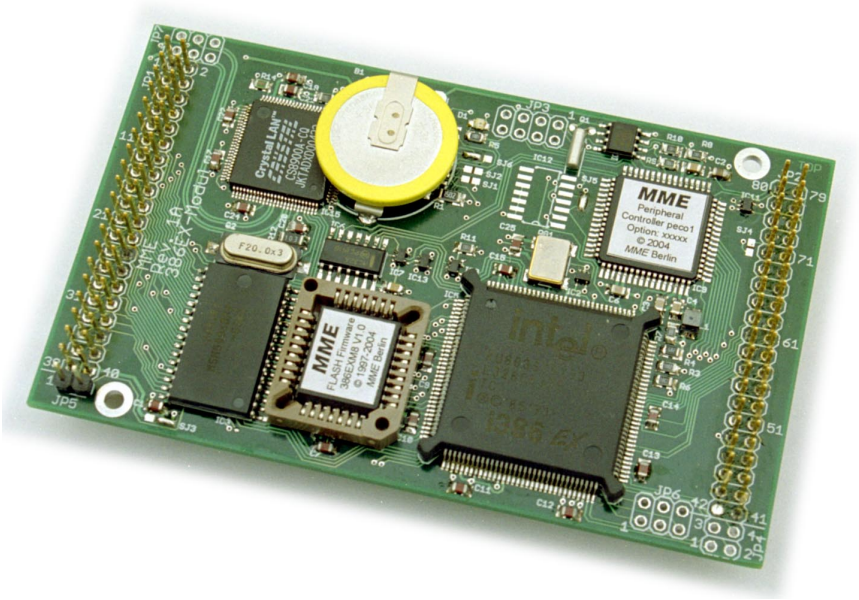
386EX-Modul

Dokumentation: 386exm8, Revision 0.9



MME Müller Mikroelektronik
Hauptstraße 8, Gewerbehaus II, 10827 Berlin (Schöneberg)
Tel.: +49-30-787.09.450, FAX: +49-30-787.09.451
E-Mail: info@mme-berlin.de, Internet: http://www.mme-berlin.de

Einführung

Das 386EX-Modul ist ein vollständiger Einplatinen-Rechner (Single Board Computer) für den universellen Einsatz in der Meß-, Steuerungs- und Regeltechnik. Durch seinen kompakten und robusten Aufbau ist es besonders für den Einsatz in Embedded Systemen geeignet. Herz des Moduls ist der 32-Bit-Microcontroller 386EX von Intel.

Der 386EX Microcontroller gehört zur Familie der X86 Prozessoren, wodurch eine breite Palette von bewährten Compilern und Tools für die Softwareentwicklung zur Verfügung steht. Das 386EX-Modul kann zur Zeit relozierte Turbo Pascal- (5, 5.5, 6 und 7) sowie Delphi 1- Programme direkt im FLASH-Speicher ausführen. Hierfür ist die bewährte Locator- Software „Pasloc" (für DOS) und „Delphi1Loc" (für Windows) weiterentwickelt worden. Die Borland Unit „Graph" sowie die BGI-Treiber können verwendet werden um komplexe Anwendungen mit graphischem Display zu realisieren. Ein Betriebssystem ist hierfür nicht erforderlich, wodurch u. a. extrem kurze „Boot-Zeiten“ von ca. 12 ms erreicht werden.

Die Verwendung des Open Watcom C Compilers 1.4 für die Ausführung von Anwendungen (direkt im FLASH-Speicher) ist ebenfalls möglich. Bei besonders kritischen Anwendungen (Militär, Raumfahrt, Medizintechnik usw.) kann der Verzicht auf ein Betriebssystem Sinn machen, um dem Anwender die volle Kontrolle über das System zu ermöglichen. Speziell bei der Verarbeitung von Interrupts in Echtzeit ist der Zustand des Systems (wann sind Interrupts gesperrt, wie groß ist der Heap/Stack usw.) von besonderer Wichtigkeit. Ein Embedded DOS mit FAT Filesystem ist in Vorbereitung.

Das 386EX-Modul ist auch als Ersatz für das seit über 10 Jahren verfügbare V25-Modul konzipiert worden, dessen Microcontroller (NEC V25) nicht mehr hergestellt wird. Die mechanischen Abmessungen sowie die Plazierung der beiden 40-poligen Stiftleisten sind identisch zum V25-Modul. Die wesentlichen Signale auf den Stiftleisten sind ebenso übernommen worden, so daß gegebenenfalls ein „Drop-In Replacement“ möglich ist.

Das Modul kann optional mit leistungsfähigen Hardwarekomponenten aufgerüstet werden, so daß auch komplexere Anwendungen mit erhöhten Anforderungen realisiert werden können. Die Funktionalität des 386EX-Moduls geht weit über die Leistungsmerkmale des V25-Moduls hinaus. Um trotzdem die Basisfunktionen kompatibel zu halten, sind die neuen Erweiterungen, wie z. B. Ethernet, I2C-Bus usw. auf separate Stiftleisten plaziert worden.

Das 386EX-Modul ist für den Einsatz unter extremen Bedingen entwickelt worden. Eine Variante für den erweiterten Temperaturbereich (-40 °C bis +85 °C) steht zur Verfügung. Das Modul wird auch RoHS-konform angeboten.

Leistungsmerkmale

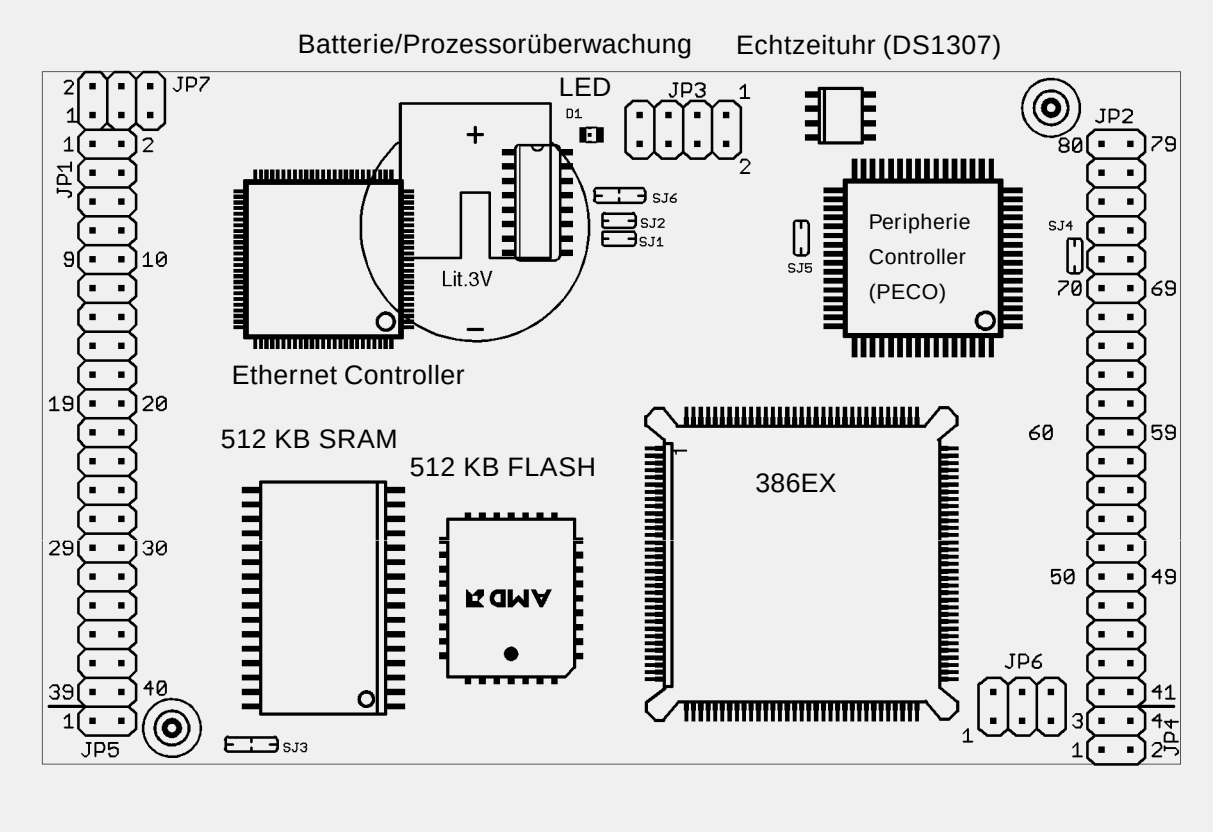
Standard Leistungsmerkmale:

- Abmessungen: Breite = 61 mm, Länge = 100 mm und Höhe = 20 mm (mit Aufsteckleiste).
- 32-Bit-Microcontroller Intel 386EX. Oszillatorfrequenz: 22,11840 MHz oder 66 MHz.
- Speicherbestückung: 512 KB FLASH und 512 KB SRAM. FLASH und SRAM sind extern auf 16 MB erweiterbar.
- 1024 Byte EEPROM
- Drei 16-Bit Timer, kompatibel zu Intel 8254.
- Zwei Interrupt Controller, kompatibel zu Intel 8259.
- Zwei serielle Schnittstellen, kompatibel zu Intel 8250, maximal 345600 Baud @22,11840 MHz.
- Eine serielle Schnittstelle über Peripherie Controller.
- 32 digitale Ein- und Ausgänge. Alle 32 Ein- und Ausgänge bleiben bei Verwendung der 8250-kompatiblen seriellen Schnittstellen erhalten.
- 8 Kanal Analog-/Digitalwandler, bis zu 10 Bit Auflösung.
- Temperaturbereich: 0 °C bis 70 °C (Standard) und -40 °C bis +85 °C (erweitert).
- Versorgungsspannung: 5 V DC, stabilisiert.
- Stromaufnahme: ca. 120 mA @22,11840 MHz.
- System „Boot-Zeit“: ca. 12 ms (ohne Bestriebssystem, Anwendung läuft im FLASH).
- “Drop-In Replacement“ für *MME* V25-Modul.
- Ausführung RoHS-konform.
- Kein Betriebssystem erforderlich.
- Anwendungen können direkt im FLASH ausgeführt werden. Unterstützte Compiler: Turbo Pascal 5.0, 5.5, 6, 7, Delphi 1 sowie Open Watcom C 1.4

„Onboard“ Erweiterungsmöglichkeiten:

- Echtzeituhr mit Batteriebackup.
- Batteriebackup für SRAM, Schutz vor Datenkorruption durch „CE-Gating“.
- I^2C-Bus, max. 400 kHz.
- 10BASE-T Ethernet

Funktionsübersicht



Hardwarekomponenten des 386EX-Moduls

Das 386EX-Modul besteht im wesentlichen aus den typischen Hardwarekomponenten eines Einplatinenrechners: Prozessor, SRAM, FLASH und Prozessorüberwachungslogik (Reset, Watchdog). Die Batteriepufferung des 512 KB SRAMs ist möglich. Um Datenkorruption während der Powerup- und Powerdown-Phase zu vermeiden kann das CE#-Signal kontrolliert („gegated“) werden.

Eine Batterie gepufferte Echtzeituhr vom Typ DS1307 ist integriert. Die Uhr zeichnet sich durch geringe Stromaufnahme im „Batteriebackup-Modus“ (<= 500 µA) aus. Zusätzlich können Interrupts mit programmierbaren Intervallzeiten am Prozessor ausgelöst werden.

Das Modul stellt den Adress- und Datenbus des Prozessors an den Stiftleisten JP1 und JP2 zur Verfügung, so daß externe Komponenten (Graphik-LCD, CompactFlash usw.) optimal angebunden werden können. Wichtiger Bestandteil des 386EX-Moduls ist der sogenannte Peripherie Controller PECO. Dieser kommuniziert mit dem I386EX Controller und stellt zusätzliche Peripheriefunktionen bereit: 1024 Byte EEPROM, A/D-Wandler, PWM, I2C- und SPI-Bus. Ein Ethernet Controller vom Typ CS8900A ermöglicht die Implementierung von 10BASE-T Netzwerkanwendungen.

Inbetriebnahme (Modul ohne Betriebssystem)

Nach dem Anlegen der Versorgungsspannung von 5 V befindet sich das Modul im sogenannten Download-Modus. Dies wird durch die blinkende LED (Blinkfrequenz ca. 1 Hz) auf dem Modul signalisiert. Der Download-Modus stellt sich immer dann ein, wenn keine ausführbare Anwendung im Modul gespeichert ist. Dies ist bei der Auslieferung der Fall.

Eine ausführbare Anwendung wird mit dem mitgelieferten Download-Programm *CMDLoad32* über die serielle Schnittstelle 1 in den FLASH-Speicher des Moduls geladen. Der Aufruf lautet z. B.:

```
CMDLoad32 hallo.fls /P
```

Nach dem Download wird die Anwendung im Modul gestartet. Die Leuchtdiode ist abgeschaltet. Deren Steuerung kann gegebenenfalls von der Anwendung übernommen werden. Die Datenübertragung wird mit Checksummen gesichert. Bei fehlerhafter oder nicht vollständiger (abgebrochener) Übertragung bleibt das Modul im Download-Modus.

Um das Modul bei geladener Anwendung erneut in den Download-Modus zu versetzen, ist die Port-Leitung *PT0/ADC0* beim Power-On (oder RESET) auf Masse zu legen. Die Anwendung kann das Modul aber auch eigenständig in den Download-Modus versetzen. Hierfür stehen entsprechende C- und Pascal-Funktionen zur Verfügung.

Funktion

Belegung der Stiftleisten:

An den Stiftleisten JP1 und JP2 sind alle wichtigen Signale herausgeführt, welche zur Anbindung von externen Komponenten erforderlich sind. Dies sind im wesentlichen:

- 16-Bit Adress-, 8-Bit Daten- und Steuerbus des Prozessors.
- Zwei serielle Schnittstellen
- 32 digitale Ein- und Ausgänge.
- 8 Kanal A/D-Wandler mit 10 Bit Auflösung
- 6 Interrupt Eingänge
- 2 PWM Ausgänge

Pin-Nr.	Bezeichnung	Pin-Nr.	Funktion
01	GND	02	GND
03	P1.0/DCD0#	04	P1.1/RTS0#
05	P1.2/DTR0#	06	P1.3/DSR0#
07	P1.4/RIO#	08	P1.5/LOCK#
09	P1.6/HOLD	10	P1.7/HLDA
11	D0	12	D1
13	D2	14	D3
15	D4	16	D5
17	D6	18	D7
19	A0	20	A1
21	A2	22	A3
23	A4	24	A5
25	A6	26	A7
27	A8	28	A9
29	A10	30	A11
31	A12	32	A13
33	A14	34	A15
35	A16	36	A17
37	A18	38	A19
39	VBATT	40	VOUT

Stiftleiste JP1

Funktion

Pin-Nr.	Bezeichnung	Pin-Nr.	Funktion
41	+5 V (VCC)	42	+5 V (VCC)
43	RXD0	44	CTS0
45	TXD0	46	RXD1
47	CTS1	48	TXD1
49	P20/RXD2	50	P21/TXD2
51	P22/AIN0	52	P23/AIN1
53	P24/PWM0	54	P25/PWM1
55	P26	56	P27
57	P3.0/INT9/TMRO0	58	P3.1/INT8/TMRO1
59	P3.2/INT0	60	P3.3/INT1
61	P3.4/INT2	62	P3.5/INT3
63	P3.6/PWRDOWN	64	P3.7/COMCLK
65	PT0/ADC0	66	PT1/ADC1
67	PT2/ADC2	68	PT3/ADC3
69	PT4/ADC4	70	PT5/ADC5
71	PT6/ADC6	72	PT7/ADC7
73	VTH	74	GND
75	RESET#	76	P2.0/CS0#
77	R/W#	78	P2.1/CS1#
79	IOSTB#	80	SCK

Stiftleiste JP2

Pin-Nr.	Bezeichnung	Pin-Nr.	Funktion
01	MISO	02	MOSI
03	SCL	04	AUX0
05	SDA	06	AUX1
07	+5 V (VCC)	08	GND

Stiftleiste JP3

JP3 führt Steuersignale für die synchrone serielle Schnittstelle (SPI), den I2C-Bus sowie zwei zusätzliche freiprogrammierbare I/O-Leitungen.

Funktion

Pin-Nr.	Bezeichnung	Pin-Nr.	Funktion
01	A20	02	A21
03	A22	04	A23

Stiftleiste JP4

Stiftleiste JP4 stellt 4 zusätzliche Adressleitungen zur Verfügung, womit sich der Speicher des Moduls auf insgesamt 16 MB aufrüsten läßt.

Pin-Nr.	Bezeichnung	Pin-Nr.	Funktion
01	RD#	02	WR#

Stiftleiste JP5

Pin-Nr.	Bezeichnung	Pin-Nr.	Funktion
01	RESET#	02	TDI
03	TDO	04	TCK
05	TMS	06	GND

Stiftleiste JP6

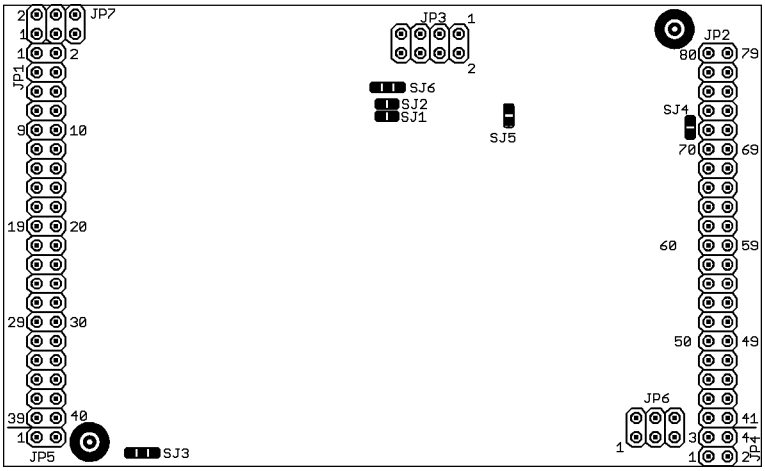
Pin-Nr.	Bezeichnung	Pin-Nr.	Funktion
01	LANLED#	02	LINKLED#
03	RXD-	04	TXD-
05	RXD+	06	TXD+

Stiftleiste JP7

Die Steuerbussignale RD# und WR# liegen auf der Stiftleiste JP5. JP6 stellt die JTAG-Schnittstelle zur Verfügung und über JP7 wird das Ethernet-Interface kontaktiert.

Funktion

Funktion der Lötbrücken (Solder Jumper):



Lötbrücke SJ1

Brücke geschlossen: Der Powerfail-Ausgang des externen Prozessorüberwachungsbausteins MAX691A ist mit dem NMI-Eingang des 386EX-Controllers verbunden. Bei einem Powerfail-Zustand wird somit ein nicht maskierbarer Interrupt am 386EX ausgelöst.

Brücke offen (default): Powerfail-Ausgang und NMI-Eingang sind getrennt.

Lötbrücke SJ2

Brücke geschlossen: Der externe Watchdog des Prozessorüberwachungsbausteins ist aktiv. Der Watchdog-Eingang muß regelmäßig (mindestens alle 1,6 Sekunden) „getoggled“ werden. Andernfalls wird ein RESET am 386EX-Controller ausgelöst.

Brücke offen (default): Der externe Watchdog ist inaktiv.

Lötbrücke SJ3

Position 

In dieser Position wird die Batteriepufferung des 512 KB SRAMs ermöglicht. Das 386EX-Modul muß hierfür mit einer Batterie bestückt sein oder die Einspeisung der Pufferspannung muß extern über den VBATT-Eingang (JP1, Pin 39) vorgenommen werden. Bei der externen Einspeisung darf **keine** Batterie auf dem Modul bestückt sein.

Position  (default)

Das 512 KB SRAM wird nicht Batterie gepuffert betrieben.

Funktion

Lötbrücke SJ4

Brücke offen (default): Die Referenzspannung für den Analog-/Digitalwandler beträgt +5 V. Die Spannung wird auf dem Modul generiert.

Brücke geschlossen: Die Referenzspannung für den Analog-/Digitalwandler wird extern über den VTH-Eingang (JP2, Pin 73) eingespeist.

Lötbrücke SJ5

Die Einstellung der Lötbrücke SJ5 darf nicht verändert werden. Sie ist abhängig von der Taktfrequenz des Moduls.

Brücke offen: Die Taktfrequenz beträgt 22,118 MHz.

Brücke geschlossen: Die Taktfrequenz beträgt 66 MHz.

Lötbrücke SJ6

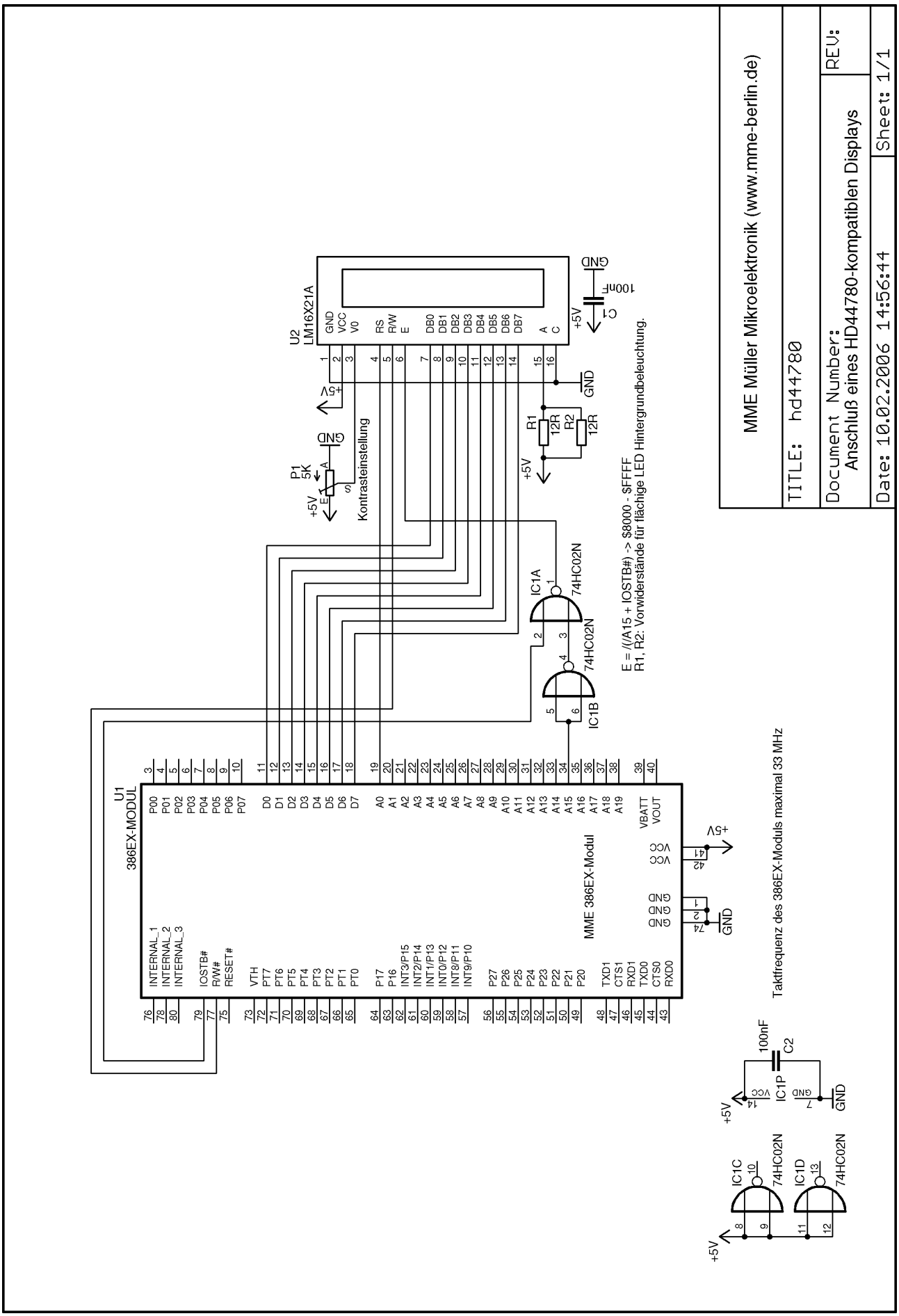
Position  (default)

Das CS#-Signal für das SRAM wird nicht kontrolliert. Diese Einstellung ist sinnvoll, wenn das SRAM nicht Batterie gepuffert betrieben wird. Das CS#-Signal wird in dieser Einstellung nicht verzögert.

Position 

Diese Einstellung ist sinnvoll, wenn das SRAM Batterie gepuffert betrieben wird. Während der Powerup- und Powerdown-Phase des Moduls wird das CS#-Signal des SRAMs inaktiv gehalten, so daß Datenkorruption vermieden wird. In diesem Modus wird das CS#-Signal um ca. 6 ns verzögert. Eventuell muß die Verzögerung bei der Einstellung der Waitstates berücksichtigt werden.

Applikation: Anschluß eines HD44780-kompatiblen Displays



Applikation: Anschluß eines HD44780-kompatiblen Displays

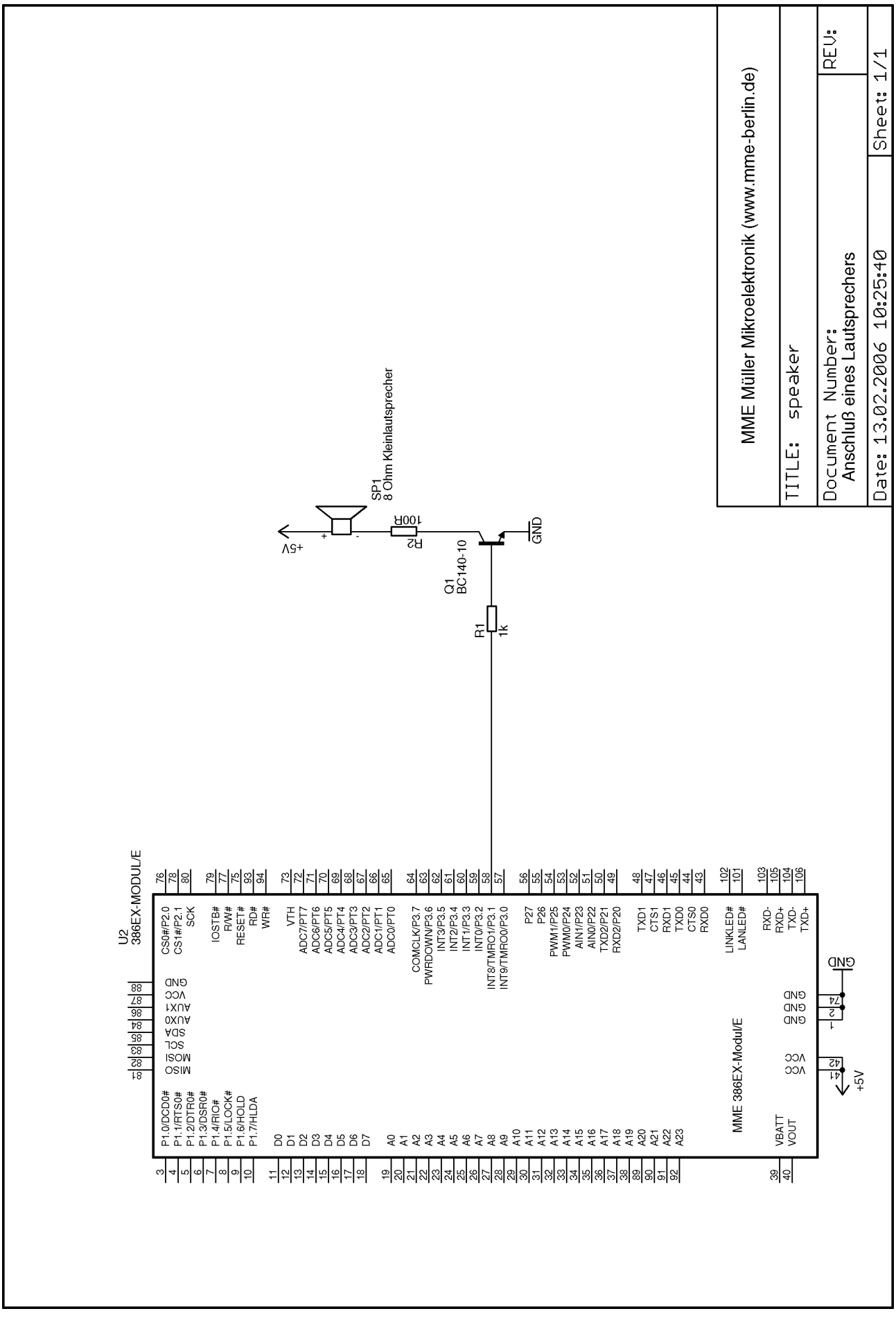
Programmbeispiel in Pascal/Delphi 1:

```
{
   Hallo.pas (C) 2006 MME Müller Mikroelektronik

   Dieses Programm schreibt "Hallo Welt" in das LCD-Display.
}
{$N-,E-}
{$M 4096,0,655360}
Program Hallo;
uses I386M8, HD44780;

begin
   HD_Write('Hallo Welt!');
   repeat
   until FALSE;
end.
```

Programmbeispiel in C:

```
/*
   Hallo.c (C) 2006 MME Müller Mikroelektronik

   Dieses Programm schreibt "Hallo Welt" in das LCD-Display.
*/

#include <stdio.h>
#include "pastypes.h"
#include "hd44780.h"

void main(void)
{
   HD_Init();
   HD_Write("Hallo Welt!");
   do {
   } while(TRUE);
}
```

Applikation: Anschluß eines Lautsprechers



Applikation: Anschluß eines Lautsprechers

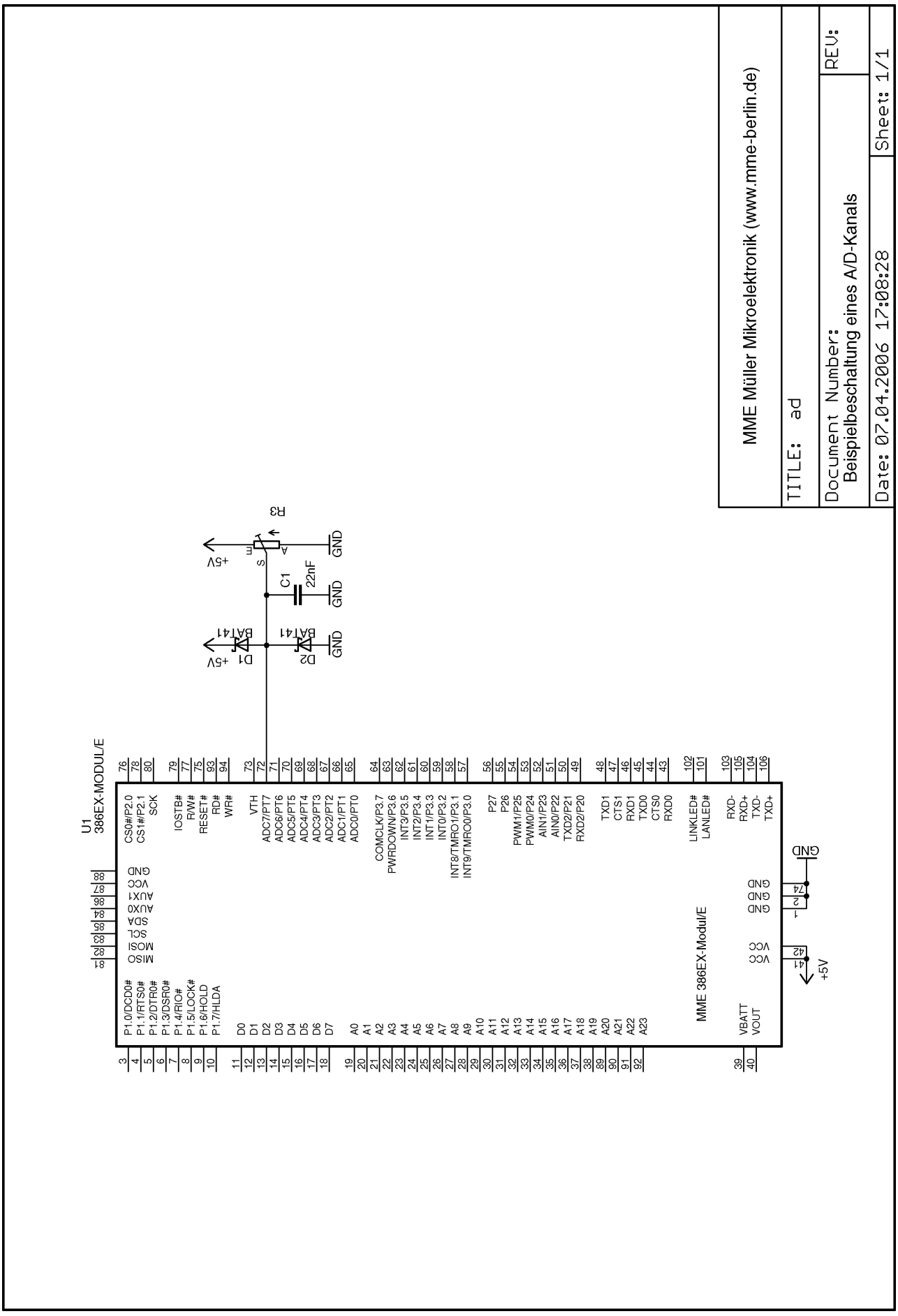
Programmbeispiel in Pascal/Delphi 1:

```
{
   Speaker.pas (C) 2006 MME Müller Mikroelektronik

   Dieses Programm demonstriert die Tonausgabe über einen Lautsprecher. Der
   Lautsprecher ist an P3.1/TMRO1 angeschlossen.
}
{$N-,E-}
{$M 4096,0,655360}
Program Speaker;
uses I386M8;

begin
   repeat
      I386_Sound(1000);                 { Tonausgabe mit 1 kHz erzeugen    }
      I386_DelayWithoutTimer(1000);     { Eine Sekunden verweilen          }
      I386_NoSound;                     { Ton abschalten                   }
      I386_DelayWithoutTimer(500);      { 500 ms verweilen                 }
   until FALSE;
end.
```

Programmbeispiel in C:

```
/*
   Speaker.c (C) 2006 MME Müller Mikroelektronik

   Dieses Programm demonstriert die Tonausgabe über einen Lautsprecher. Der
   Lautsprecher ist an P3.1/TMRO1 angeschlossen.
*/
#include <stdio.h>
#include "pastypes.h"
#include "i386m8.h"

void main(void)
{
   do {
      I386_Sound(1000);                 // Tonausgabe mit 1 kHz erzeugen
      I386_DelayWithoutTimer(1000);     // Eine Sekunden verweilen
      I386_NoSound();                   // Ton abschalten
      I386_DelayWithoutTimer(500);      // 500 ms verweilen
   } while(TRUE);
}
```

Applikation: Beispielbeschaltung des A/D-Wandlers



Applikation: Beispielbeschaltung des A/D-Wandlers

Programmbeispiel in Pascal/Delphi 1:

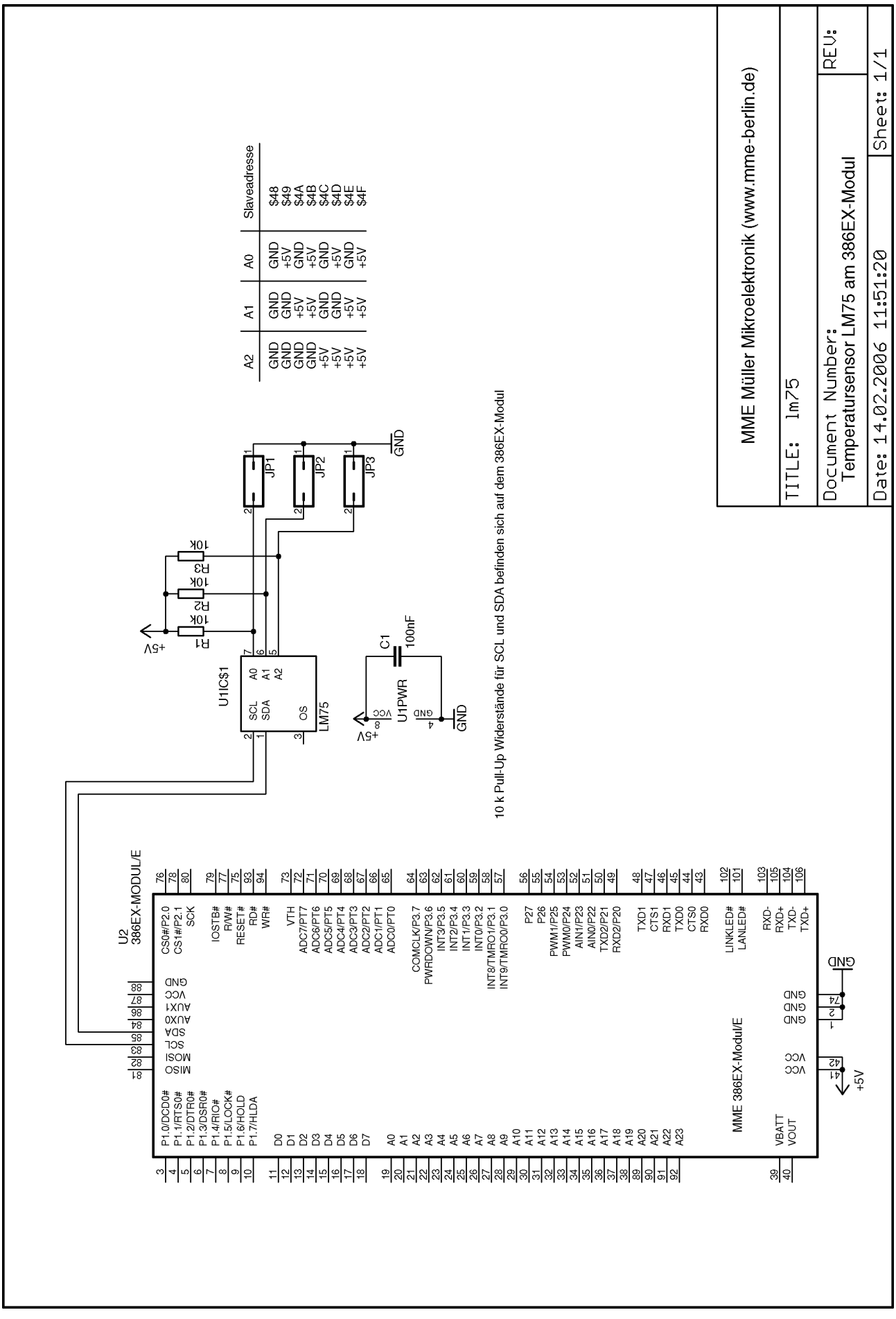
```
{
   AD.pas (C) 2006 MME Müller Mikroelektronik

   Dieses Programm misst die Spannung am Eingang ADC7.
}
{$N-,E-}
{$M 4096,0,655360}
Program Ad;
uses I386M8, HD44780;

var
   rVoltage: Real;
   bADValue: Byte;

function RealToStr(rValue: Real; bWith, bDecimals: Byte): String;
{
   Wandelt eine Real-Variable in einen String.
}
var
   s: String;
begin
   str(rValue:bWith:bDecimals, s);
   RealToStr:= s;
end;

begin
   {
      AD-Wandler initialisieren: Referenzspannung = AVCC, Kanal = 7
      Pullupwiderstand von ADC7 entfernen.
   }
   I386_AD_Init(adrsAVCC, 7, $80);
   HD_Write('Kanal 7:');
   repeat
      HD_GotoXY(10, 1);
      bADValue:= I386_AD_GetValue8BitFast;          { Wert einlesen und auf...   }
      rVoltage:= (5 * bADValue) / 255;              { ...5 V skalieren           }
      HD_Write(RealToStr(rVoltage, 2, 2) + ' V');
   until FALsE;
end.
```

Applikation: Beispielbeschaltung des A/D-Wandlers

Programmbeispiel in C:

```
/*
   AD.c (C) 2006 MME Müller Mikroelektronik

   Dieses Programm misst die Spannung am Eingang ADC7.
*/
#include <stdio.h>
#include <string.h>
#include "pastypes.h"
#include "i386m8.h"
#include "hd44780.h"

void main(void)
{
   float flVoltage;
   Byte bADValue;
   char s[16];

   HD_Init();
   /*
       AD-Wandler initialisieren: Referenzspannung = AVCC, Kanal = 7
       Pullupwiderstand von ADC7 entfernen.
   */
   I386_AD_Init(adrsAVCC, 7, 0x80);
   HD_Write("Kanal 7:");
   do {
       HD_GotoXY(10, 1);
       bADValue= I386_AD_GetValue8BitFast();    /* Wert einlesen und auf...   */
       flVoltage= (5 * bADValue) / 255;         /* ...5 V skalieren           */
       sprintf(s, "%.2f V", flVoltage);
       HD_Write(s);
   } while(TRUE);
}
```

Applikation: Anschluß eines LM75 Temperatursensors am I2C-Bus



Applikation: Anschluß eines LM75 Temperatursensors am I2C-Bus

Programmbeispiel in Pascal/Delphi 1:

```
{
   Temp.pas (C) 2006 MME Müller Mikroelektronik

   Dieses Programm zeigt die Temperatur an, welche von einem LM75 Temperatur-
   sensor geliefert wird. Der Sensor ist am I2C-Bus angeschlossen und hat die
   Slaveadresse $48.
}
{$N-,E-}
{$M 4096,0,655360}
Program Temp;
uses I386M8, HD44780, LM75;
const
   LM75_ADDRESS = $48;
var
   rTemperature: Real;

function RealToStr(rValue: Real; bWith, bDecimals: Byte): String;
{ Wandelt eine Real-Variable in einen String. }
var
   s: String;
begin
   str(rValue:bWith:bDecimals, s);
   RealToStr:= s;
end;

begin
   if LM75_IsSensorPresent(LM75_ADDRESS) = TRUE then begin
       { Sensor ist da. Also Temperatur einlesen }
       HD_Write('Temperatur:');
       repeat
          LM75_GetTemperatureAsReal(LM75_ADDRESS, rTemperature);
          HD_GotoXY(1, 2);
          HD_Write(RealToStr(rTemperature, 2, 1) + ' Grad Cels.');
          I386_DelayWithoutTimer(300);         { Nächster Zugriff erst in 300 ms }
       until FALSE;
   end else begin
       { Kein Sensor angeschlossen }
       HD_Write('Kein Sensor!');
       repeat
       until FALSE;
   end;
end.
```

Applikation: Anschluß eines LM75 Temperatursensors am I2C-Bus

Programmbeispiel in C:

```
/*
   Temp.c (C) 2006 MME Müller Mikroelektronik

   Dieses Programm zeigt die Temperatur an, welche von einem LM75 Temperatur-
   sensor geliefert wird. Der Sensor ist am I2C-Bus angeschlossen und hat die
   Slaveadresse $48.

*/
#include <stdio.h>
#include <string.h>
#include "pastypes.h"
#include "i386m8.h"
#include "hd44780.h"
#include "lm75.h"

#define LM75_ADDRESS 0x48

void main(void)
{
   float flTemperature;
   char s[16];

   HD_Init();
   if (LM75_IsSensorPresent(LM75_ADDRESS) == TRUE) {
      /* Sensor ist da. Also Temperatur einlesen */
      HD_Write("Temperatur:");
      do {
         LM75_GetTemperatureAsReal(LM75_ADDRESS, &flTemperature);
         HD_GotoXY(1, 2);
         sprintf(s, "%.1f", flTemperature);
         strcat(s, " Grad Cels.");
         HD_Write(s);
         I386_DelayWithoutTimer(300);         // Nächster Zugriff erst in 300 ms
      } while(TRUE);
   } else {
      /* Kein Sensor angeschlossen */
      HD_Write("Kein Sensor!");
      do {
      } while (TRUE);
   }
}
```

Programmbeispiel: Umgang mit Batterie gepufferten Variablen

Programmbeispiel in Pascal/Delphi 1:

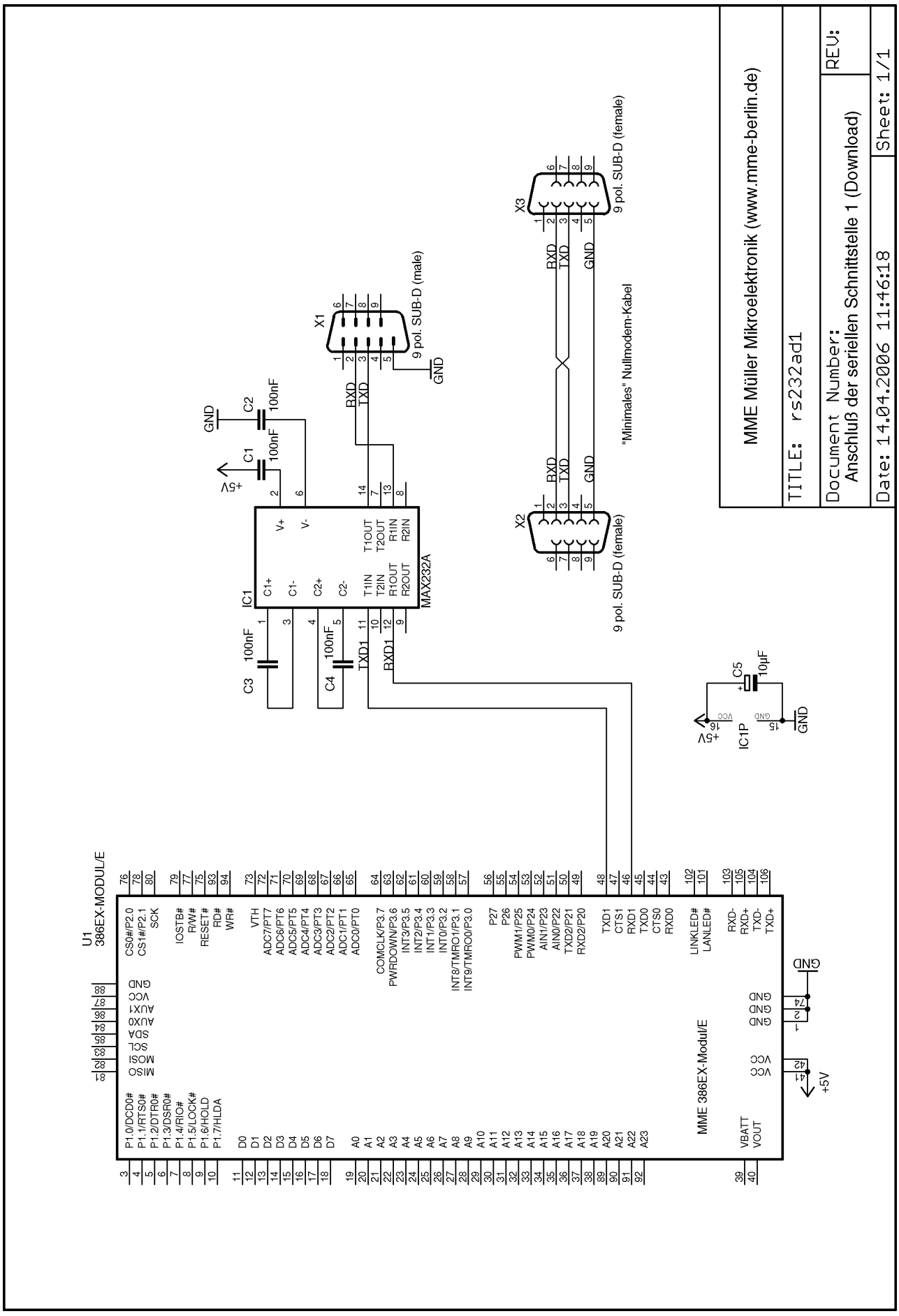
```
{
   BattVar.pas (C) 2006 MME Müller Mikroelektronik

   Bei Anwendungen mit Batterie gepufferten Variablen ist zu unterscheiden, ob
   eine Anwendung erstmalig in Betrieb genommen wird (zufällige Variablenwerte),
   oder ob schon gültige Werte vorliegen.

   Hinweis: Typisierte Konstanten werden nach jedem Reset initialisiert. Ihr
   Inhalt kann also nicht Batterie gepuffert werden! Bei Verwendung von
   Turbo/Borland Pascal 7.0/Delphi 1 muß mit dem /V-Parameter reloziert werden,
   weil sonst alle globalen Variablen vom Runtime-System auf 0 gesetzt werden.
}
{$N-,E-}
{$M 4096,0,655360}
Program Battvar;
uses I386M8, HD44780;
var
   {
      Die Variable "lSignatur" enthält einen bestimmten Wert, wenn die zu
      puffernden Variablen gültig sind.
   }
   lSignature: Longint;
   bBattVar: Byte;   { Dies ist die zu puffernde Variable }
   s: String;
begin
   if lSignature <> $12345678 then begin
      {
         Die Signatur stimmt nicht, also ist dies das erste Mal. Die
         Signatur muß also gesetzt und die zu puffernde Variable
         initialisiert werden.
      }
      lSignature:= $12345678;
      bBattVar:= 0;
      HD_Write('Das erste Mal!');
   end else begin
      {
         Die Signatur ist ok. Die zu puffernde Variable ist gültig.
      }
      HD_Write('Variablen g' + #245 + 'ltig');
      inc(bBattVar);
   end;
   {
      Zur Kontrolle wird die zu puffernde Variable aus gegeben. Beim
      ersten Mal ist sie 0, danach wird sie nach jedem Einschalten inkre-
      mentiert.
   }
   HD_GotoXY(1, 2);
   HD_Write('BattVar:');
   str(bBattVar:2, s);
   HD_Write(s);
   repeat
   until FALSE;
end.
```

Programmbeispiel: Umgang mit Batterie gepufferten Variablen

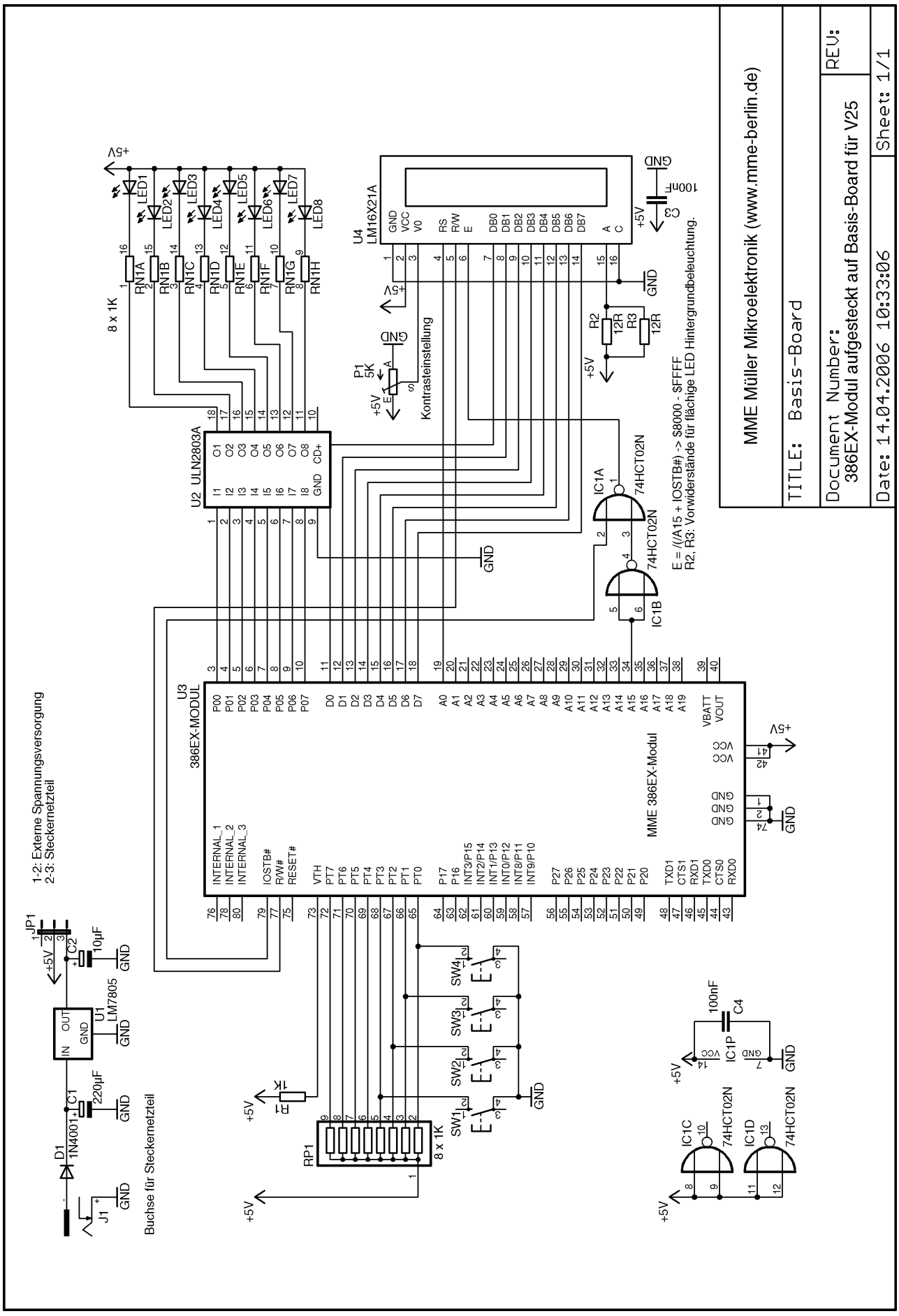
Programmbeispiel in C:

```
/*
   battvar.c (C) 2006 MME Müller Mikroelektronik

   Bei Anwendungen mit Batterie gepufferten Variablen ist zu unterscheiden, ob
   eine Anwendung erstmalig in Betrieb genommen wird (zufällige Variablenwerte),
   oder ob schon gültige Werte vorliegen.

   Hinweis: Bei Verwendung von Open Watcom 1.4 muß bei der Flashkonvertierung
   der /V-Parameter angegeben werden. Andernfalls würden alle globalen Variablen
   vom Runtime-System auf 0 gesetzt werden.
*/

#include <stdio.h>
#include "pastypes.h"
#include "hd44780.h"
   /*
      Die Variable "lSignatur" enthält einen bestimmten Wert, wenn die zu
      puffernden Variablen gültig sind.
   */
   Longint lSignature;
   Byte bBattVar;   /* Dies ist die zu puffernde Variable */

void main(void)
{
   char s[16];

   HD_Init();
   if (lSignature != 0x12345678) {
      /*
         Die Signatur stimmt nicht, also ist dies das erste Mal. Die
         Signatur muß also gesetzt und die zu puffernde Variable
         initialisiert werden.
      */
      lSignature= 0x12345678;
      bBattVar= 0;
      HD_Write("Das erste Mal!");
   } else {
      /*
         Die Signatur ist ok. Die zu puffernde Variable ist gültig.
      */
      HD_Write("Variablen g\245ltig");
      bBattVar++;
   }
   /*
      Zur Kontrolle wird die zu puffernde Variable aus gegeben. Beim
      ersten Mal ist sie 0, danach wird sie nach jedem Einschalten inkre-
      mentiert.
   */
   HD_GotoXY(1, 2);
   HD_Write("BattVar: ");
   sprintf(s, "%d", bBattVar);
   HD_Write(s);
   do {
   } while(TRUE);
}
```

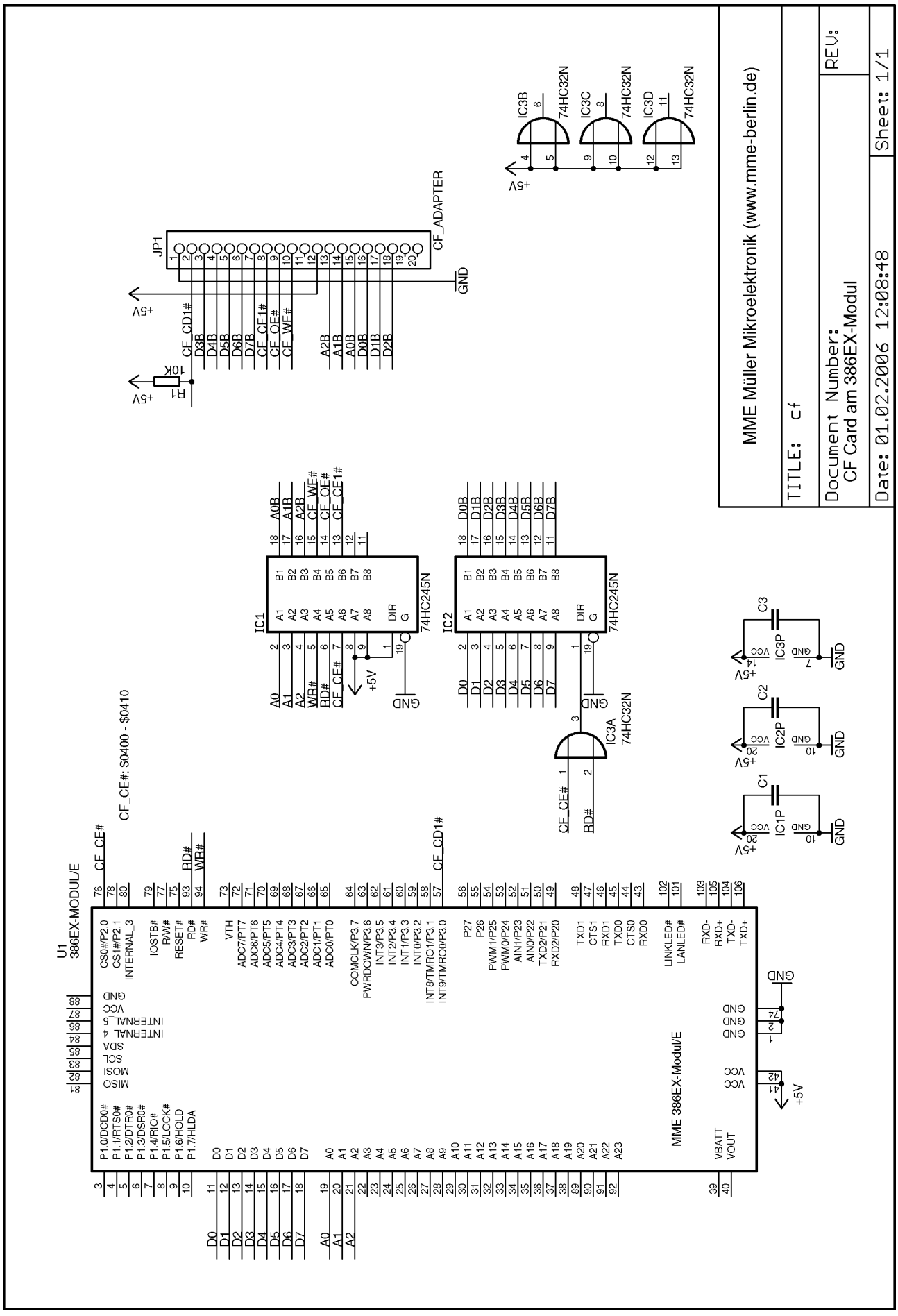
Schaltplan: Anschluß der seriellen Schnittstelle 1 (Download)



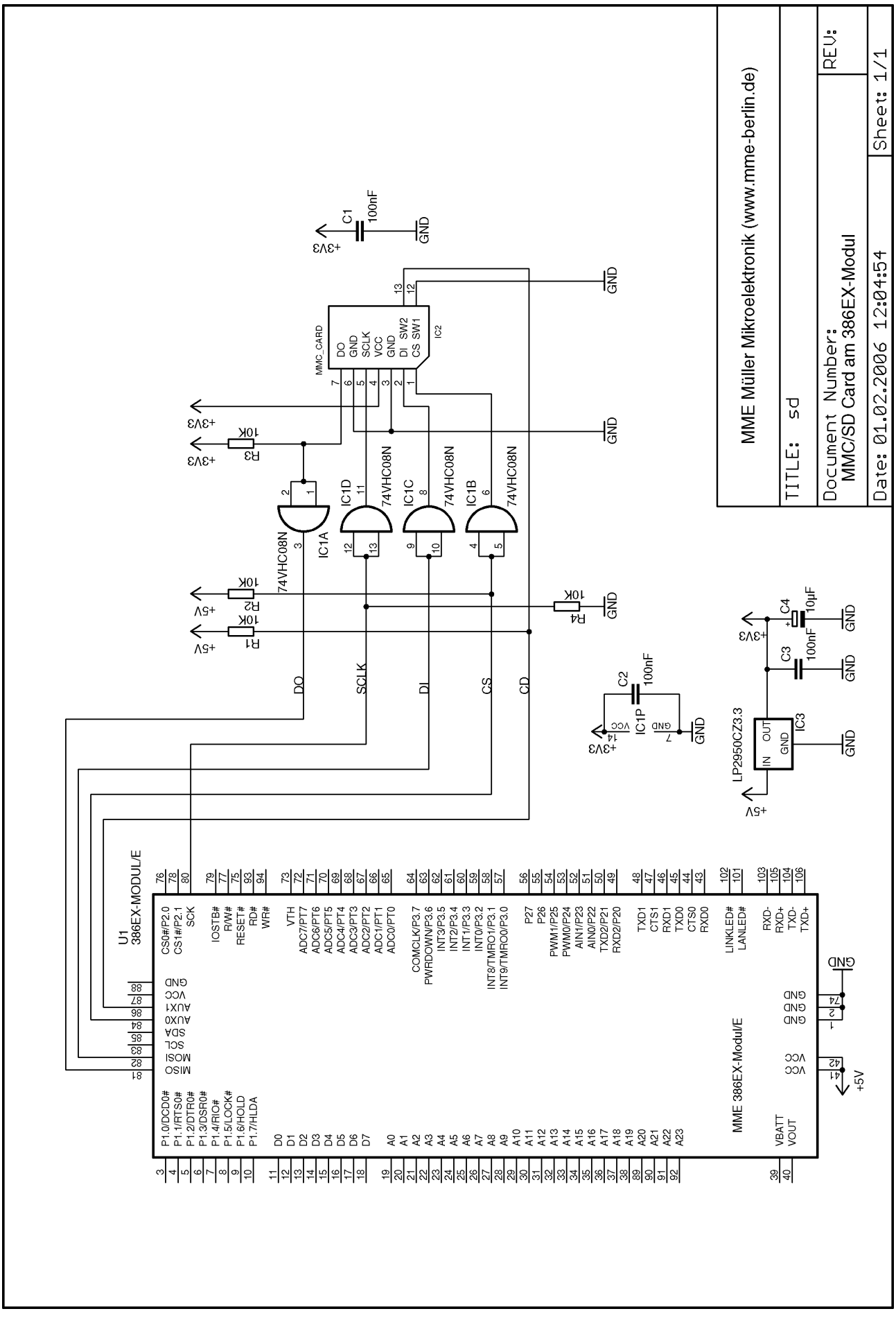
Schaltplan: 386EX-Modul aufgesteckt auf Basis-Board vom V25-Modul



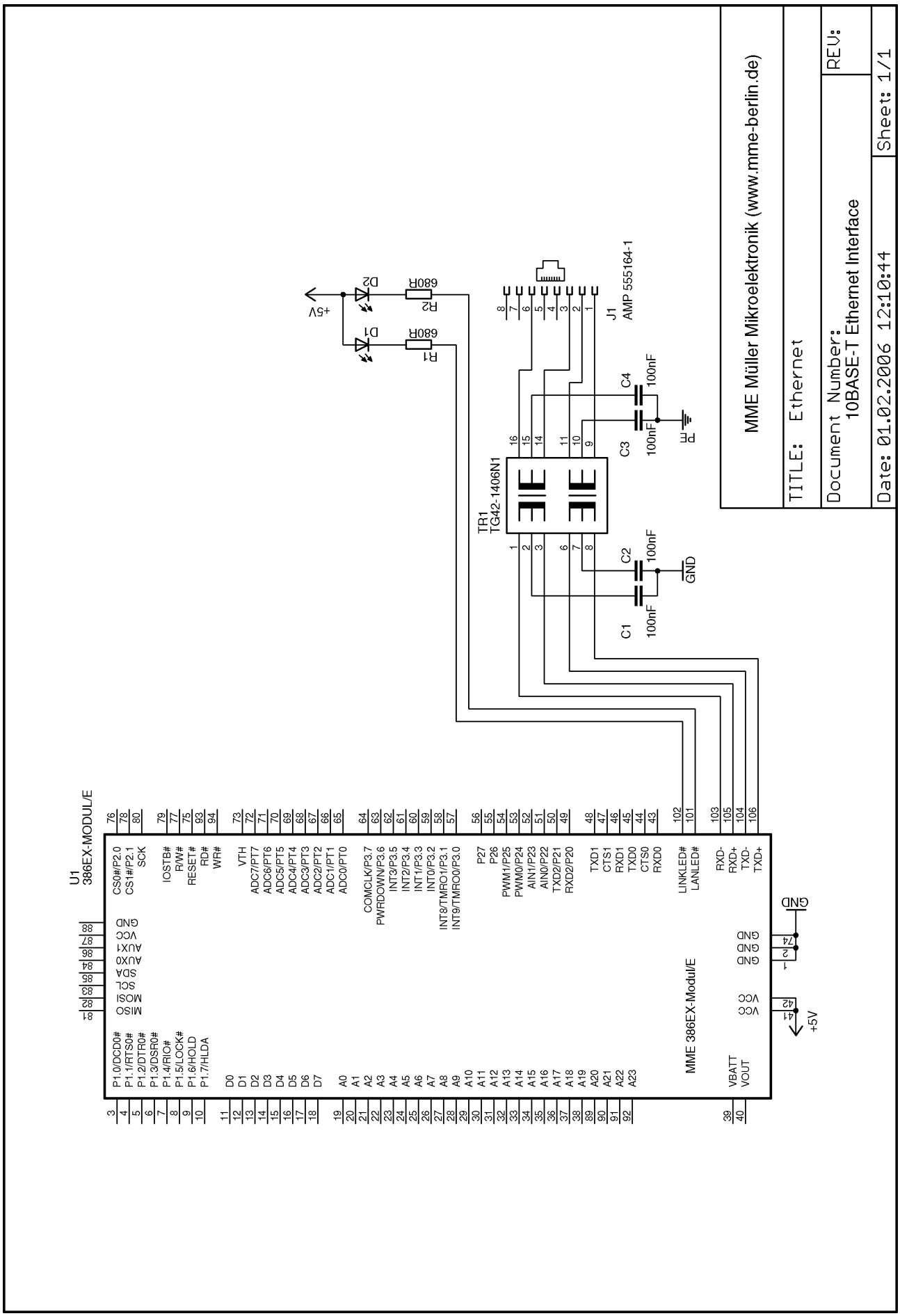
Schaltplan: Anschluß einer CompactFlash (CF) Card



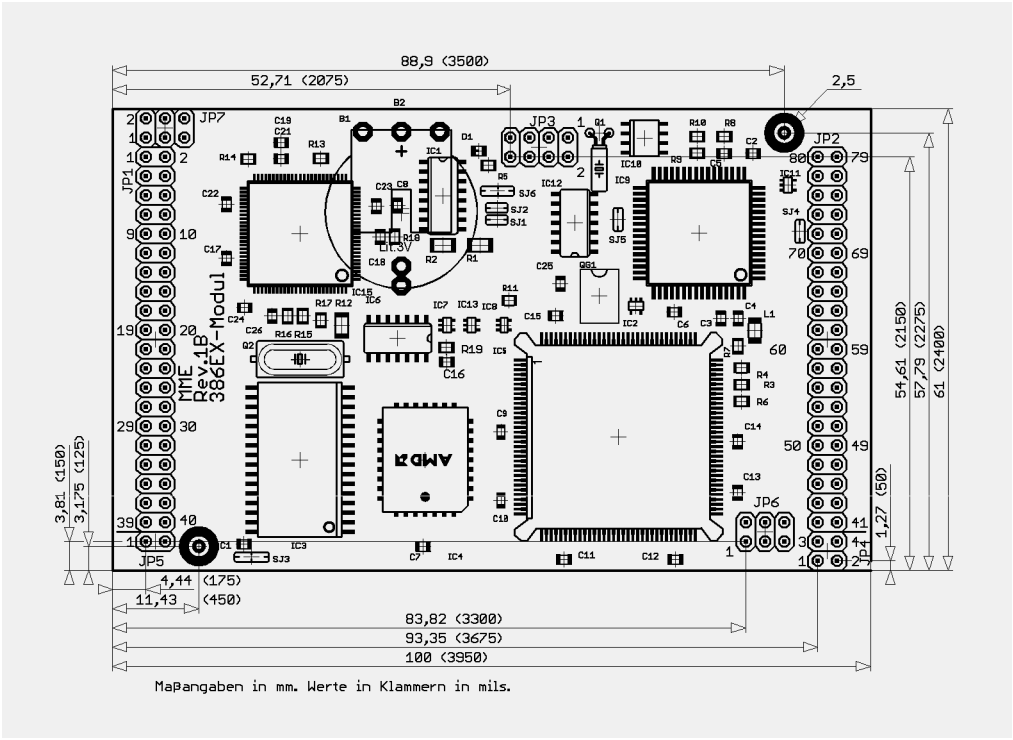
Schaltplan: Anschluß einer MMC/SD Card



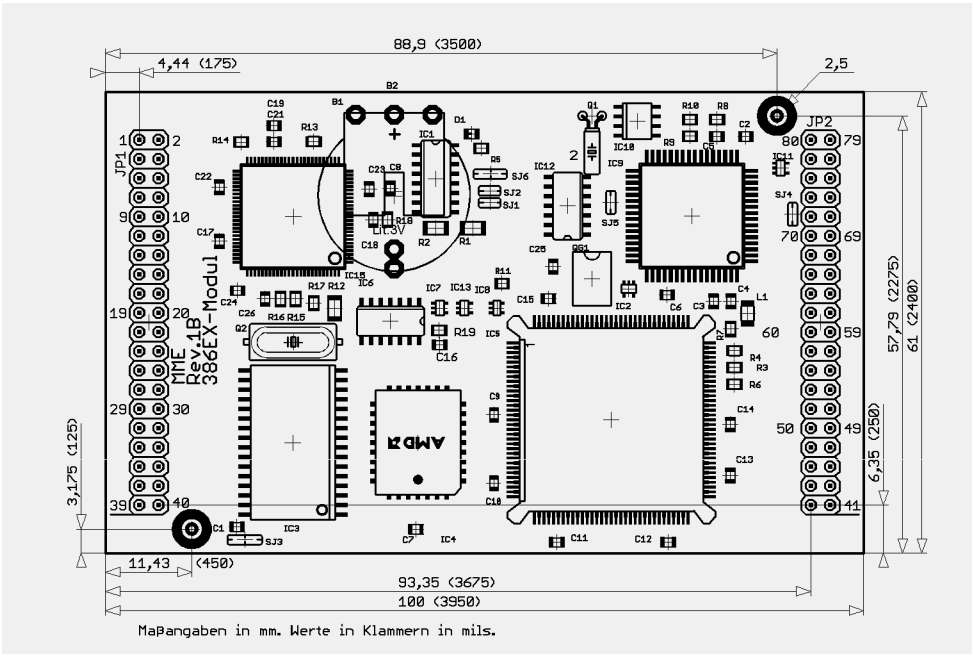
Schaltplan: 10BASE-T Ethernet Interface



Mechanische Abmessungen



Bemaßung des 386-Moduls



Bemaßung des 386EX-Moduls. Stifleisten JP1 und JP2 kompatibel zu *MME* V25-Modul

Alle Stiftleisten im Rastermaß 2,54 mm (100 mils).