Vorschlag:



5.1 TABELLE für die nächsten Menüzustände :

		0	1	2	3	4
0..9	0					
↑	1					
←	2					
→	3					
↓	4					
C/Jmp	5					
ENTER	6					

```
/*****************************************************/
/*       Tabelle für die MENÜ Funktionen         */
/*****************************************************/

 code (*menu_pnt[7])()= {  M0,M1,M2,M3,M4,M5,SUB1 };
```

- **M0,M1...M5 Sub1 etc. sind mögliche Funktionsnamen**

5.2 BEISPIEL:

```
/*****************************************************/
/*          Tabelle für die MENÜ State Machine       */
/*****************************************************/
 code unsigned char nextmenu_tab [7][7]=
      {
      { 0,1,2,3,4,5,6 },
      { 0,1,1,2,3,4,5 },
      { 0,1,2,3,4,5,6 },
      { 0,1,2,3,4,5,6 },
      { 1,2,3,4,5,5,6 },
      { 0,1,2,3,4,5,6 },
      { 0,1,6,3,4,5,6 }
      };
```

Aufruf der Funktionen:

```
main(){
  init_LCD();
  blank_LCD();
  M0();
  wait_taste();
  while(1) {
     if(gedrueckt)
      {
        EA=0;
         gedrueckt=0;
         input=decoder_tab[taste];
         menu=nextmenu_tab[input][menu];
         (*menu_pnt[menu])();
        EA=1;
      }
  }
}
```

6 TABELLEN :

Zustandstabellen

Ein momentaner Zustand Aus

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0																				
1																				
2																				
3																				
4																				
5																				
6																				

Ein momentaner Zustand Aus

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0																				
1																				
2																				
3																				
4																				
5																				
6																				