Modul - PORT I/O:

Version 0.9 BETA – Oktober 2003

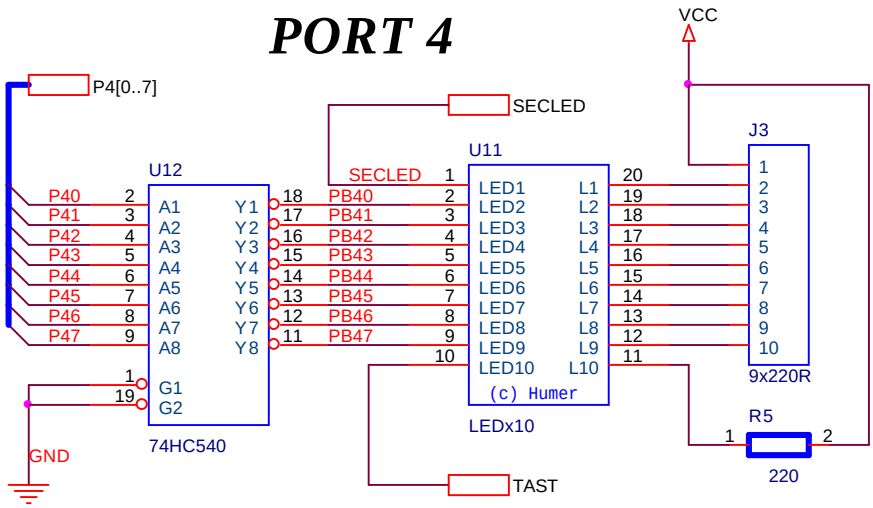
Reischer / Humer

1 Ausgabe an Port 4:

1.1 LED - Balken

Für die Visualisierung von digitalen Ausgangszuständen wurde ein LED-Balken gewählt. Der LED Balken besteht aus 10 einzelnen LEDs, die in einem Gehäuse vergossen sind. Für die Darstellung der Ausgangszustände von Port 4 werden jedoch nur 8 LEDs benötigt, sodass die übrigen 2 LEDs anderwertig verwendet sind. Die oberste LED (Anordnung auf der Platine) leuchtet bei einem Tastaturinterrupt kurz auf, die unterste LED stellt das 1Hz-Signal optisch dar.

1.2 Schaltplan

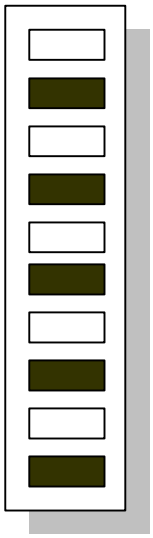


1.3 Anwendung

Der LED – Balken kann digitale Signal am Port 4 visualisieren. Für diese Form eignen sich Programmbeispiele wie Lauflicht, Binärzähler udgl. Die LED-Balkenanzeige am Port 4 kann jedoch auch noch unterschiedliche Helligkeiten durch die alternative PWM-Funktion darstellen (siehe Modul PWM).

1.4 HEX Zahl ausgeben

Nachfolgend ist ein sehr einfache Programm für die Ausgabe von HEX-Zahlen aufgelistet. Das Programmieren eines einzelnen Portpins auf den Zustand „1“ bringt die LED zum Leuchten. Selbstverständlich kann mit einem einzigen Befehl der gesamte Port P4 beschrieben werden.



Anzeige Tastaturinterrupt

P 4.0

P 4 .1

P 4.2

.
.
.
.
.

Anzeige – Sekundentakt (1 Hz)

= ON

= OFF

1.4.1 Listing

```
/* ****************************************************   */
/* Dateiname:              Hexaus.c                       */
/* Name:    Matthias Reischer, 5CIH, 2003/2004            */
/* Datum:                                  8.11.2003      */
/* ****************************************************   */

#include <reg517a.h.>
main()
{
     while(1) // Der Wert 1 wird nie erreicht = Endlosschleife
     {
          P4 = 0x55;      // MSB 0101 0101 LSB
                          //     |       |
                          //    P4.7    P4.0
     }
}
```

1.5 Lauflicht

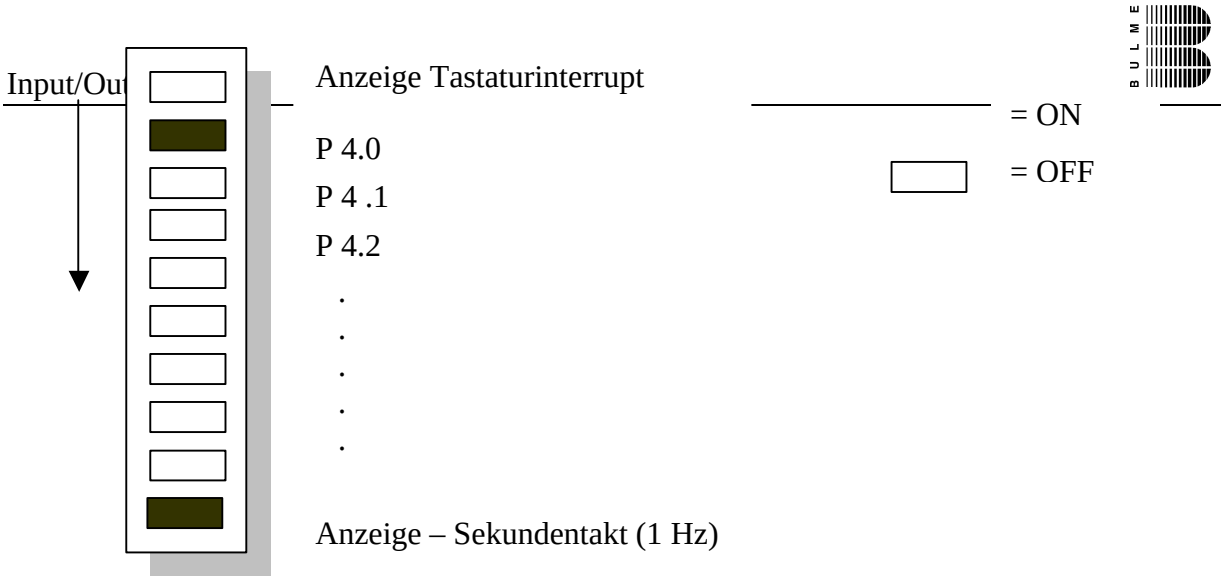


Bild: Momentaufnahme

Das Licht soll sich immer nur in eine Richtung bewegen. Als Beispiel sei hier der Lauf nach unten gezeigt. Es wird hierbei ein „Schiebbefehl“ verwendet (<< bzw. >>). Damit dieses Lauflicht nicht zu schnell abläuft, wurde eine Verzögerungsschleife eingebaut.

Die Oberfläche in Keil schaut wie folgt aus:

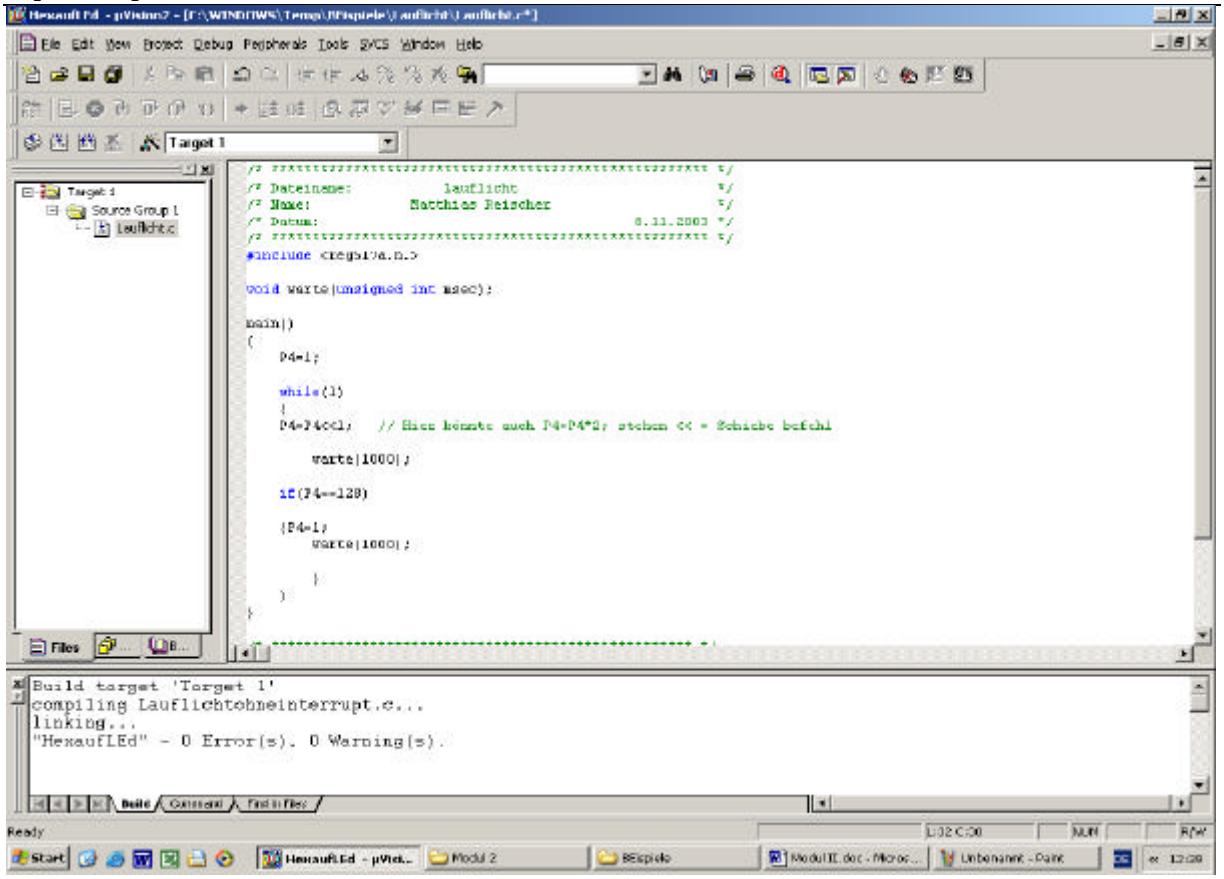
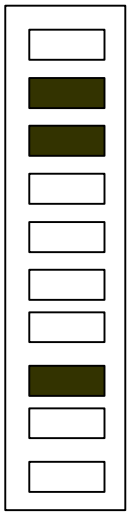


Bild: Arbeiten mit dem Keil-Compliler (uVision)

1.5.1 Listing

```
/* ************************************************   */
/* Dateiname:              Lauflicht                   */
/* Name:    Matthias Reischer, 5CIH, 2003/2004         */
/* Datum:                                 8.11.2003    */
/* ************************************************   */

#include <reg517a.h.>

void warte(unsigned int msec); // Prototyp der Fkt. warte()

main()
{
     P4=1;           // Setzen von Portpin P4.0
     while(1)        // Endlosschleife
     {
      P4=P4<<1; // Hier könnte auch P4=P4*2; stehen << = Schiebe
                //befehl

          warte(1000);    // Verzögerungsschleife

       if(P4==128)   // Ist P4.7 gleich 1 ?
          {
           P4=1;
           warte(1000);
          }
     }
}

/* ************************************************  */
/*                      Funktionen                    */
/* ************************************************  */

void warte(unsigned int msec)
{
     data char i;
     for (msec; msec!=0; msec--)
     for (i=0; i<100; i++);
}
```

1.6 Binärzähler



Anzeige Tastaturinterrupt

P 4.0

P 4 .1

P 4.2

.
.
.
.
.

Anzeige – Sekundentakt (1 Hz)

= ON

= OFF

(So sollte es fertig aussehen)

Der Binärzähler verläuft vom Programm Code her ähnlich wie das Lauflicht, es sind nur ein paar kleine Änderungen vorzunehmen. Ansonsten ist noch zu bemerken, dass der Zähler nur bis 255 zählt. Er muss bei 255 abgefangen und wieder auf 1 zurückgesetzt werden, da es sonst ein ERROR entsteht. Der Schiebe befehl muss außerdem durch ein P4++ ersetzt werden.

1.6.1 Listing

```
/* ************************************************** */
/* Dateiname:              Binärzähler                */
/* Name:              Matthias Reischer               */
/* Datum:                                  8.11.2003  */
/* ************************************************** */
#include <reg517a.h.>

void warte(unsigned int msec);

main()
{
     P4=1;

     while(1)
     {
     P4++;
          warte(1000);

     if(P4==255)

     {P4=1;
          warte(1000);

          }
     }
}

/* ************************************************** */
/*                    Funktionen                      */
/* ************************************************** */

void warte(unsigned int msec)
{data char i;
     for (msec; msec!=0; msec--)
     for (i=0; i<100; i++);
}
```

2 Input

2.1 Tastatur

Für Digital-In kann die am Microcontrollerboard vorhandene Tastatur verwendet werden. Es befinden sich 12 Tasten auf dem Board die in einer Matrixschaltung angeordnet sind.

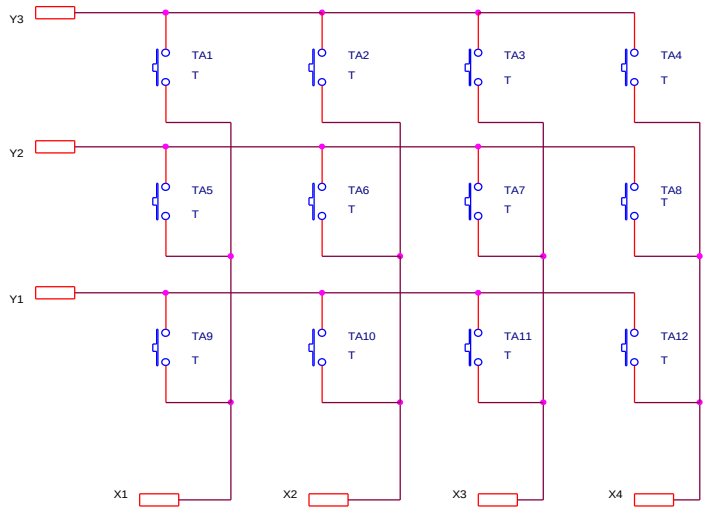


Bild: Matrixanordnung der Tastatur

Um die Tasten zu Entprellen und entsprechend zu kodieren, wurde ein eigener Microcontroller vorgesehen.

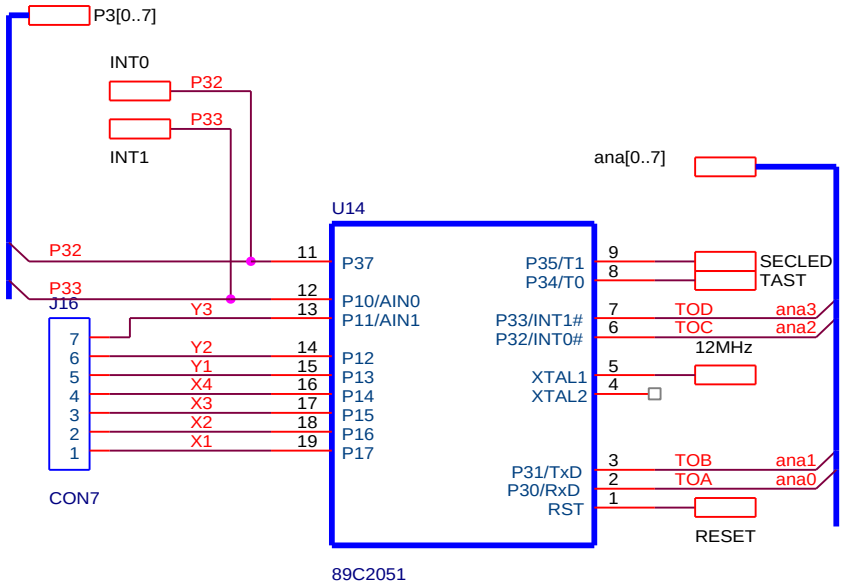


Bild: Tastaturentprellung und –kodierung mit einem eigenen Microcontroller

Der Baustein 89C2051 (U14) stellt den Tastenkode auf den Anschlüssen TOA...TOD zur Verfügung. Diese Anschlüsse sind mit den Portleitungen P7.5 bis P7.7 verbunden.
Dieser Microcontroller (U14) generiert an P37 einen Interrupt (negative Flanke, verbunden mit dem INT0 Eingang des 80C517A). Eine entsprechende Visualisierung erfolgt über eine LED (TAST).
Weiters generiert der Baustein U14 auch einen Sekundentakt – Verbunden mit INT1 des 80C517A, bzw. mit einer LED der Balkenanzeige.

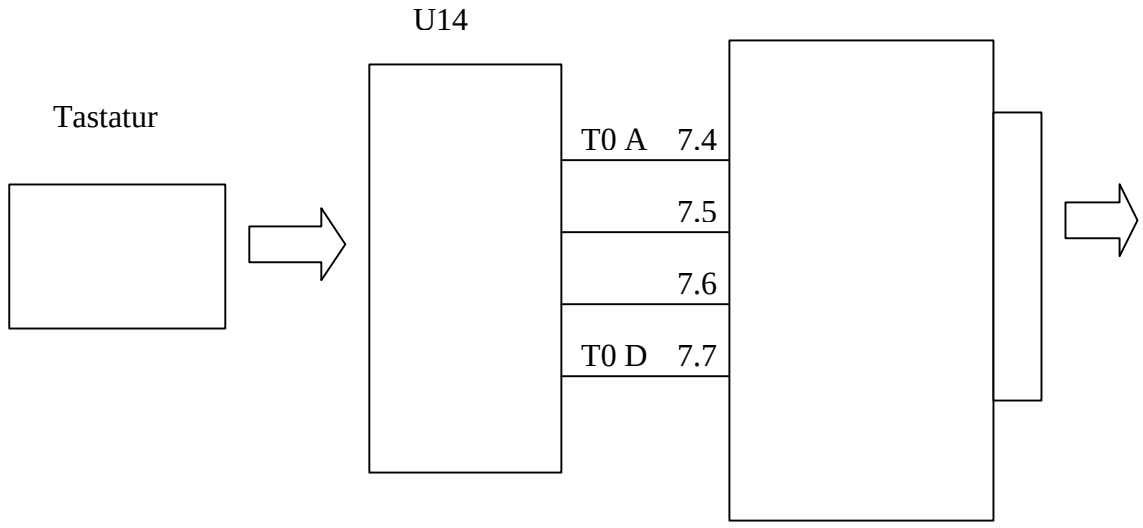
Der Tastaturkode kann sehr einfach mit dem Befehl (P7>>4) eingelesen werden.

Beispiel: tasten_code=P7>>4; // Datentyp char

2.2 Einlesen der Tastatur

Es soll ein kleines Programm geschrieben werden. Der Tastenkode soll eingelesen werden und an der LED Balkenanzeige dargestellt werden.

2.2.1 Blockschaltbild



2.2.2 Listing

```
/* ************************************************** */
/* Dateiname:         tast_in                         */
/* Name:              Matthias Reischer               */
/* Datum:                                  15.11.2003 */
/* ************************************************** */

#include <reg517a.h.>

main()
{
      P4 = 0x00;
      while(1)
      {
      P4 = P7>>4;
      }
}
```

2.3 Der Tastaturkode

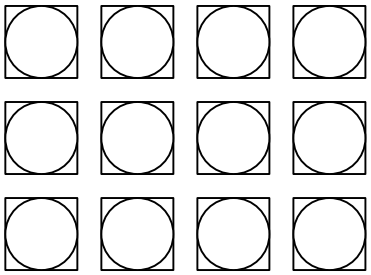


Bild : Anordnung der Tasten

1	9	5	13
2	10	6	14
0	8	4	12

Bild: Hardware mäßig festgelegter Tastaturkode

2.4 Tastaturkode umlegen

Es soll ein Programm geschrieben werden in dem die Tastatur auf eine gewisse Zahlenreihenfolge umgelegt wird (1,2,3…).

2.4.1 Tastaturänderung:

IST:

1	9	5	13
2	10	6	14
0	8	4	12

Bild: Zeigt die Hardware mäßig festgelegte Zahlenfolge

SOLL:

8	9	10	11
4	5	6	7
0	1	2	3

Bild: Zeigt die umgelegte Zahlenfolge

2.4.2 Listing

```
/* ************************************************** */
/* Dateiname:         tast_um                         */
/* Name:              Matthias Reischer               */
/* Datum:                                  15.11.2003 */
/* ************************************************** */

#include <reg517a.h.>

unsigned char taste [15] = {0,8,4,12,2,10,6,14,1,9,5,3,3,14,7};

main()
{
      P4 =0x00;
      while(1)
      {
      P4 = taste[P7>>4];
      }
}
```