13 Mikrocontroller

Mikrocontroller (µC) bilden das Herzstück von unzähligen Geräten und Steuersystemen, wie z.B. Spielkonsolen, Kaffeemaschinen, Mobiltelefone, Fernsehgeräten, Armbanduhren und vieles mehr. Ein Mikrocontroller beinhaltet hauptsächlich einen Prozessor, Speicher, digitale Ein- und Ausgänge und verschiedene Peripherien wie Timer oder A/D-Wandler. Alles ist in einem einzigen Chip integriert.

Ein µC ist in der Lage, wie ein PC-Prozessor auch, sehr viele Befehle in sehr kurzer Zeit durchzuführen. Er kann Zahlen miteinander verrechnen und vergleichen, digitale Signale einlesen und ausgeben. Die Befehlsliste, auch "Programm" genannt, wird als binären Code im Speicher abgelegt und der µC arbeitet diesen der Reihe nach ab, wobei auch Sprünge in der Befehlsliste programmierbar sind. Der Mikrocontroller hat somit keine Eigenintelligenz. Er führt nur das aus, was das "Programm" beinhaltet.

Bei unserem Board setzen wir ein MSP430-Mikrocontroller von Texas Instruments ein. Dieser besitzt 6 Ports mit je 8 Anschluss-Pins die als digitale Ein-/Ausgänge dienen oder Spezialfunktionen bieten. Auf unserem µC-Board sind nun mehrere Komponenten mit dem Mikrocontroller verbunden um unzählige Anwendungen realisieren zu können. Die externen Quarze liefern die Takte für den Mikrocontroller. Eine Übersicht der einzelnen Komponenten auf dem µC-Board liefert die Abbildung 13.1.

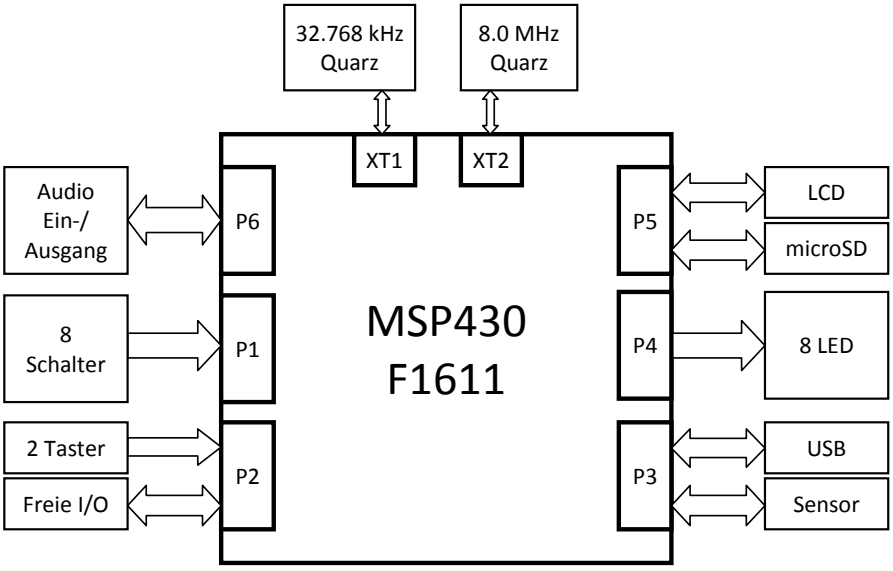


Abbildung 13.1: Übersicht der Komponenten auf dem µC-Board.

13.1 Programmierung des Mikrocontrollers

Wie gesagt, muss dem µC die Befehlsliste (Programm) in digitaler Form, also als binären Code, abgeliefert werden. Dies ist die einzige "Sprache", die der µC versteht. Um den µC zu "programmieren" schreibt man nun nicht den binären Code direkt, sondern man benutzt eine sogenannte Hochsprache. Mit Hilfe eines Übersetzungsprogramms, dem sogenannten Compiler, wird das Programm dann in binären Code übersetzt.

Wir benutzen als Hochsprache die Programmiersprache **C**. Das C-Programm wird als Text (sog. Quelltext) in einer oder mehreren Textdateien (sog. Quelldateien) auf dem PC geschrieben. Dieser Text kann in einem simplen Texteditor geschrieben werden. Quelldateien haben die Endung ".c". Der Compiler übersetzt die einzelnen Quelldateien zu Objektdateien, die binären Code enthalten. Ein weiteres Programm, der sog. Linker, fügt schliesslich alle Objektdateien zu einer einzelnen Binärdatei zusammen. Diese beinhaltet den ausführbaren Binärcode, der nun in den µC "heruntergeladen" werden kann.

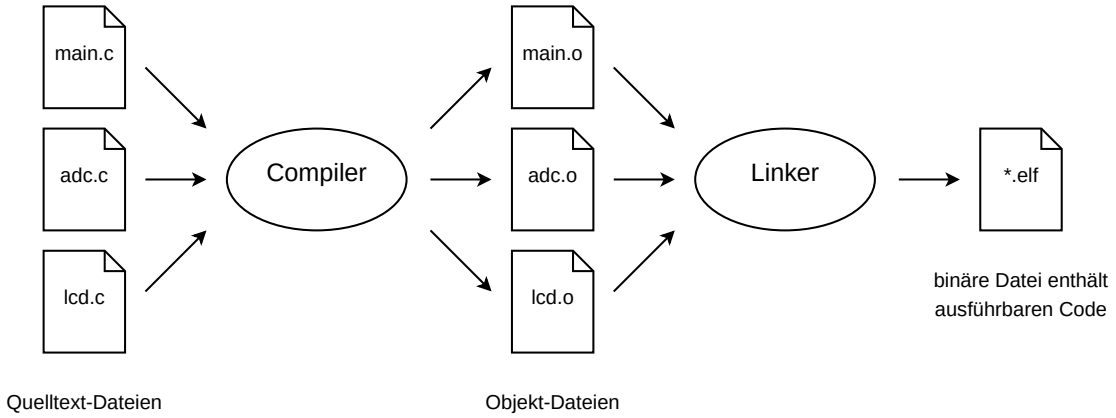


Abbildung 13.2: Übersetzung der Quelldateien zum ausführbaren Code.

Eine Beschreibung um die nötige Software wie Texteditor, Compiler und USB-Treiber zu installieren ist im Abschnitt 13.4.2 zu finden.

13.1.1 Die Struktur eines C-Programms

In diesem Abschnitt werden die wichtigsten Bestandteile eines C-Programms anhand eines Beispiels erläutert. Ein C-Programm besteht aus einzelnen "Bausteinen", sogenannte *Funktionen*, die eine bestimmte Aufgabe lösen und sich gegenseitig aufrufen können. Wobei jedes C-Programm eine Funktion mit dem Namen `main()` besitzt. Diese Funktion bildet das steuernde Hauptprogramm und wird automatisch nach jedem Start des µC aufgerufen. Die Bedeutung der weiteren Codezeilen kann im Buch *C kurz und gut* auf den angegebenen Seiten nachgeschlagen werden.

Listing 13.1: beispiel.c

```
1  /*
2      Dateiname:      beispiel.c
3      Beschreibung:   Dies ist ein Beispielprogramm.
4  */
5
6  #include "../System.h"           // Praeprozessor-Direktiven (Buch S.69)
7
8  int Summe(int a, int b);         // Funktionsprototyp (Buch S.59)
9
10 int main(void)                   // Hauptfunktion main()
11 {
12     int Zahl1;                   // Variable (Buch S.13)
13     int Zahl2;                   // Variable (Buch S.13)
14
15     Init_System();               // Funktionsaufruf (Buch S.62)
16
17     Zahl1 = 3;                   // Zuweisung (Buch S.24)
18     Zahl2 = 7;                   // Zuweisung (Buch S.24)
19
20     P4OUT = Summe(Zahl1,Zahl2); // Funktionsaufruf und Zuweisung
21
22     return 0;                    // Rueckgabewert (Buch S.62)
23 }
24
25 int Summe(int a, int b)          // Funktionsdefinition (Buch S.60)
26 {
27     return a+b;                  // Rueckgabewert (Buch S.62)
28 }
```

Das obige Beispiel zeigt die Struktur eines C-Programms. Das Programm besteht aus den Funktionen `main()` und `Summe()`. Es zeigt die Summe zweier Zahlen als binäre Zahl durch die LEDs an.

Grössere C-Programme werden in der Regel in mehreren Quelldateien aufgeteilt und getrennt bearbeitet. Dabei werden zusammengehörende Funktionen in dieselbe Quelldatei zusammengefasst. Informationen, die in mehreren Quelldateien erforderlich sind, wie z.B. Funktionsprototypen, werden in Header-Dateien (*.h) gestellt. Diese Dateien können mit der `#include`-Direktive in eine Quelldatei kopiert werden.

Weitere Details über den Bestandteilen eines C-Programms, die ein Programmierer kennen muss, findet man im Buch auf den folgenden Seiten:

Variablen und Datentypen	Buch S. 12-20
Arithmetische Operatoren, ...	Buch S. 21-31
Anweisungen (if, while, switch, ...)	Buch S. 34-40
Funktionen	Buch S. 58-64

Tabelle 13.1: Die wichtigsten Kapitel aus dem Buch

13.2 Funktionen aus `System.h`

Um die Programmierung mit unserem µC-Board zu erleichtern, wurde eine Reihe von nützlichen Funktionen geschrieben, die die verschiedenen Komponenten auf dem Board ansteuern. Diese Funktionen sind in der Quelldatei `System.c` enthalten und können über die Header-Datei `System.h` in das eigene Projekt mit `#include` importiert werden. Diese Funktionen werden nun hier aufgelistet:

void Init_System(void) Diese Funktion setzt die wichtigsten Grundeinstellungen, die für das Mikrocontroller-Board nötig sind, wie z.B. das Initialisieren der Ports oder das Aktivieren der externen Quarze für die Taktgenerierung.

void Delay_ms(unsigned long d) Diese Funktion haltet die Programmausführung für eine definierte Zeit an. Die Zeit wird mittels dem Parameter d in Millisekunden angegeben. Gültiger Wertebereich für d: 0...4'294'967'295.

void Init_TimerA(unsigned int CCR0_Wert) Diese Funktion initialisiert den Timer A. Mit dem Parameter *CCR0_Wert* wird das Zeitintervall bestimmt, nach dessen Ablauf der Timer periodisch ein Interrupt auslöst. Die Wiederholfrequenz des Timers berechnet sich wie folgt: $F(\text{in Hz}) = \frac{8000000}{\text{CCR0\_Wert}}$. Gültiger Wertebereich: 0...65'535.

void Init_UART(void) Initialisiert die serielle Schnittstelle (UART), die über USB mit dem PC kommuniziert. Die Einstellungen sind wie folgt:

Baudrate	9600
Datenbits	8
Parität	N
Stoppbits	1

void Init_LCD(void) Diese Funktion schaltet das Display ein und führt die nötigen Initialisierungen durch.

void lcdInstr(char cmd) Mittels dieser Funktion können verschiedene Befehle an das Display gesendet werden. Mögliche Befehle sind:

LCD_ON	Display einschalten.
LCD_OFF	Display ausschalten.
LCD_CLEAR	Display löschen.
LCD_LINE1	Springt zum Anfang der ersten Zeile.
LCD_LINE2	Springt zum Anfang der zweiten Zeile.
LCD_CURSORON	Schaltet den Cursor ein.
LCD_CURSOROFF	Schaltet den Cursor aus.

void printf_Auswahl(int Eingabe) Mittels dieser Funktion kann definiert werden, auf welcher Hardware die *printf*- und *puts*-Funktion ihre Zeichen ausgeben soll. Mögliche Werte für den Parameter *Eingabe* sind:

LCD	Wählt das Display als Ausgabemedium. Standardeinstellung.
UART	Wählt die serielle Schnittstelle (USB) als Ausgabemedium.

void Init_ADC12(int Eingabe) Diese Funktion initialisiert den Analog/Digital-Wandler. Mit dem Parameter *Eingabe* wird der entsprechende Eingang selektiert, der gewandelt werden soll. Die Auflösung beträgt 12 Bit und die Referenzspannung U_{ref} wird entsprechend dem Modus gesetzt. Mögliche Parameterwerte sind:

MUSIK	Wählt den Audio-Eingang (Stereo-Signal). $U_{ref} = 3.3\,V$
POTI	Wählt das Potentiometer. $U_{ref} = 3.3\,V$
TEMP	Wählt den Temperatursensor. $U_{ref} = 2.5\,V$

int ADC_Einzelwandlung(void) Diese Funktion führt eine einzelne A/D-Wandlung durch und gibt den digitalisierten Wert als Resultat zurück. Der Messwert liegt zwischen 0 (0 Volt) und 4095 (U_{ref}). Diese Funktion kann nur in Modus *POTI* oder *TEMP* benutzt werden.

void Init_DAC12(void) Diese Funktion initialisiert den Digital/Analog-Wandler. Die Auflösung beträgt 12 Bit und die Referenzspannung beträgt 3.3 V. Der D/A-Wandler ist fest mit dem Audio-Ausgangsverstärker verbunden.

13.3 Grundlegende Datentypen und Operatoren

Datentypen

Typ	Speicherplatz	Wertebereich
char	1 Byte	-128 bis 127
unsigned char	1 Byte	0 bis 255
short	2 Byte	-32768 bis 32767
unsigned short	2 Byte	0 bis 65535
int	2 Byte oder 4 Byte	-32768 bis 32767 oder -2147483648 bis 2147483647
unsigned int	2 Byte oder 4 Byte	0 bis 65535 oder 0 bis 4294967295
long	4 Byte	-2147483648 bis 2147483647
unsigned long	4 Byte	0 bis 4294967295

Typ	Speicherplatz	Wertebereich	Genauigkeit
float	4 Byte	1.2E-38 bis 3.4E+38	6 Stellen
double	8 Byte	2.3E-308 bis 1.7E+308	15 Stellen

Arithmetische Operatoren

Operator	Bedeutung	Beispiel	Ergebnis
*	Multiplikation	x * y	Produkt von x und y
/	Division	x / y	Quotient aus x und y
%	Modulodivision	x % y	Rest bei der Division x/y
+	Addition	x + y	Summe von x und y
-	Subtraktion	x - y	Differenz von x und y
++	Inkrement	x++	x wird inkrementiert (x=x+1)
--	Dekrement	x--	x wird dekrementiert (x=x-1)

Zuweisungsoperatoren

Operator	Bedeutung	Beispiel	Ergebnis
=	einfache Zuweisung	x = y	an x den wert von y zuweisen
op=	zusammengesetzte Zuweisung	x += y	x op= y ist äquivalent mit x = x op y

Vergleichsoperatoren

Operator	Bedeutung	Beispiel	**Ergebnis:** 1 (wahr) oder 0 (falsch)
<	kleiner	x < y	wahr, falls x kleiner als y
<=	kleiner gleich	x <= y	wahr, fall x kleiner oder gleich y
>	grösser	x > y	wahr, falls x grösser als y
>=	grösser gleich	x >= y	wahr, falls x grösser oder gleich y
==	gleich	x == y	wahr, falls x gleich y
!=	ungleich	x != y	wahr, falls x ungleich y

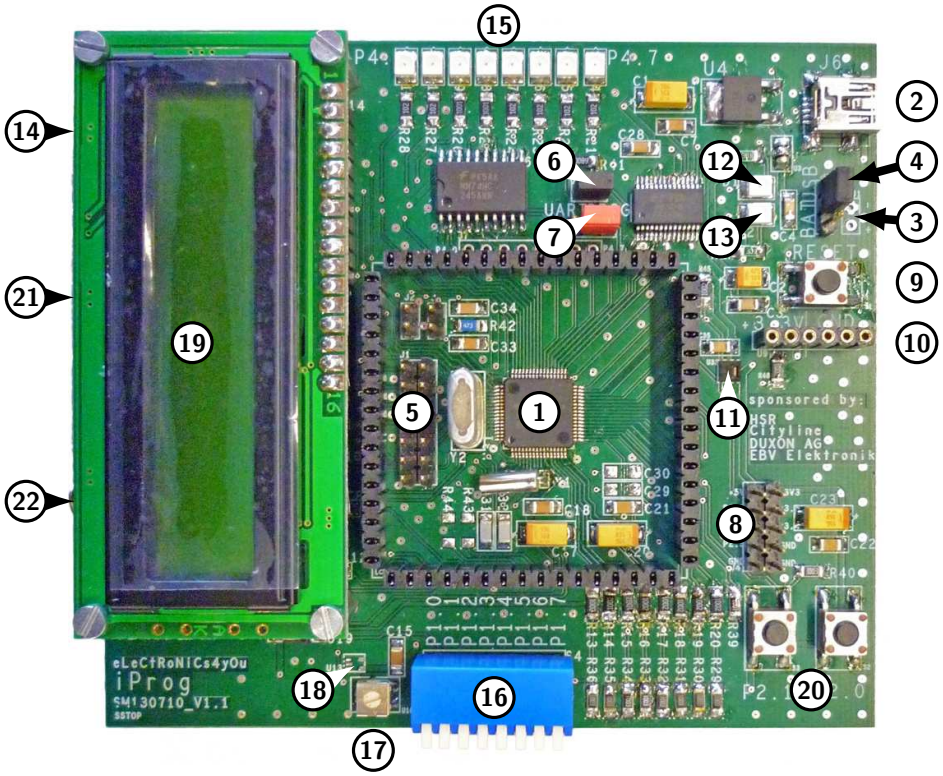
Logische Operatoren

Operator	Bedeutung	Beispiel	**Ergebnis:** 1 (wahr) oder 0 (falsch)
&&	logisches UND	x && y	wahr, falls x und y ungleich 0
||	logisches ODER	x || y	wahr, falls x oder y oder beide ungleich 0
!	logisches NICHT	!x	wahr, falls x gleich 0

13.4 Das Mikrocontrollerboard

Nun besitzt du dein erstes eigenes Mikrocontrollerboard und damit du es auch noch zu Hause verwenden kannst, werden hier die wichtigsten Informationen über das Board und die Software beschrieben.

13.4.1 Die Hardware



1. **MSP430F1611** Der Mikrocontroller steuert die verschiedenen Peripherien des Boardes an und ist somit der wichtigste Bestandteil, man könnte sagen, das Herz der Platine.

2. **USB-Schnittstelle** Über die USB-Schnittstelle werden die Programme herunter geladen, ausserdem kann das Board via USB gespeist werden.

3. **Batterieanschluss** Es besteht die Möglichkeit das Board mit Hilfe einer 9V-Batterie zu speisen.

4. **Wahl der Speisung** Diesen Jumper setzt man, um die gewünschte Speisungsart (USB oder Batterie) zu wählen.

5. **JTAG-Schnittstelle** Diese Stiftleiste ist eine andere Art von Programmier-Schnittstelle, die einen speziellen Adapter benötigt. Sie erledigt im Grunde dieselbe Arbeit, wie die USB-Schnittstelle, einfach wesentlich schneller. Da neuere Computer oft nicht mehr über den nötigen Parallelport verfügen, ist es wohl eher selten, dass diese Schnittstelle zu Hause verwendet werden kann.

6. **Jumper TCK** Will man das Programm nun doch über die JTAG-Schnittstelle laden, ist es wichtig, den Jumper U8 zu unterbrechen.

7. **Jumper UART/PROG** Dieser Jumper bestimmt die Funktion der USB-Schnittstelle. Um die Schnittstelle für die Programmierung des Mikrocontrollers zu aktivieren, muss der Jumper auf die Stellung "PROG" gesetzt werden. Die Stellung "UART" bedeutet, dass die Schnittstelle für die Datenkommunikation mit dem PC benutzt werden kann.

8. **Schnittstelle UART0 und freie Pins** Auf dieser Stiftleiste sind sowohl die zweite UART Schnittstelle wie auch freie Pins heraus geführt. Weitere Peripherien können bei Bedarf hier angeschlossen werden.

9. **Reset-Taste** Dieser Taster ist dazu gedacht, ein Programm neu zu starten. Durch drücken des Tasters, wird der Mikroprozessor zurück gesetzt, das geladene Programm wird somit neu gestartet.

10. **Speisungsbuchsen** Diese Buchsenleiste dient dazu, die vorhandene Speisung (3.3V/GND) zu messen bzw. etwas damit zu speisen.

11. **Temperatur- und Feuchtigkeitssensor** Mit diesem digitalen Sensor kann die Temperatur und die Luftfeuchtigkeit gemessen werden. Er ist am I^2C-Bus angeschlossen.

12. **Tx-LED** Diese LED leuchtet, wenn der Mikroprozessor Daten an den Computer sendet.

13. **Rx-LED** Diese LED leuchtet, wenn der Mikroprozessor Daten vom Computer empfängt.

14. **SD-Card Slot** Hier kann eine microSD-Card eingesetzt werden um Daten abzuspeichern. Es werden nur Karten mit max. 2 GB unterstützt.

15. **Ausgabe-LEDs** Diese acht LEDs sind mit dem Port 4 des MSP430 verbunden. Programmiert man eine 1 an einen der Ausgänge von Port 4, leuchtet die entsprechende LED.

16. **Eingabeschalter** Diese acht Schalter können über die kleinen Tasten ein- und ausgeschaltet und dessen Zustände über den Port 1 des MSP430 einzeln eingelesen werden.

17. **Potentiometer** Das Potentiometer ist ein verstellbarer Widerstand, der die Speisespannung (3 V) entsprechend aufteilen kann. Diese eingestellte Spannung wird mit dem Analogeingang A3 des Analog/Digital-Wandlers gemessen und digitalisiert.

18. **Temperatursensor** Dieser Sensor misst die Temperatur und gibt eine entsprechende Spannung an den Analogeingang A2 weiter, wo die Spannung gemessen werden kann.

19. **Display** Das Display besitzt zwei Zeilen mit je 16 Zeichen. Man kann es beispielsweise dazu verwenden, gewandelte Werte des Potentiometers anzuzeigen.

20. **Taster** Die beiden Taster sind mit dem Port 2 verbunden (P2.0 und P2.1). Man kann sie einzeln einlesen oder damit Interrupts auslösen.

21. **Audioeingang** Diese Buchse dient als Line-In. Das Stereo-Musiksignal von z.B. einem MP3-Player wird auf die Analogeingänge A0 bzw. A1 des Analog-Digital-Wandlers geführt. Damit kann die Musik digitalisiert und später z.B. auf die SD-Karte gespeichert werden.

22. **Audioausgang** Diese Buchse dient als Line-Out. Die Analogausgänge DAC0 und DAC1 des Digital-Analog-Wandlers sind über einen Vorverstärker mit dieser Buchse verbunden. Damit kann ein digitales Audiosignal ausgegeben werden und z.B. mit einem Kopfhörer gehört werden.

13.4.2 Die Programmierumgebung

Unsere Programmierumgebung besteht aus drei Teilen: einem **Texteditor**, einem **Compiler** (Übersetzer) und einem **Downloader**. Den Texteditor benötigen wir, um unsere Programme zu schreiben. Sobald das Programm als Quelltext geschrieben ist, benötigen wir den Compiler, um den Quelltext in binären Code zu übersetzen. Die Übersetzung ist wichtig, weil der Mikrocontroller nur binären Code versteht und keinen Quelltext. Schliesslich muss der binäre Code in den Mikrocontroller geladen werden. Dazu ist der Downloader nötig. Der Downloader kommuniziert über das USB-Kabel mit dem Mikrocontroller und überträgt so das neue Programm.

Installation des USB-Treibers

Damit du zu Hause das Mikrocontrollerboard am PC betreiben und programmieren kannst, musst du zuerst den USB-Treiber installieren. Diesen findest du auf unserer Webseite als Datei **ftdi_usb_treiber.exe**. Führe zuerst die Installation dieses Treibers durch und schliesse dann das Board an den PC an.

Installation des Texteditors

Den Installer des Texteditors SciTE findest du auf unsere Website als Datei **scite_setup.exe**. Starte den Installer und es erscheint ein Installationsfenster:

1. Klicke 2-mal auf **Next**.
2. Wähle einen geeigneten Speicherort (*C:\Programme\Scintilla Text Editor* ist OK) und klicke auf **Next**.
3. Es wird gefragt, ob du den Ordner "Scintilla Text Editor" nennen willst, klicke auf **Next**.
4. Auswahl zu den Icons, alles so lassen und klicke auf **Next**.
5. Klicke auf **Install**.
6. Klicke auf **Finish**.

Der Text Editor ist jetzt installiert. Kopiere nun die Datei **SciTEGlobal.properties** aus *install_software.zip*, die auf unserer Website zu finden ist, in den Installationsordner von SciTE (z.B. *C:\Programme\Scintilla Text Editor*). Dieser Ordner enthält bereits eine gleichnamige Datei. Überschreibe sie einfach mit der neuen.

Wenn du auf deinen Desktop schaust, wirst du feststellen, dass dort eine neue Verknüpfung entstanden ist, welche "Scintilla Text Editor" heisst. Benutze diese Verknüpfung, um den Texteditor zu starten oder rechts-klick auf eine Quelltext-Datei und wähle *Edit with SciTE*, um diese zu öffnen.

Installation des MSPGCC

Die weiteren Tools, wie Compiler und Downloader sind im Softwarepaket *MSPGCC* enthalten. Diese Software findest du wiederum auf unserer Webseite als Datei **mspgcc-20081230.exe**. Starte diesen Installer und es erscheint ein Installationsfenster. Führe die folgenden Schritte aus, um die Software zu installieren:

1. Klicke auf **Next**.
2. Klicke auf **I Agree**.
3. Klicke auf **Next**.
4. Wähle einen geeigneten Speicherort (C:\mspgcc ist eigentlich OK) und klicke auf **Next**.
5. Klicke auf **Install**.
6. Es erscheint ein schwarzes Fenster, drücke eine beliebige Taste und das Fenster verschwindet wieder.
7. Es erscheint wieder das normale Installationsfenster. Wähle die beiden Häkchen und klicke auf **Next**.
8. Klicke auf **Finish**.
9. Es erscheint ein Textfile, welches du einfach schliessen kannst.

Nun ist der Compiler auf deinem Computer installiert. Das einzige, was du jetzt noch tun musst, ist die beiden Dateien **make.exe** und **rm.exe** aus *install_software.zip* in den MSPGCC-Ordner (z.B. C:\mspgcc\bin) zu kopieren. Diese zwei Dateien werden benötigt, damit aus dem Texteditor die Kommandos im Makefile aufgerufen werden können. Wichtig noch zu wissen ist, dass du den Compiler und den Downloader nicht extra starten musst. Diese werden direkt aus dem Texteditor gestartet und arbeiten im Hintergrund.

13.4.3 Dein erstes Programm

Nun bist du bereit, um dein erstes Programm zu schreiben. In dieser Anleitung steht, wie du am besten vorgehst. Zuallererst solltest du auf deinem PC einen neuen Ordner erstellen, in dem du alle deine Programme abspeicherst. Der Ordnername könnte z.B. *Meine_Programme* heissen. Wenn du nun ein neues Programm schreibst, erstelle in deinem Ordner *Meine_Programme* dafür einen Unterordner und speichere die Daten dort hinein. So hast du immer eine gute Übersicht und findest deine Programme sehr schnell wieder.

Wie du ein neues Programm für den Mikrocontroller erstellst, wird hier nun erklärt:

1. Erstelle im Ordner *Meine_Programme* einen Unterordner mit dem Namen **Schalter**.
2. Kopiere in den Ordner *Schalter* die Datei **makefile**, sowie die beiden Dateien **System.c** und **System.h**. Diese drei Dateien findest du auf unserer Webseite in der Datei **uc_programme_xxxx.zip**.
3. Starte den Texteditor SciTE und es erscheint eine weisse Schreibfläche mit dem Registernamen “untitled”.
4. Tippe den untenstehenden Quelltext in diese Schreibfläche ab.

Listing 13.2: Schalter.c

```
1  #include "System.h"
2
3  int main( void )
4  {
5      Init_System();
6      while(1) {
7          P4OUT = P1IN;
8      }
9      return 0;
10 }
```

5. Klicke nun im Menü auf **File** und dann auf **Save as**..., es erscheint ein Dialogfenster. Wähle dort deinen Ordner *Schalter* aus, gib dem Programm den Namen **Schalter.c** und klicke auf **Save**. (**WICHTIG:** Du musst immer ein ".c" beim Namen anfügen, ansonsten funktioniert das Programm nicht!)
6. Klicke erneut im Menu auf **File** und wähle **Open**..., es erscheint ein Dialogfenster. Wähle die Datei **makefile** an, die du vorhin im gleichen Ordner hinein kopiert hast und klicke auf **Open**. Du hast nun zwei Registerkarten geöffnet, einmal dein Programm *Schalter.c* und einmal das *makefile*.
7. Im nächsten Schritt geht es darum, das Makefile deinem Programm anzupassen. In der Zeile: `OBJECTS =`
   musst du nun deine Dateien einfügen. In diesem Falle sind es *Schalter.c* und *System.c*, denn diese beiden Files benötigst du für dein Programm. Die Dateien fügt man folgendermassen ein:
   ```
   OBJECTS = Schalter.o System.o
   ```
   WICHTIG: Man schreibt "**.o**" und nicht "**.c**" im Makefile!
8. Die nächste Einstellung, welche du im Makefile vornehmen musst, ist die des COM-Ports. Mache dafür einen Rechtsklick auf das Arbeitsplatz-Symbol auf dem Desktop. Wähle **Einstellungen**, es erscheint das Einstellungsfenster. Klicke auf die Registerkarte **Hardware** und danach auf **Geräte-Manager**. Es erscheint ein neues Fenster mit diversen Informationen über deinen PC. Klappe den Punkt **Anschlüsse** auf. Dort siehst du deine verschiedenen Anschlüsse (Kommunikationsport etc.). Ein Port heisst **USB Serial Port** und in Klammer steht "**COM x**". Das "x" steht für eine Zahl, die ist von Computer zu Computer verschieden. Es steht zum Beispiel die Zahl 10. Gehe nun zurück in den SciTE zum Makefile. Dort gibt es die Zeile:
   ```
   download-bsl: all
       msp430-bsl -c N -e --invert-test --invert-reset $(NAME).elf
   ```
   Anstelle des "N" kommt nun die **UM 1 KLEINERE Zahl** (N=x-1), als du im Geräte-Manager gelesen hast. Das heisst, wenn im Geräte-Manager "COM 10" stand, dann sieht deine Zeile so aus:
   ```
   download-bsl: all
       msp430-bsl -c 9 -e --invert-test --invert-reset $(NAME).elf
   ```
   Falls du nun ein neues Programm schreibst, kannst du dein neu eingestelltes Makefile verwenden. Kopiere es einfach in den Ordner des neuen Programmes. Worauf du noch achten musst, ist, in der Zeile "OBJECTS = " die richtigen Dateinamen einzufügen.

9. Nun hast du ein funktionstüchtiges Programm, welches nur noch darauf wartet heruntergeladen zu werden. Klicke dafür im Menü auf **Tools**. Es erscheint ein Register mit verschiedenen Auswahlmöglichkeiten. Klicke dann auf **Download**. Der Compiler beginnt nun zuerst mit der Übersetzung deines Programms. Falls dein Programm keine Schreibfehler enthält, wird der fertige Binärcode mit dem Downloader auf den Mikrocontroller geladen. Falls du das Programm nur übersetzen willst ohne das Download, dann wähle **Build**. Die Ausgabe des Compilers und des Downloaders erscheint in der rechten Hälfte des Fensters. Somit siehst du immer was gerade geschieht. Wenn der Download fehlerfrei abgeschlossen ist, sollten auf der rechten Seite ungefähr diese Zeilen stehen:

```
-------------------------------------------------------
MSP430 Bootstrap Loader Version: 2.0
Mass Erase...
Transmit default password ...
Invoking BSL...
Transmit default password ...
Current bootstrap loader version: 1.61 (Device ID: f16c)
Program ...
4734 bytes programmed.
Verify ...
>Exit code: 0 Time: 16.504
-------------------------------------------------------
```

10. Nun kannst du das Programm testen und schauen, ob es auch funktioniert. Kippe einen Schalter und schaue, ob die dazugehörige LED leuchtet.

Viel Spass beim Programmieren!

Menge	Name	Beschreibung	Wert/Typ	Lieferant	Art.Nr.
5	C1, C6, C17, C20, C23	Tantal Elko	10uF	Distrelec	810983
1	C2	Keramikkondensator	4.7uF	Farnell	1463404
13	C3, C4, C5, C7, C14, C15, C16, C18, C21, C22, C28, C34, C35	Keramikkondensator	100nF	Farnell	1650885
5	C8, C9, C10, C12, C19	Keramikkondensator	1uF	Farnell	1650888
2	C11, C13	Tantal Elko	100uF	Distrelec	811025
4	C24,C25,C26,C27	Keramikkondensator	1nF	Farnell	1414709
4	C29, C30, C31, C32	Keramikkondensator	12pF	Farnell	1650892
1	C33	Keramikkondensator	10nF	Farnell	1520307
1	D1	LED grün		Distrelec	254569
9	D2, D4, D5, D6, D7, D8, D9, D10, D11	LED rot	LED	Distrelec	254571
1	D3	Diodenarray	BAV99	Distrelec	603652
1*	J1*	Stiftleiste	2x7pol	Distrelec	122511
3*	J2*, J3*, U8	Stifleiste	1x2pol	Distrelec	122350
1	U2, U7	Stifleiste	1x3pol	Distrelec	122351
1	U14	Stiftleiste	2Stk. 1x8pol	Distrelec	122382

*nicht bestücken

1	J4	Buchsenleiste	2x5pol	Distrelec	121570
1	J5	Buchsenleiste	1x10pol	Distrelec	121550
1	J7	Buchsenleiste	1x8pol	Distrelec	120593
1	U9	Buchsenleiste	1x6pol	Distrelec	120592
2*	U32*	Buchsenleiste	1x36pol	Distrelec	122215
1	J6	Mini USB B-Typ Buchse	USB	Farnell	1243250
2	R1, R2	Widerstand	300R	Farnell	1653109
10	R3, R4, R5, R6, R7, R8, R9, R10, R37, R38	Widerstand	20k	Farnell	1469993
1	R11	Widerstand	560R	Farnell	1653148
11	R12, R29, R30, R31, R32, R33, R34, R35, R36, R39, R40	Widerstand	100k	Farnell	1469975
10	R13, R14, R15, R16, R17, R18, R19, R20, R45, R46	Widerstand	10k	Farnell	1469970
8	R21,R22, R23, R24, R25, R26, R27, R28	Widerstand	1k	Farnell	1469965
1	R41	Widerstand	680R	Farnell	9240934
1	R42	Widerstand	47k	Farnell	1470016
2	R43, R44	Widerstand	0R	Farnell	1469963
3	S1, S2, S3	Taster		Farnell	3801287
1	S4	Piano Schalter	8pol	Distrelec	210334
1	U3	EMI Filter	NFE31P	Farnell	9528172
1	U4	3.3 V Spannungsregler	BA033FP	Distrelec	645131
1	U6	USB UART	FT232RL	Farnell	1146032
1	U10	Stereo Audioverstärker	TPA6111A2	Farnell	8456593
2	U11, U12	Klinkenbuchse	3.5mm	Distrelec	150419
1	U13	Temperatursensor	LM20	Farnell	1469223
1	U16	Trimmpotentiometer	10k	Distrelec	740037
1	U19	Analogschalter SPDT	ISL84544IBZ	Farnell	9664025
1	U26	Octal Bus Transceiver	74HC245A	Distrelec	643332
1	-	MicroSD Card	2GB	Distrelec	682890
1	U28	MicroSD Card Sockel		Farnell	1558178
1	U31	Mikrocontroller	MSP430F1611	Farnell	1470487
1	U33	Feuchte- und Temperatursensor	SHT21	Sensirion	
1	Y1	Uhrenquarz	32768Hz	Distrelec	644842
1	Y2	Quarz	8MHz	Distrelec	644082
1	-	Punktmatrix-LCD	DEM16213SYH	Distrelec	661416
1	-	Buchsenleiste für LCD	1x16pol	Distrelec	122215
3	-	Kurzschlussstecker		Distrelec	121538
4	-	Distanzhalter	M2.5x11mm	Farnell	1466760
8	-	Zylinderschrauben Schlitz	M2.5x4mm	Distrelec	343610
4	-	Gummifüsse		Distrelec	340469

Tabelle 13.2: Stückliste des Mikrocontrollerboards

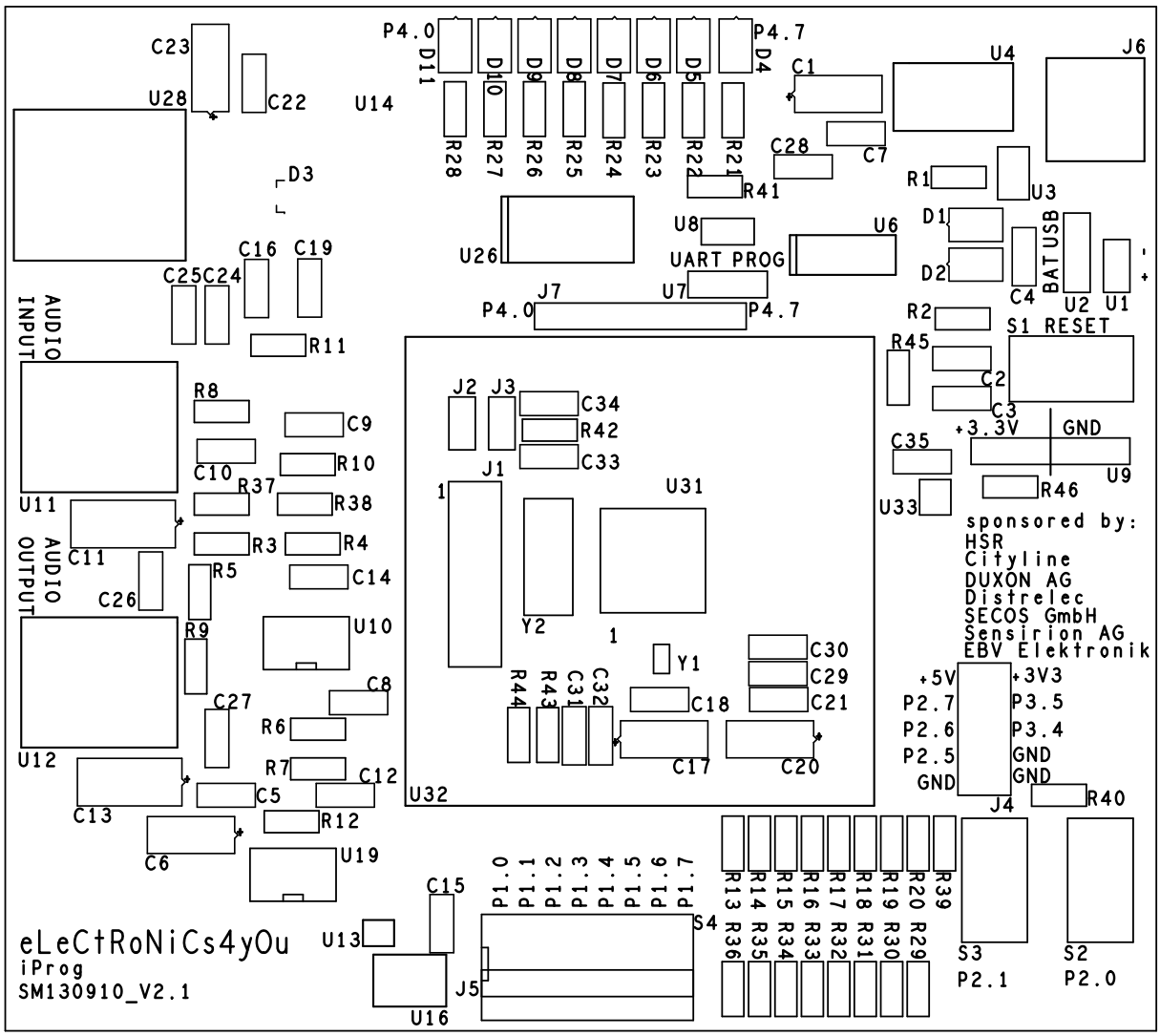


Abbildung 13.3: Bestückungsplan der Boardoberseite.

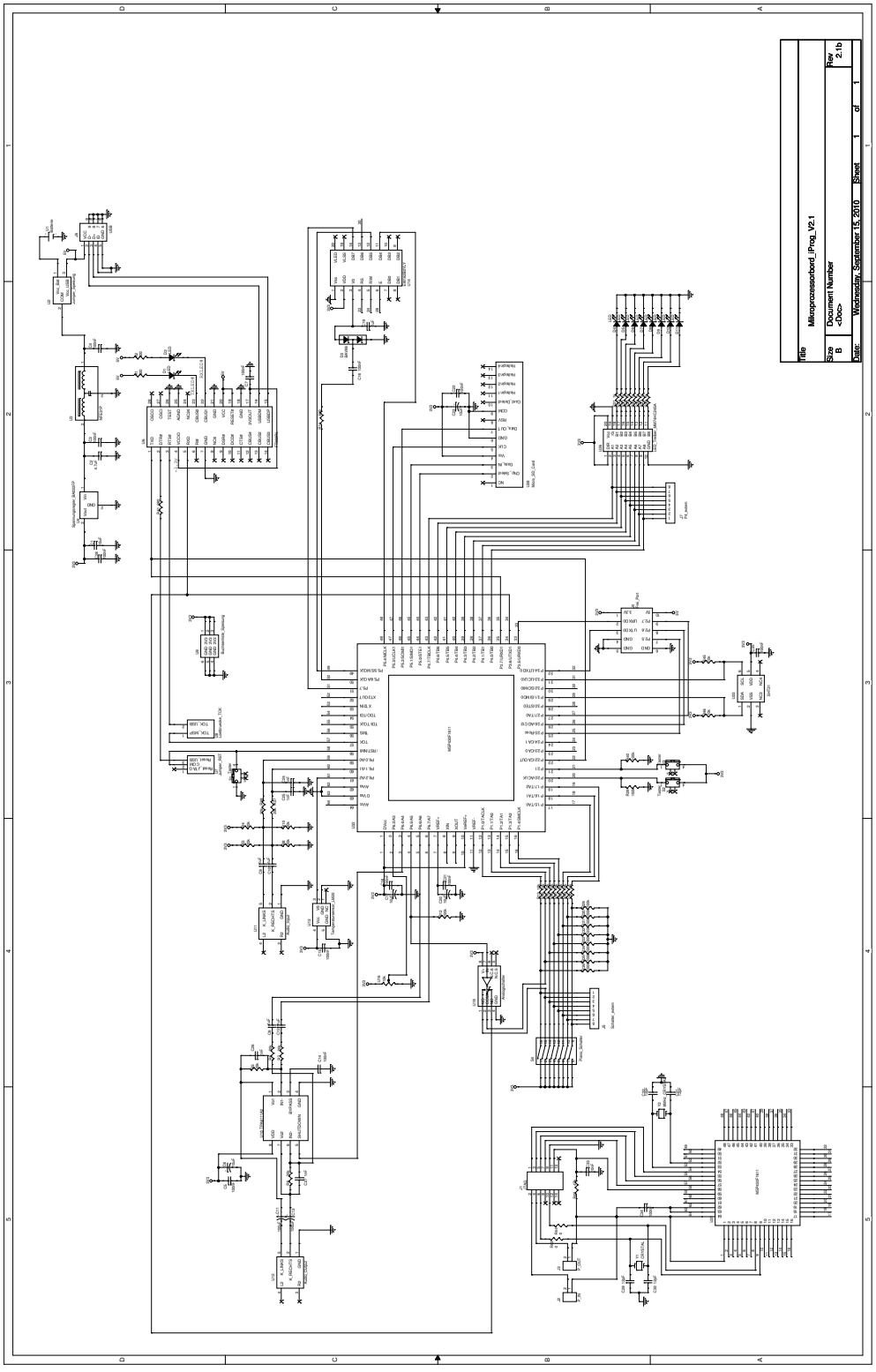


Abbildung 13.4: Schema des Mikrocontrollerboards.