9.1.3 Maxim MAX7219 LED-Display Treiber

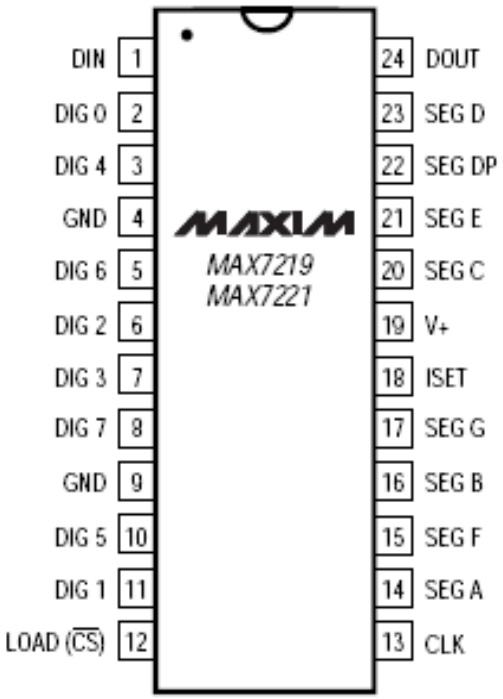
Bei diesem IC handelt es sich um einen seriell anzusteuernden Display Treiber, der hauptsächlich zur Ansteuerung einer großen Anzahl an LEDs bzw. 7-Segmenten verwendet wird. Dabei müssen 7-Segmente mit gemeinsamer Kathode (common-cathode) eingesetzt werden.

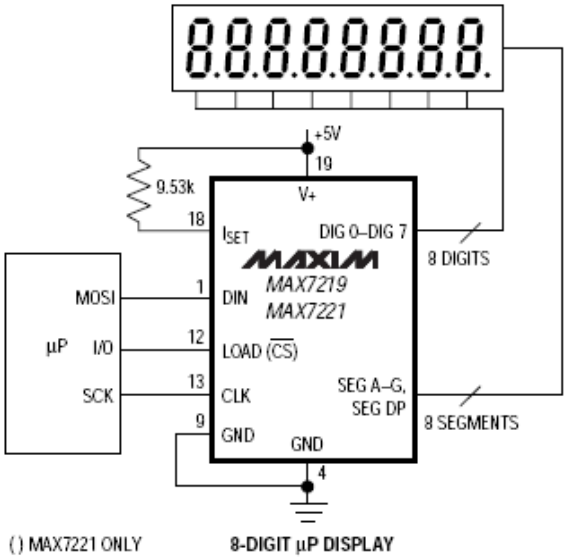
Er bietet unschätzbare Vorteile gegenüber anderen Ansteuerungsmethoden:

- minimalen Außenbeschaltung, von nur einem Widerstand
- Ansteuerung von bis zu 8 7-Segment Digits oder 64 einzelnen LEDs
- „On-chip BCD Code-B“ Dekoder und „No-Decode“ Modus
- Interner Multiplexer
- Segment und Digit Treiber
- 8x8 statisches RAM
- Belegung von nur 3 I/O Pins des Mikrokontrollers
- 150µA Shutdown Mode zum Stromsparen
- Analoge und digitale Helligkeitskontrolle
- Test Mode für einen Funktionstest der 7-Segmente (alle LEDs eingeschaltet)

Durch den eingebauten RAM und dem Multiplexer, entlastet der MAX7219 den Mikrokontroller enorm. Außerdem ist der MAX7221, eine verbesserte Variante im Bereich EMV, mit den Ansteuerungsmethoden SPI, QSPI und Microwire kompatibel.

Pinout des Treibers: Standardschaltung:





Der MAX7219 ist in seiner Bauform als PDIP im Rastermaß 7,62mm ausgeführt und wird mit +5V versorgt.

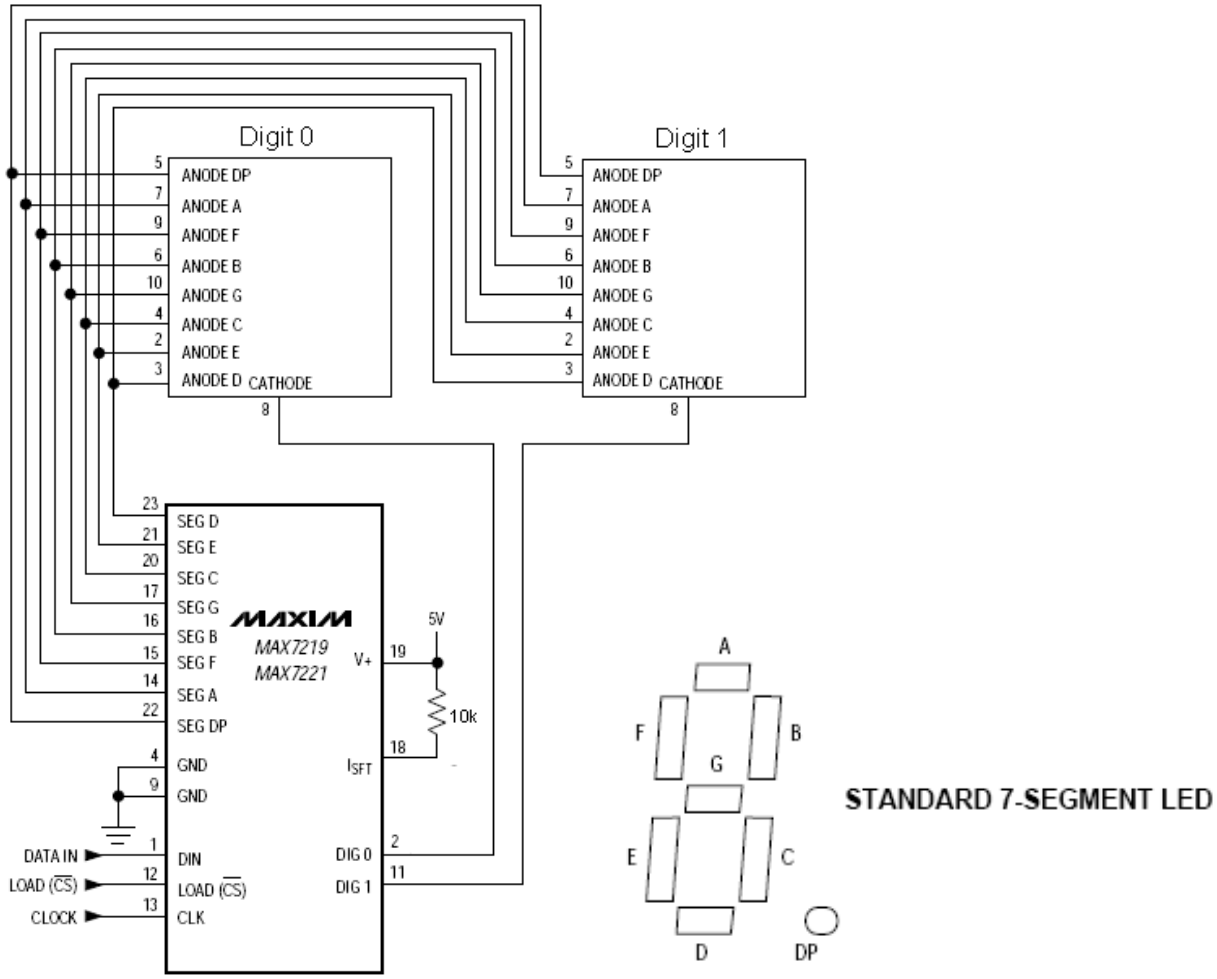
Die Helligkeit der 7-Segmente kann hardwaremäßig mittels eines Widerstandes zwischen V+ und ISET gewählt werden. Je kleiner dieser Widerstand, desto mehr Strom fließt in die 7-Segmente. Ein zu kleiner Widerstand würde ein 7-Segment überlasten, was zu einer thermischen Zerstörung führen würde. Bei 10Ohm würde ein Segment mit rund 40mA belastet werden.
Wir haben einen 10kOhm Widerstand für unsere Displays eingesetzt um eine Überlastung zu verhindern und um die Lebensdauer der Displays nicht unnötig zu verkürzen.
Die Helligkeit kann auch digital, mittels eines Kontrollregisters geändert werden.

Die Pins SEG A - SEG G müssen mit den entsprechenden 7-Segmenten verbunden werden.
Der Pin SEG DP ist der Kommapunkt eines jeden Digits.
Der MAX7219 schaltet nacheinander DIG 0 bis DIG 7 in einer Frequenz von rund 800Hz gegen GND. Wenn zum Beispiel DIG 0 momentan auf GND geschaltet ist, schalten der MAX7219 die SEG-Ausgänge entsprechend der anzuzeigenden Zahl

gegen V+ → die Zahl erscheint am Digit. Anschließend erfolgt das Gleiche bei DIG 1 – DIG 7. Da das menschliche Auge zu langsam ist, um ein Blinken der Anzeigen zu erkennen, nehmen wir die angezeigten Messergebnisse als starres Bild auf.

9.1.3.1 Schematische Darstellung

… wie die 7-Segment Digits an den MAX7219 angeschlossen werden:



Die Kommunikation zwischen Mikrokontroller und MAX7219 erfolgt auf serieller Basis über 3 Leitungen.

DIN: Dieser Input Pin ist für die seriellen Daten zuständig. Er empfängt die vom Mikrokontroller gesendeten Daten in 16-bit Paketen.

CLK: An diesem Input Pin liegt ein Takt an, der vom Mikrokontroller generiert wird. Positive Signalflanken veranlassen das Einlesen der Daten in ein Schieberegister des MAX7219.

LOAD: Hier wird nach jedem empfangenen 16-bit Datenpaket ein kurzer negativer Impuls angelegt. Mit der steigenden Flanke dieses Impulses werden die Daten in das zuvor angesprochene Register des MAX7219 geladen.

9.1.3.2 Serieller Adressierungsmodus

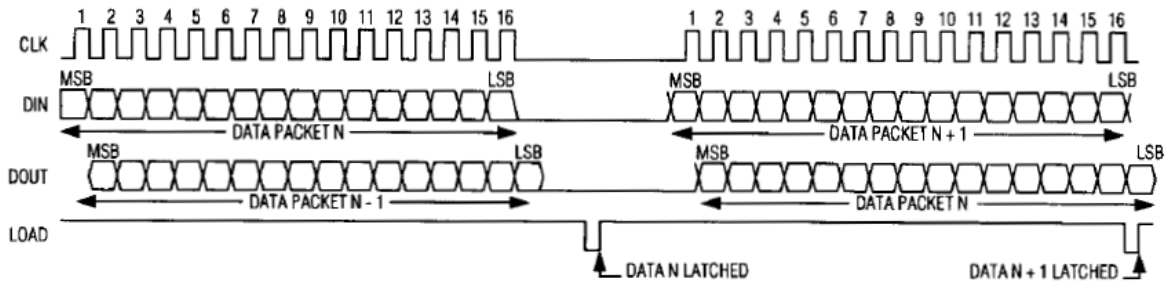
Das serielle Datenformat des LED-Treibers:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				MSB	DATA						LSB

9.1.3.3 Funktionsprinzip

In den MAX7219 werden die seriellen Daten an DIN, die in 16-bit Paketen gesendet werden, bei jeder steigenden Signalflanke an CLK in ein internes 16-bit Schieberegister abgelegt. Dies geschieht unabhängig von Ereignissen, die auf LOAD auftreten.
Die Daten im Schieberegister werden anschließend mit einer steigenden Signalflanke an LOAD in ein Kontrollregister (Intensity, Decode Mode,…) oder ein Digitregister (Digit 0, Digit 1,…) geschrieben. LOAD muss unmittelbar nach 16 Takten an CLK auf „high“ gehen, da sich nach 16 Takten ein Datenpaket vollständig im 16-bit Schieberegister befindet. Wird LOAD zu spät auf „high“ gesetzt, gehen Daten verloren.
Die Daten, die sich im Schieberegister befinden, tauchen 16,5 Taktzyklen später am Ausgangspin DOUT des MAX7219 auf.
Die Daten werden mit der fallenden Flanke von CLK ausgegeben.

9.1.3.4 Timing Diagramm



- Die Bits D15-D12 sind Dummy Bits. Sie können beliebig beschrieben werden und haben keinen Nutzen.
- Die Bits D11-D8 beinhalten die Registeradressen.
- Die Bits D7-D0 beinhalten die eigentlichen Daten.

Adressenübersicht (D8-D11):

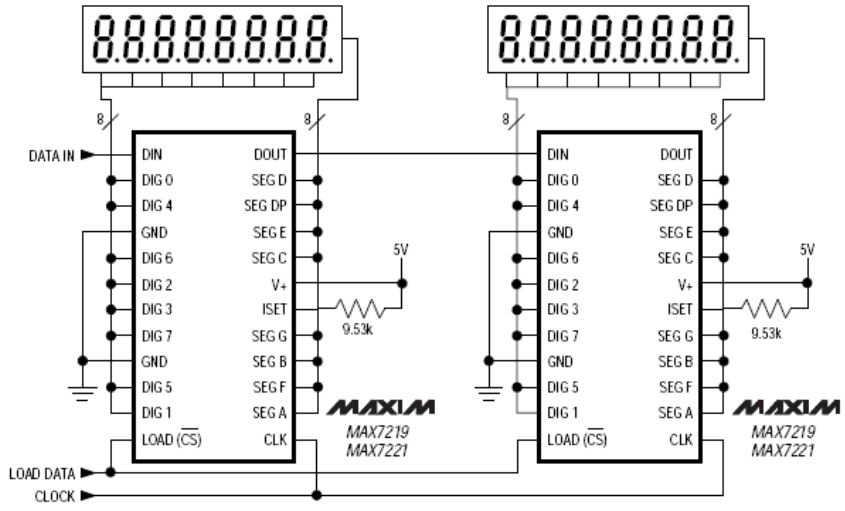
REGISTER	ADDRESS					HEX CODE
	D15–D12	D11	D10	D9	D8	
No-Op	X	0	0	0	0	X0
Digit 0	X	0	0	0	1	X1
Digit 1	X	0	0	1	0	X2
Digit 2	X	0	0	1	1	X3
Digit 3	X	0	1	0	0	X4
Digit 4	X	0	1	0	1	X5
Digit 5	X	0	1	1	0	X6
Digit 6	X	0	1	1	1	X7
Digit 7	X	1	0	0	0	X8
Decode Mode	X	1	0	0	1	X9
Intensity	X	1	0	1	0	XA
Scan Limit	X	1	0	1	1	XB
Shutdown	X	1	1	0	0	XC
Display Test	X	1	1	1	1	XF

Das NO-OP Register 0x00

Dieses Register wird bei Kaskadierung von mehreren MAX7219 beschrieben.

Bei Kaskadierung werden alle LOAD und CLK Eingänge miteinander verbunden. Der DOUT-Pin des vorhergehenden MAX7219 wird mit dem DIN-Pin des nachfolgenden MAX7219 verbunden.

Beispiel einer Kaskadierung von mehreren MAX7219:



Wenn man zum Beispiel in das 6te Display schreiben will, muss das gewünschte 16-bit Wort für Display 6 gesendet und anschließend 5-mal das NO-OP Register 0x00 mit beliebigen Dummy Bits beschrieben werden.

Shutdown Register 0x0C

Wenn das Shutdown Register 0x0C mit „0“ beschrieben wird, werden alle Anoden der 7-Segment LEDs auf GND gelegt. Das bewirkt ein komplettes Ausschalten der 7-Segmente.

Wird das Register mit „1“ beschrieben, werden alle Anoden der 7-Segment LEDs wieder auf die internen Steuerleitungen gelegt.

Im Shutdown Mode bleiben alle Daten erhalten und der MAX7219 kann programmiert werden.

Der Shutdown Mode kann als Stromspar Funktion oder als blinkende Warnsignalisierung genutzt werden.

Decode Mode Register 0x09

In diesem Register wird gewählt, ob der BCD-Code (0-9, E, H, L, P, -), oder ob kein Code für die Ansteuerung der Digits gewählt wird. Wenn kein Code gewählt wird, muss man sich die gewünschten Zeichen und Zahlen selbst programmieren.
Jedes Bit in diesem Register gehört zu einem Digit.
Da wir nur Zahlen ausgeben wollen, wird in unserem Programm der BCD-Code für alle Digits mit den Registerdaten 0xFF verwendet.

DECODE MODE	REGISTER DATA								HEX CODE
	D7	D6	D5	D4	D3	D2	D1	D0	
No decode for digits 7–0	0	0	0	0	0	0	0	0	00
Code B decode for digit 0 No decode for digits 7–1	0	0	0	0	0	0	0	1	01
Code B decode for digits 3–0 No decode for digits 7–4	0	0	0	0	1	1	1	1	0F
Code B decode for digits 7–0	1	1	1	1	1	1	1	1	FF

Intensity Register 0x0A

Mit diesem Register kann die Helligkeit der 7-Segmente bestimmt werden.
Die geringste Helligkeit kann mit 0x00, die höchste Helligkeit mit 0x0F gewählt werden. Der MAX7219 betreibt intern eine Pulsweitenmodulation, die die Helligkeit entsprechend dem gewählten Wert ändert.

Scan Limit Register 0x0B

Hier wird gewählt, mit wie vielen Digits gearbeitet werden soll.
Die Auswahl reicht von 0x00 (1 Digit) bis zu 0x07 (8 Digits).
Zu beachten ist, dass bei hoher Anzahl an Digits, die Helligkeit aufgrund der niedrigeren Abtastrate sinkt. Die Abtastrate beträgt 800Hz bei 8 verwendeten Digits.

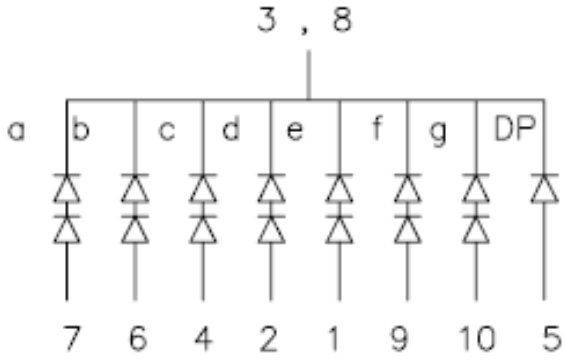
Display Test Register 0x0F

Dieses Register wird für das Testen der Segment Funktion genutzt.
Im normalen Modus kann das Display für Ausgaben genutzt werden. Im Testmodus werden alle Segmente mit Strom versorgt und leuchten auf. Dies kann für einen Funktionstest verwendet werden.

Wir entschieden uns aufgrund der vielen Vorteile gegenüber anderen Methoden der Ansteuerung für die Maxim MAX7219 LED-Treiber.

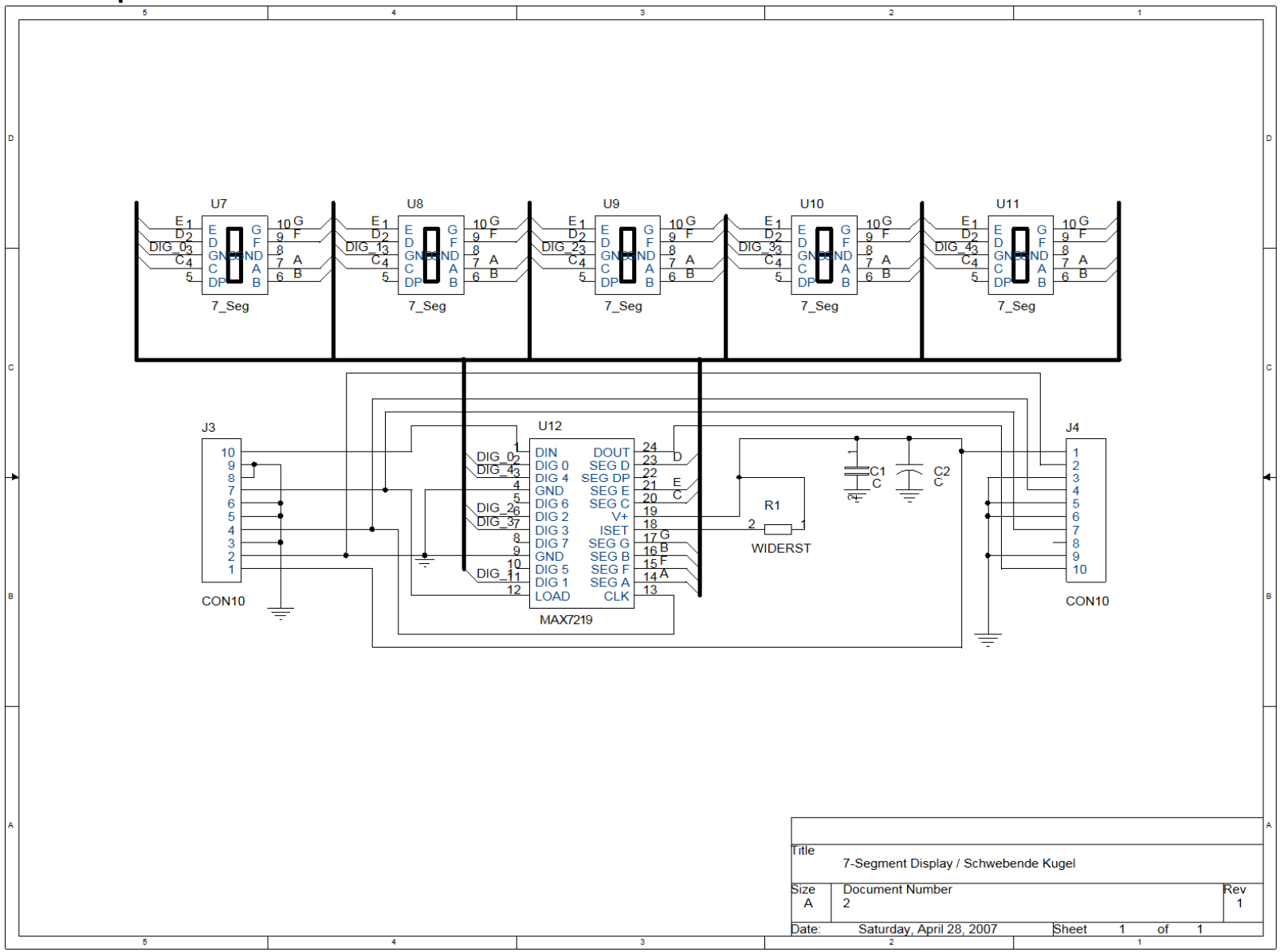
Es wurden 45 1.0inch 7-Segment Digits bei RS-Components der Marke KINGBRIGHT in der Standardfarbe Rot bestellt (Best.-Nr: 235-8985). Bei dieser Ausführung handelt es sich um eine Version mit gemeinsamer Kathode (common cathode), da diese für den MAX7219 7-Segment Treiber benötigt wird.

Anschlussschema

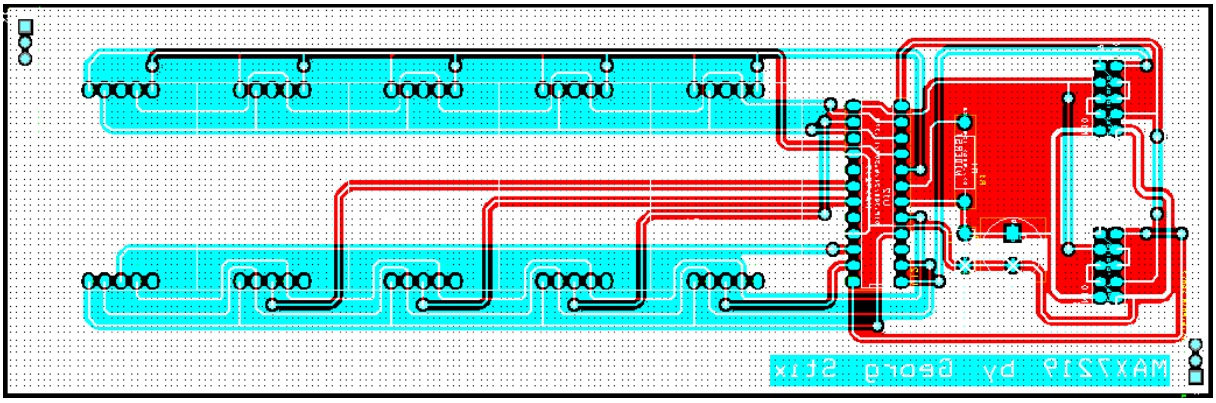


Anschließend wurde der Schaltplan des Displays mittels OrCAD Capture gezeichnet und mit OrCAD Layout Plus geroutet.

9.1.3.5 Schaltplan



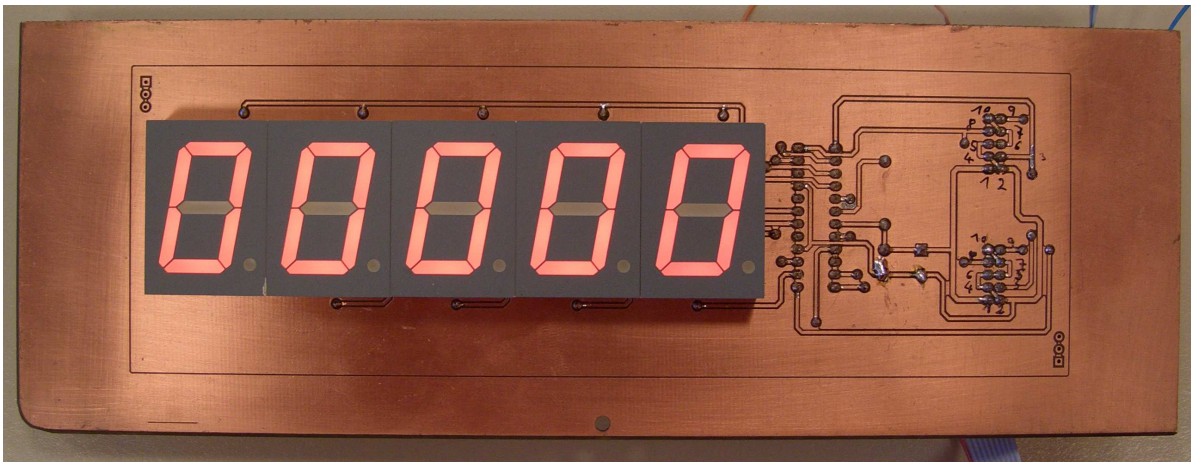
9.1.4 Geroutete Platine



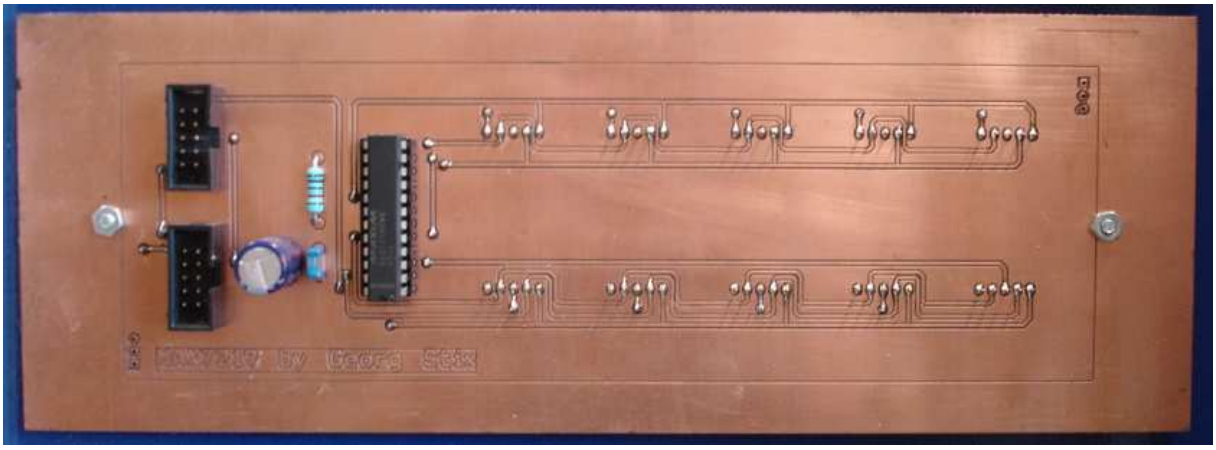
Aufgrund der vielen Leiterbahnen und der Tatsache, dass sich die Digits wegen Platzmangel bei Montage der Displays auf die Frontplatte auf einem anderen Layer als der MAX7219 befinden müssen, mussten doppelseitige Leiterplatten gefräst werden.
Die Dummy Footprints in der linken oberen und rechten unteren Ecke dienten der Nachjustierung bei Drehung der Leiterplatte zum Fräsen des Top-Layers, nachdem der Bottom-Layer gefräst und gebohrt wurde. Gefräst wurde die Platine mit dem hauseigenen PCB-Fräser.

9.1.5 Fertig aufgebaute Platine mit Digits und MAX7219

Top-Layer



Bottom-Layer



9.1.5.1 Technische Details

Die Versorgungsspannung des Displays ist +5V, die über ein Flachbandkabel von der Mikrokontrollerplatine geliefert wird. Unmittelbar nach der Spannungsversorgung über die Pfosten-Stiftleiste J4, wird die Spannung mittels des Kondensators C2 geglättet. Der nachgeschaltete Kondensator C1 dient als Filter gegen hohe Spannungsspitzen. Die Versorgungsspannung wird anschließend zum MAX7219 und zur zweiten Pfosten-Stiftleiste J3 geleitet, welche zum Anschluss des nächsten Displays dient.

An den Pins DIG 0 – DIG 4 wurden die gemeinsamen Kathoden der 5 Digits angeschlossen. An den Pins SEG A – SEG G wurden die einzelnen Segmente aller Digits angeschlossen.

Zwischen den Pins V+ und ISET des MAX7219 befindet sich ein 10kOhm Widerstand, der zum Einstellen der Helligkeit der Segmente dient.

9.1.6 Stückliste

Bezeichnung	Beschreibung	Anzahl
U12	Maxim MAX7219CNG	9
	IC-Fassung 24-polig Dual-Inline RM 7,62mm	9
R1	Metallschichtwiderstand 10kOhm	9
C2	Elektrolytkondensator 220µF	9
C1	Folienkondensator 100nF	9
J3, J4	Pfostenstiftleiste 10-polig	18
U7-U11	7-Segment Anzeige 1.0inch rot	45

9.1.7 Programm

Da es sich beim C515 um ein 8051-Derivat handelt, dessen Programm sich problemlos in der Hochsprache „C“ programmieren lässt, haben wir uns für „C“ entschieden.

Da uns „C“ seit der ersten Klasse unserer HTL-Laufbahn bekannt ist, mussten wir keine Einarbeitungsphase für die C-Syntax aufwenden.

Lediglich das Prinzip der seriellen Ansteuerung, der Maxim MAX7219CNG LED-Display Treiber, musste aus dem Datenblatt des Bausteins entnommen werden.

9.1.8 Allgemein

Um die Messwerte auf den 7-Segment Anzeigen ausgeben zu können, müssen Daten in benötigte Control und Digitregister des MAX7219 geschrieben werden. Am Mikrokontroller werden lediglich 3 I/O Pins für die Kommunikation benötigt.

Die benötigten Pins wurden sinnvollerweise im Programm mit den I/O Pins des MAX7219 bezeichnet.

DIN: Dieser Pin ist für die seriellen Daten zuständig. An ihm liegen die zu sendenden Daten, die in 16-bit Paketen versendet werden.

CLK: An diesem Pin wird ein Takt generiert, dessen positive Signalflanken für das Einlesen der Daten in ein Schieberegister des MAX7219 zuständig sind.

LOAD: Hier wird nach jedem gesendeten 16-bit-Datenpaket ein kurzer negativer Impuls erzeugt. Mit der steigenden Flanke werden die Daten in das zuvor angesprochene Register des MAX7219 geladen.

9.1.9 Wichtige Programmschritte

- Initialisierung der Displays
- Warten auf Startbefehl
- Abspeichern der Messergebnisse
- Messergebnisse auf Displays ausgeben
- Messergebnisse an Computer senden
- Rückkehr zu Punkt 2.)

9.1.10 Source Code vom C515 Hauptprogramm

C:\Schule2006\Diplomarbeit\Programme\Hauptprogramm+Displays\seriell_dokument...\Final.C

```
/* Diplomarbeit Schwebende Kugel Stix / Gamperl  */

/* Mikrokontroller Hauptprogramm und Ansteuerung der 7-Segment Displays  */


#include <reg515.h>      //C515 Header File

//Taster
sbit s = 0x90;           //Taster zum freigeben auf Pin P1.0
sbit test = 0x91;        //Taster 7-Seg Test auf Pin P1.1
sbit sensor10 = 0x92;    //Sensor 10 auf Pin P1.2

//Hilfsbits
bit _s, hs, _test, htest, a, b, _sensor10, hsensor10;

//Zaehlgroesse 0-4.294.967.295
unsigned long hx;

//Feld t1 in dem die Timerwerte abgelegt werden
unsigned long t1[10];

//Feldvariable
unsigned char x;


/*****************
Konstante Register
******************/
//Hier werden KontrollRegister definiert

#define REG_DECODE 0x09
#define REG_INTENSITY 0x0a
#define REG_SCAN_LIMIT 0x0b
#define REG_SHUTDOWN 0x0c
#define REG_DISPLAY_TEST 0x0f

/*******************
Konstanten Allgemein
*******************/
//6 Routinen, um die Ausgangspins high oder low zu setzen

#define DIN_PORT P4
#define DIN_DDR P4
#define DIN_BIT 0x04    //liegt auf Pin10 der Pfosten-Stiftleiste
#define DIN_0()         (DIN_PORT &= ~DIN_BIT)    //Pin 4.2 auf low
#define DIN_1()         (DIN_PORT |= DIN_BIT)     //Pin 4.2 auf high


#define CLK_PORT P4
#define CLK_DDR P4
#define CLK_BIT 0x01    //liegt auf Pin4 der Pfosten-Stiftleiste
#define CLK_0()         (CLK_PORT &= ~CLK_BIT)     //Pin 4.0 auf low
#define CLK_1()         (CLK_PORT |= CLK_BIT)      //Pin 4.0 auf high

#define LOAD_PORT P4
#define LOAD_DDR P4
#define LOAD_BIT 0x10  //liegt auf Pin7 der Pfosten-Stiftleiste
#define LOAD_0()        (LOAD_PORT &= ~LOAD_BIT) //Pin 4.4 auf low
#define LOAD_1()        (LOAD_PORT |= LOAD_BIT)  //Pin 4.4 auf high

#define INTENSITY_MAX 0x0f      //Konstante fuer maximale Leuchtstaerke


/*************
Unterprogramme
**************/
```

```
void MAX7219_Init (void);
void MAX7219_ShutdownStart (void);
void MAX7219_ShutdownStop (void);
void MAX7219_DisplayTestStart (void);
void MAX7219_DisplayTestStop (void);
void MAX7219_SetBrightness (char brightness);
void MAX7219_Clear (void);
void ziffern (void);

void MAX7219_Write (unsigned char reg_number, unsigned char dataout);

unsigned char z; //wird fuer den NO-OP Befehl des MAX7219 benoetigt

bit v; //dient zum einmaligen Ausgeben der Messwerte

void main()
   {
   unsigned char y;      //For-Schleifenvariable

   EAL=1;       //Alle Interrupts enabled
   EX0=1;       //externen Interrupt enablen
   IT0=1;       //externer Interrupt negative Flanke
   TMOD=0x22;   //8 bit auto-reload timer

   a=0;        //externer Interrupt gesperrt
   b=0;        //hx Ruecksetz Rountine soll durch den ersten Sensor aufgerufen werden

   x=0;        //damit Feld von 0 beginnt
   y=0;        //For-Schleifenvariable fuer Feld t1 0-setzen
   hx=0;       //Zaehlvariable 0-setzen

   //Port 1 als eingang:
   P1 = 0xFF;

   /*Hilfsvariablen fuer negative Flanken Abfrage */
   hs=s;
   htest=test;
   hsensor10=sensor10;


   /*****************************
   Serielle Kommunikation mit PC
   *****************************/

   /*Baudraten-Bestimmung (4800 Baud) fuer 12 MHZ Quarz
     SMOD = 1 -> TH1 = 242,979;
     243 gewaehlt -> Baud = 4807.69 (liegt innerhalb der Tolleranz */
   TH1 = TL1 = 243;

   /*Im Sfr PCON muss das Bit SMOD gesetzt werden */
   PCON |= 0x80;

   /*Setze SCON fuer "Serial mode 1 SM1=1" & "Receive enable REN=1" */
   SCON = 0x50;

   /*Timer1 einschalten*/
   TR1 = 1;

   /*Timer0 enablen*/
   ET0=1;

   v=0;
   for (z=0;z<=8;z++  //9 mal die Initilisierung durchfuehren
       )
      {
      CLK_0();
      CLK_1();
      MAX7219_Init(); //Initialisierung aufrufen um die Displays zu Initialisieren
```

```
      }
   while (1)
      {
       s = s;
      if (!_s & hs)
         {
         /*Timerwert vorladen*/
         TL0=TH0=0xEC; //256ms-236ms=20ms

         x=0;     /*Damit beim neuen Durchlauf die
                    Felder wieder von 0 beschrieben werden */
         b=1;     /*Damit hx 0-setzen nur Anfangs funktioniert*/

         TR0=0;  //Timer0 stop

         hx=0;    //Timerstand ruecksetzen
         a=1;     //externer Interrupt freigegeben
         for (y=1; y<=9; y++)
            {
            t1[y]=0;    //Timerstand 0-setzen
            }
         for (z=0;z<=8;z++)
            {
            MAX7219_Write(REG_DECODE, 0xFF);         //Decode Mode B einschalten fuer al
le 8 Digits
            MAX7219_SetBrightness(INTENSITY_MAX);    //Helligkeit der Segmente definiere
n
            MAX7219_Write(REG_SCAN_LIMIT, 0x04);     //5-Digits aktivieren
            MAX7219_ShutdownStop();                  //normaler Betrieb - kein Shutdown
Mode
            MAX7219_DisplayTestStop();               //stoppen des Displaytests
            MAX7219_Clear();                         //alle Digits loeschen
            }
         v=1;
         }
      hs = _s;

      _test = test;
      if (!_test & htest)
         {
         for (z=0;z<=8;z++)
            {
            MAX7219_DisplayTestStart(); // starten des Display Tests
            }
         }
      htest = _test;

       sensor10 = sensor10;
      if (x==10)
         {
         /*Wenn x 10 erreicht,
           soll der ext-Interrupt gesperrt
           werden und mit der Ausgabe der
           Messwerte begonnen werden*/
         TR0=0; //Timer0 stoppen
         a=0;    /*damit Interrupt gesperrt
                 wird (erneut scharfmachen) */

         ziffern (); //Ausgabe starten

         x=0;    //um die If-Anweisung zu sperren
         }
      else
         {
         if (!_sensor10 & hsensor10)
            {
            /*Wenn der 10-te Sensor anspricht,
              soll der ext-Interrupt gesperrt
```

```
              werden und mit der Ausgabe der
              Messwerte begonnen werden */

            TR0=0; //Timer0 stoppen
            a=0;   /*damit Interrupt gesperrt
                   wird (erneut scharfmachen) */

            ziffern (); //Ausgabe starten
            }
         }
      hsensor10 = _sensor10;
      }
   }

void Sensoren () interrupt 0    //externer Interrupt
   {
   if (a)
      {
      if (b)
         {
         TR0=1;      //Timer0 start
         b=0;        /*Damit nur der erste Sensor
                       Timer0 starten kann */
         }
      t1[x]=hx;   //Timerstand abspeichern
      x++;        //Feldvariable erhoehen
      }
   }

void INTTIM0 () interrupt 1     // Timer0
   {
   hx++;  //alle 20ms wird hx um 1 erhöht
   }

void MAX7219_Init (void)
   {
   // DIN, CLK & LOAD als Ausgang definieren
   DIN DDR |= DIN BIT;
   CLK DDR |= CLK BIT;
   LOAD_DDR |= LOAD_BIT;

   MAX7219 Write(REG DECODE, 0xFF);         //Decode Mode B einschalten f□r alle Digits
   MAX7219_SetBrightness(INTENSITY_MAX);    //Helligkeit der Segmente definieren
   MAX7219 Write(REG SCAN LIMIT, 0x04);     //5-Digits aktivieren
   MAX7219 ShutdownStop();                  //normaler Betrieb - kein Shutdown Mode
   MAX7219_DisplayTestStart();              //Test Betrieb - alle Digits leuchten!
   }

void MAX7219_ShutdownStart (void)

/* Schaltet das Display ab - Shutdown Mode */
   {
   MAX7219_Write (REG_SHUTDOWN, 0);
   }

void MAX7219_ShutdownStop (void)

/* Bringt das Display aus dem Shutdown Mode */
   {
   MAX7219_Write (REG_SHUTDOWN, 1);
   }

void MAX7219_DisplayTestStart (void)

/* Schaltet alle Segmente ein - Test Mode */
   {
   MAX7219_Write (REG_DISPLAY_TEST, 1);
   }
```

```
void MAX7219_DisplayTestStop (void)

/* Bringt das Display aus dem Test Mode */
   {
   MAX7219_Write (REG_DISPLAY_TEST, 0);
   }

void MAX7219_SetBrightness (char brightness)

/* Setzt die Helligkeit des Displays*/
   {
   brightness &= 0x0f;
   MAX7219_Write (REG_INTENSITY, brightness);
   }

void MAX7219_Clear (void)

/* Loescht das Display*/
   {
   char k;
   for (k=1; k<=5; k++)
      {
      MAX7219_Write (k, 0x00);
      }
   }

void MAX7219_Write (unsigned char reg_number, unsigned char dataout)

/* Hier werden die Daten in die Register der Displays geschrieben - MSB (15) zuerst */
   {
   char i,g;
   LOAD_0();

   // Datenbits D15-D12 werden beliebig (z.b.: '0') uebertragen:
   for (i=0;i<4;i++)
      {
      DIN_0();
      CLK_1();
      CLK_0();
      }
   // Datenbits D11-D8 (Registeradresse) werden uebertragen:
   for (i=4;i>0;i--)
      {
      unsigned char mask = 1 << (i-1);
      if (reg_number & mask)
         {
         DIN_1();
         }
      else
         {
         DIN_0();
         }
      CLK_1();
      CLK_0();
      }
   // Datenbits D7-D0 (Daten) werden uebertragen:
   for (i=8; i>0; i--)
      {
      unsigned char mask = 1 << (i-1);
      if (dataout & mask)
         {
         DIN_1();
         }
      else
         {
         DIN_0();
         }
```

```
      CLK_1();
      CLK_0();
      }
   // No-op Register beschreiben
   if (z>>0)
      {
      for (g=0;g<z;g++)
         {
         for (i=0;i<4;i++ //beliebig (z.b.: '0') uebertragen
             )
            {
            DIN_0();
            CLK_1();
            CLK_0();
            }
         for (i=4;i>0;i--  //In NO-OP Register 0x00 schreiben
             )
            {
            DIN_0();

            CLK_1();
            CLK_0();
            }
         for (i=8;i>0;i-- //beliebig (z.b.: '0') uebertragen
             )
            {
            DIN 0();
            CLK_1();
            CLK_0();
            }
         }
      }
   LOAD_1();
   }

void ziffern (void)
   {
   if (v==1)
      {
      /*Hier findet die Auswertung der Timerwerte die im Feld t1
        gespeichert wurden statt. Jede Ziffer muss aus der Zahl
        mithilfe von Rechenoperationen ermittelt werden */

      unsigned int aa, xx, b1,d1,f1,g1;
      unsigned long f,ziffer;
      unsigned char fehl;

      fehl = 0; //Fehler Zehlvariable 0 setzen
      for (aa=1;aa<=9;aa++)
         {
         if (t1[aa] == 0)
            {
            fehl++;
            }
         }
      if (fehl == 0 //es sind alle beschrieben
         )
         {
         for (aa=1;aa<10;aa++)
            {
            z=aa-1;
            xx=5;

            f=t1[aa];
            ziffer= f/10000;
            MAX7219_Write (xx,(unsigned char)ziffer);

            b1=f%10000;
```

```
            ziffer = b1/1000;
            xx--;
            MAX7219_Write  (xx,(unsigned char)ziffer);

            d1=b1%1000;
            ziffer=d1/100;
            xx--;
            MAX7219_Write  (xx,(unsigned char)ziffer);

            f1=d1%100;
            ziffer=f1/10;
            xx--;
            MAX7219_Write  (xx,(unsigned char)ziffer);

            g1=f1%10;
            ziffer=g1;
            xx--;
            MAX7219_Write  (xx,(unsigned char)ziffer);
            MAX7219_SetBrightness(INTENSITY_MAX);
            }
         /* EasyCASE ( 19
            Serielle Ausgabe */
         SBUF = 115;  //'s' als Startzeichen fuer VB6-Programm
         while (!TI)
            {
            ; //warten auf das Senden eines Zeichens
            }
         TI=0;
         for (aa=1;aa<10;aa++)
            {
            z=aa-1;
            xx=5;

            f=t1[aa];
            ziffer= f/10000;
            SBUF = (unsigned char)ziffer+48;
            while (!TI)
               {
               ; //warten auf das Senden eines Zeichens
               }
            TI=0;

            b1=f%10000;
            ziffer = b1/1000;
            xx--;
            SBUF = (unsigned char)ziffer+48;   //zurueckschicken
            while (!TI)
               {
               ; //warten auf das Senden eines Zeichens
               }
            TI=0;

            d1=b1%1000;
            ziffer=d1/100;
            xx--;
            SBUF = (unsigned char)ziffer+48;   //zurueckschicken
            while (!TI)
               {
               ; //warten auf das Senden eines Zeichens
               }
            TI=0;

            f1=d1%100;
            ziffer=f1/10;
            xx--;
            SBUF = (unsigned char)ziffer+48;   //zurueckschicken
            while (!TI)
               {
```

```
               ; //warten auf das Senden eines Zeichens
                 }
              TI=0;

              g1=f1%10;
              ziffer=g1;
              xx--;
              SBUF = (unsigned char)ziffer+48;    //zurueckschicken
              while (!TI)
                 {
                 ; //warten auf das Senden eines Zeichens
                 }
              TI=0;

              SBUF = 32;  //Leerzeichen senden
              while (!TI)
                 {
                 ; //warten auf das Senden eines Zeichens
                 }
              TI=0;

              SBUF = 44;   //Kommazeichen senden
              while (!TI)
                 {
                 ; //warten auf das Senden eines Zeichens
                 }
              TI=0;

              SBUF = 32;  //Leerzeichen senden
              while (!TI)
                 {
                 ; //warten auf das Senden eines Zeichens
                 }
              TI=0;
              }
           SBUF = 101;  //'e' als Endezeichen fuer VB6-Programm
           while (!TI)
              {
              ; //warten auf das Senden eines Zeichens
              }
           TI=0;

           }
        else
           {
           for (z=0;z<=8;z++)
              {
              MAX7219 DisplayTestStart ();
              /* MAX7219 DisplayTestStart
              soll Sensorfehler anzeigen */
              }
           }
        v=0;
        }
     }
```