ANHANG

Diplomarbeit 2008

Höhere Fachschule Uster

© ***Philippe Villard 05-E (Elektronik) / 19.08.2008***

Doku_DA_ANHANG.doc

1 Inhaltsverzeichnis

1 Inhaltsverzeichnis 1
2 Betreuungssitzungen 2
2.1 Betreuungssitzung 1 2
2.2 Betreuungssitzung 2 3
2.3 Betreuungssitzung 3 4
2.4 Betreuungssitzung 4 5
3 Stückliste 6
4 Gehäuseskizze 8
4.1 Frontansicht 8
4.2 Seitenansicht (Rechte Seite) 9
5 Quellcode mit Kommentaren 10
5.1 main.c 10
5.2 leser.c 14
5.3 leser.h 23
5.4 beep.c 25
5.5 beep.h 27
5.6 Tastatur.c 29
5.7 Tastatur.h 32
5.8 Nokia_lcd.c 33
5.9 Nokia_lcd.h 39
5.10 Div_Funk.c 44
5.11 Div_Funk.h 46

2 Betreuungssitzungen

2.1 Betreuungssitzung 1

Höhere Fachschule Uster ***HFU***

Protokoll Betreuungssitzung Nr: 1

Art der Arbeit	O Vordiplomarbeit ☑ Diplomarbeit O Abschlussarbeit NDS
Student	Phillippe Villard
Betreuer	Matthias Kläy
Thema	RFID
Ort, Datum, Zeit	BZU, 2008-05-21, 17.00 - 17.30

Arbeitsstand	• Ziele definiert • Zeitplan erstellt • Evaluationsmaterial beschafft • Dokugerüst erstellt
Probleme, Fragen	• Tastatur zu definieren • Display zu definieren
Weiteres Vorgehen	• Hardwarematerial beschaffen • Schema erstellen bis spätestens nächste Sitzung • Doku laufend erstellen
Beschlüsse	Die Aufgabenstellung ist verstanden. Sie ist im Pflichtenheft ergänzt und präzisiert. Das Pflichtenheft ist genehmigt. (Hinweis: dieser Satz ist für die erste Besprechung verbindlich und wird durch die Unterschriften bestätigt. Für die weiteren Besprechungen muss er gelöscht werden.) • Ziele definiert, Pflichten mit präzisierten Zielen am nächsten Sitzung
Nächster Termin	2008-06-27

Der Betreuer gibt dem Studenten jeweils eine Kopie. Er sammelt die ausgefüllten Protokolle und bewahrt sie auf bis 1 Monat nach der Präsentation (Ablauf Rekursfrist).

Unterschrift Betreuer Unterschrift Student

Freigabe: Schulleitung HFU / 25.4.2006 H40405-10 1 von 1
\\ch02sf00\klaey-1$_Matthias\Admin\MK\HFU\DA\2008 - Phillippe Villard\Protokoll Betreuungssitzung 1.doc

2.2 Betreuungssitzung 2

Höhere Fachschule Uster **HFU**

Protokoll Betreuungssitzung Nr: 2

Art der Arbeit	O Vordiplomarbeit ☑ Diplomarbeit O Abschlussarbeit NDS
Student	Phillippe Villard
Betreuer	Matthias Kläy
Thema	RFID
Ort, Datum, Zeit	BZU, 2008-06-27, 17:00-17:30
Arbeitsstand	• Prototyp HW erstellt inkl. Tastatur, Display • IAR MSP430 Quickstart läuft
Probleme, Fragen	• ca. 1½ Wochen Rückstand ⇒ SD-Karte elektrisch OK, SW wird nicht umgesetzt • Ziemlich Respekt vor der SW-Entwicklg
Weiteres Vorgehen	• HW bis 2008-06-29 fertigstellen • SW Design, dann Programmierung • Abklärungen bez. vertraulichem RFID Protokoll Philippe → Hans Muller
Beschlüsse	• Wenn gewünscht Source-Code bis 1 Wo vor des nächsten Sitz. • Design der Software mit Hilfe von gelehrten Techniken vor dem Programmier
Nächster Termin	2008-07-17

Der Betreuer gibt dem Studenten jeweils eine Kopie. Er sammelt die ausgefüllten Protokolle und bewahrt sie auf bis 1 Monat nach der Präsentation (Ablauf Rekursfrist).

Unterschrift Betreuer Unterschrift Student

Freigabe: Schulleitung HFU / 25.4.2006 H40405-10 1 von 1
\\ch02sf00\klaey-1$_Matthias\Admin\MK\HFU\DA\2008 - Phillippe Villard\Protokoll Betreuungssitzung 2-4.doc

2.3 Betreuungssitzung 3

Höhere Fachschule Uster ***HFU***

Protokoll Betreuungssitzung Nr: 3

Art der Arbeit	O Vordiplomarbeit ☑ Diplomarbeit O Abschlussarbeit NDS
Student	Phillippe Villard
Betreuer	Matthias Kläy
Thema	RFID
Ort, Datum, Zeit	BZU, 2008-07-17, 18:15-19:00

Arbeitsstand	• Elektronik läuft • Periferie-Software läuft
Probleme, Fragen	• Programlogik implementier • ca. 1 Woche hinter Zeitplan
Weiteres Vorgehen	• State-Event Diagramm ausbauen State-Maschine implementieren • Einfaches Gehäuse bauen
Beschlüsse	• Programlogik bis 2008-07-31 implementieren
Nächster Termin	2008-08-06

Der Betreuer gibt dem Studenten jeweils eine Kopie. Er sammelt die ausgefüllten Protokolle und bewahrt sie auf bis 1 Monat nach der Präsentation (Ablauf Rekursfrist).

Unterschrift Betreuer Unterschrift Student

Freigabe: Schulleitung HFU / 25.4.2006 H40405-10 1 von 1

\\ch02sf00\klaey-1$_Matthias\Admin\MK\HFU\DA\2008 - Phillippe Villard\Protokoll Betreuungssitzung 2-4.doc

2.4 Betreuungssitzung 4

Höhere Fachschule Uster ***HFU***

Protokoll Betreuungssitzung Nr: 4

Art der Arbeit	O Vordiplomarbeit ☑ Diplomarbeit O Abschlussarbeit NDS
Student	Phillippe Villard
Betreuer	Matthias Kläy
Thema	RFID
Ort, Datum, Zeit	BZU, 2008-08-06, 17:45 - 19:15
Arbeitsstand	Statemachine implementiert Programm läuft Doku zu 70% erstellt, Sicherheit & Testing Flo und Aufbau erstellt fehlt noch
Probleme, Fragen	Display-Kontaktierung nicht optimal
Weiteres Vorgehen	Testing, inkl. Doku Doku abschliessen
Beschlüsse	Statemachine: Zähler → Enum!
Nächster Termin	Abgabe, Präsentation

Der Betreuer gibt dem Studenten jeweils eine Kopie. Er sammelt die ausgefüllten Protokolle und bewahrt sie auf bis 1 Monat nach der Präsentation (Ablauf Rekursfrist).

Unterschrift Betreuer Unterschrift Student

Freigabe: Schulleitung HFU / 25.4.2006 H40405-10 1 von 1
\\ch02sf00\klaey-1$_Matthias\Admin\MK\HFU\DA\2008 - Phillippe Villard\Protokoll Betreuungssitzung 2-4.doc

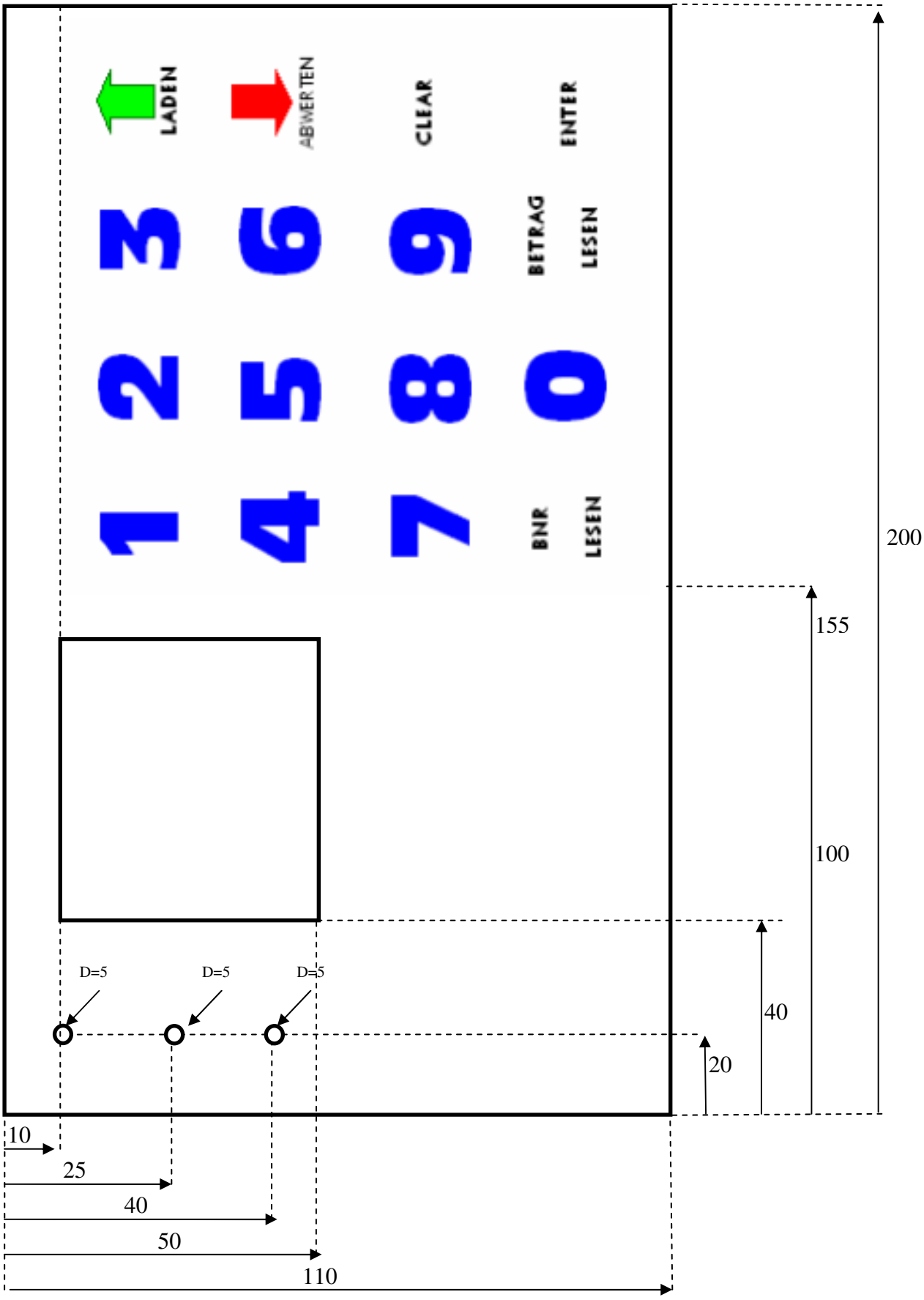
3 Stückliste

Comment	Description	Designator	Distrelec	Quantity
MHDR1X12	RFID Reader	P2	RFID Reader	1
MSP430F169 Olimex	MSP430F169	U3	Redacom	1
32.768kHz	Quarz 32.768kHz	Q1	Header Board	1
8MHz	Quarz 8MHz SMD	Q2	Header Board	1
JTAG	Flat Cable Connector (IDC), PCB Transition Connector, 2 Rows, Kinked Solder Pin, 14 Contacts	P1	Header Board	1
Kondensator Header Board	Kondensator Header Board	C10, C15, C18	Header Board	3
Kondensator Header Board	Kondensator Header Board	C11, C12	Header Board	2
Kondensator Header Board	Kondensator Header Board	C16, C19	Header Board	2
P_IN	Jumper Wire	W1	Header Board	1
P_OUT	Jumper Wire	W2	Header Board	1
Gehäuse	Bopla 32205062	-	Bopla 32205062	1
Widerstand Header Board	Widerstand Header Board	R3	Header Board	1
Header 8	Matrix LCD Nokia 3310	P3	Ammontec amt-500-1001	1
Steckernetzgerät	12VDC / 700mA	-	921171	1
Gerätedose Triad	T01-0580-S03	-	115004	1
Gerätestecker Triad	T01-0550-P03	-	114989	1
Buchse (50Pol.)	Sockel für Bauteile	-	121562	5
Laborkarten Europa	RE 317 LF - Laborkarten Europa	-	451586	1
Battery	Batteriehalter 9 V	BAT1	970033	1
Cap 0603	Capacitor 0603	C6	831547	1
Kondensator Speisung 3.3V	Kondensator Speisung 3.3V	C1, C2	810326	2
Kondensator Speisung 5V	Kondensator Speisung 5V	C3, C4	810326	2
Kondensator Display	Kondensator Display	C20	810322	1
Cap Pol1	Polarized Capacitor (Radial)	C5, C7	800918	2
Kondensator MAX232	Kondensator MAX232	C8, C9, C13, C14, C17	800915	5
Widerstand Tastatur	Widerstand Tastatur	R4, R5, R6, R7	714196	4
Widerstand 1 Display	Widerstand 1 Display	R19	714136	1
Widerstand Display	Widerstand Display	R17, R18	714107	2
Widerstand Speisung 3.3V	Widerstand Speisung 3.3V	R1	714083	1
Widerstand Speisung 5V	Widerstand Speisung 5V	R2	714083	1
Widerstand Status LED	Widerstand Status LED	R8, R9, R10, R11, R12	714083	5
Res0805	Resistor	R13, R14, R15, R16	713338	4
MAX232ACPE	Powered, Multi-Channel RS-232 Driver/Receiver	U4	647219	1

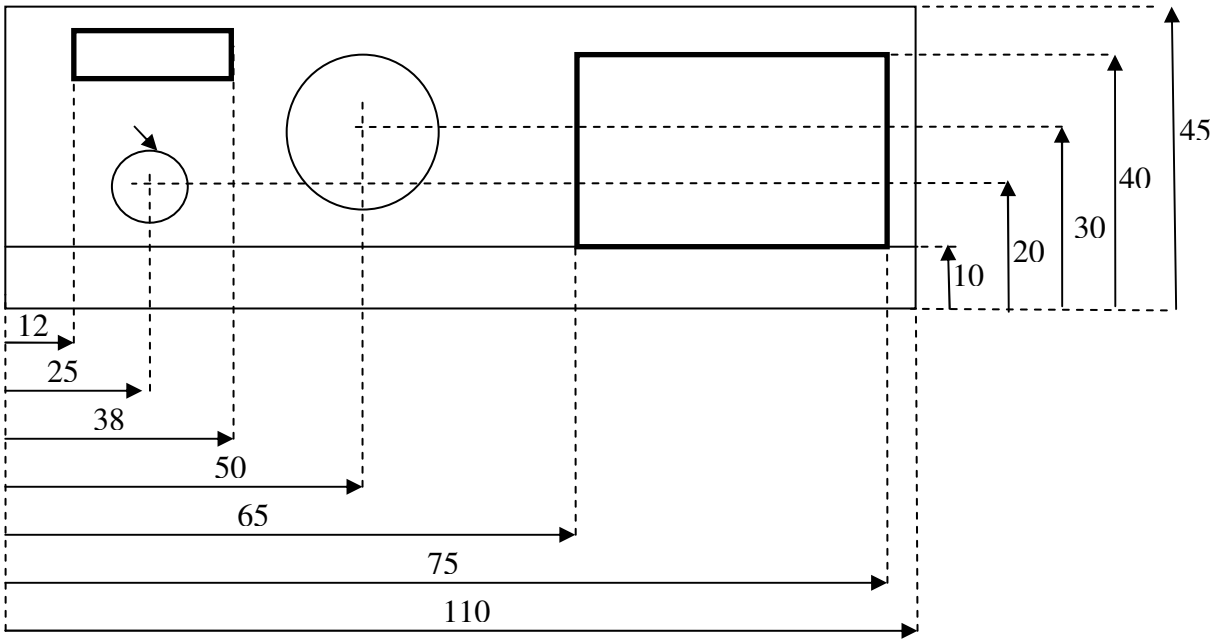
LM1117T-5.0	800mA Low-Dropout Regulator For 5.0V Conversion	U2	640700	1
LM1117T-3.3	800mA Low-Dropout Regulator For 3.3V Conversion	U1	640699	1
BC 547B	BC 547B	Q3	610007	1
Diode 1N4148	High Conductance Fast Diode	D4, D5, D9, D10	603016	4
Summer	Summer	LS1	560650	1
Status LED	Status LED	D3	254726	1
LED grün	LED grün	D1, D2	254725	2
LED Blau	LED blau	D6, D7, D8	251638	3
SW-PB	Folientastatur	S4- S19	220872	16
SW-SPST	Single-Pole, Single-Throw Switch	S2	202530	1
SW-SPDT	Miniaturschalter	S1, S3	202374	2
SD-RSMT-2-MQ	SD Card Holder Push-Push	J1	125442	1

4 Gehäuseskizze

4.1 Frontansicht



4.2 Seitenansicht (Rechte Seite)



5 Quellcode mit Kommentaren

5.1 main.c

```
/*****************************************************************************/
/*  main.c                              Letzte Änderung: 06.08.2008          */
/*                                                                           */
/*                                                                           */
/*                          Diplomarbeit 2008                                */
/*                                                                           */
/*                                                                           */
/*  Schule: Höhere Fachschule Uster (HFU)                                    */
/*  Autor: Philippe Villard (PVI)                                            */
/*                                                                           */
/*  Beschreibung:                                                            */
/*  Mit einer 4*4 Matrixtastatur können verschiedene Befehle einem           */
/*  Legic RFID Leser gesendet werden. Es können Beträge geladen und          */
/*  abgewertet werden und die jeweilige Benutzernummer oder der              */
/*  vorhandene Betrag kann gelesen werden. Mit einem akustischen Signal      */
/*  wird ein erfolgreiches Schreiben auf den Datenträger bestätigt.          */
/*                                                                           */
/*  History:                                                                 */
/*  V0.00    15.07.2008    PVI         SW Gerüst erstellt                    */
/*  V0.01    16.07.2008    PVI         Einbindung Modul Nokia_lcd, Test      */
/*  V0.02    16.07.2008    PVI         Einbindung Modul Tastatur, Test       */
/*  V0.03    17.07.2008    PVI         Einbindung Modul Leser, Test          */
/*  V0.04    18.07.2008    PVI         Einbindung Modul beep, Test           */
/*  V0.05    19.07.2008    PVI         Anpassungen Modul Leser, Test         */
/*  V0.06    31.07.2008    PVI         Verbesserung Laden / Abwerten         */
/*  V0.06    01.08.2008    PVI         Div. Kommentare hinzugeführt, Tests   */
/*  V0.07    06.08.2008    PVI         Enum eingefügt                        */
/*                                                                           */
/*****************************************************************************/
#include <MSP430x16x.h>      // Prozessorspezifische Definitionen
#include "Nokia_lcd.h"       // Funktionen für die Nokia LCD-Ansteuerung
#include "Tastatur.h"        // Funktionen für die Tastatur-Ansteuerung
#include "beep.h"            // Funktionen für die Signalgeber-Ansteuerung
#include "Leser.h"           // Funktionen für die Leser-Ansteuerung
#include "Div_Funk.h"        // Diverse Funktionen


// Variablendklarationen globale Variablen
int Rp_betrag=0;             // Variable für den Rappenbetrag
int FunktionKey=0;           // Variable für Funktions Taste (z.B. Laden, ENTER)
int Writebetrag=0;           // Sollen die eingegebenen Zeichen angezeigt werden
enum ESTATE {BNRLesen, BetragLesen, Enter, Clear, Abwerten, Laden, Nothing};
enum ESTATE State;

// Hauptprogramm
void main (void)
{
WDTCTL=WDTPW + WDTHOLD;      // Watchdog ausschalten
Aufstartvorgang();           // Aufstartequenz


for(;;)                      // Endlosschleife
{
Rp_betrag=0;
FunktionKey=0;
LCDClear();
LCDStr (0,0, "BITTE WAEHLEN:");
LCDStr (2,0, "-Laden");
LCDStr (3,0, "-Abwerten");
```

```
LCDStr (4,0, "-Betrag lesen");
LCDStr (5,0, "-BNR lesen");
LCDUpdate();

State=Nothing;
Tastatur(0, &Rp_betrag, &FunktionKey);
// an FunktionKey eine Funktion zuweisen
if (FunktionKey==10)
  {State=BNRLesen;}
if (FunktionKey==11)
  {State=BetragLesen;}
if (FunktionKey==12)
  {State=Enter;}
if (FunktionKey==13)
  {State=Clear;}
if (FunktionKey==14)
  {State=Laden;}
if (FunktionKey==15)
  {State=Abwerten;}

speiloe_Leser();

switch (State)
  {
case BNRLesen:
  BNR_lesen();
  break;

case BetragLesen:
    LCDClear();
    LCDStr (1,0, "Betrag:       ");
    LCDint (2,0,1, Betrag_lesen());
    LCDUpdate();
    warten(300000);
    LCDClear();
    LCDUpdate();
    break;

case Enter:
    if (Rp_betrag==2009)
    {
    gag();
    }
    LCDUpdate();
    break;

case Clear:
    startok_beep();
    Rp_betrag=0;
    FunktionKey=0;
    LCDClear();
    LCDUpdate();
    break;

case Laden:
    LCDClear();
    LCDStr (0,0, "Rappen-Betrag");
    LCDStr (1,0, "eingeben:");
    LCDStr (2,0, "(TAG auflegen)");
    LCDUpdate();
    Rp_betrag=0;
    Tastatur(1, &Rp_betrag, &FunktionKey);
    LCDClear();
    LCDStr (0,0, "Betrag laden?");
    LCDint(2,0,1, Rp_betrag);
    LCDStr (4,0, "(ENTER/CLEAR?)");
    LCDUpdate();
```

```
    Writebetrag=Betrag_lesen()+Rp_betrag;
  if (Writebetrag<=11000 & Writebetrag>=0)
  {
  Tastatur(0, &Rp_betrag, &FunktionKey);

    switch (FunktionKey)
      {
    case 13: // CLEAR gedrückt
        startok_beep();
        Rp_betrag=0;
        FunktionKey=0;
        LCDClear();
        LCDUpdate();
      break;

     case 12: // ENTER gedrückt
        if (Betrag_schreiben(Writebetrag)!=0)
        {
        LCDClear();
        LCDStr (1,0, "Neuer Betrag:");
        LCDint (2,0,1, Writebetrag);
        laden_beep();
        LCDUpdate();
        warten(270000);
        }
        else
        {
        LCDStr (1,0, "TAG weg!");
        LCDUpdate();
        warten(270000);
        }
      break;

        default:
        LCDClear();
        startok_beep();
        LCDUpdate();
        break;
      }
  }
        else
        {
        warten(50000);
        LCDClear();
        LCDStr (4,0, "Betrag wird");
        LCDStr (5,0, "zu gross!");
        LCDUpdate();
        warten(270000);
        }
      break;

case Abwerten:
    LCDClear();
    LCDStr (0,0, "Rappen-Betrag");
    LCDStr (1,0, "eingeben:");
    LCDStr (2,0, "(TAG auflegen)");
    LCDUpdate();
    Rp_betrag=0;
    Tastatur(1, &Rp_betrag, &FunktionKey);
    LCDClear();
    LCDStr (0,0, "Betrag abw.?");
    LCDint(2,0,1, Rp_betrag);
    LCDStr (4,0, "(ENTER/CLEAR?)");
    LCDUpdate();
    Writebetrag=Betrag_lesen()-Rp_betrag;
    if (Writebetrag<=11000 & Writebetrag>=0)
    {
```

```
      Tastatur(0, &Rp_betrag, &FunktionKey);

      switch (FunktionKey)
        {
      case 13: // CLEAR gedrückt
          startok_beep();
          Rp_betrag=0;
          FunktionKey=0;
          LCDClear();
          LCDUpdate();
        break;

     case 12: // ENTER gedrückt
        if (Betrag_schreiben(Writebetrag)!=0)
        {
          LCDClear();
          LCDStr (1,0, "Neuer Betrag:");
          LCDint (2,0,1, Writebetrag);
          abwerten_beep();
          LCDUpdate();
          warten(270000);
        }
        else
        {
          LCDStr (1,0, "TAG weg!");
          LCDUpdate();
          warten(270000);
        }
       break;

    default:
        LCDClear();
        startok_beep();
        LCDUpdate();
       break;
      }
  }
  else
  {
    warten(50000);
    LCDClear();
    LCDStr (4,0, "Betrag wird");
    LCDStr (5,0, "zu klein!");
    LCDUpdate();
    warten(270000);
  }
  break;


  default:                    // Gleich wie clear
        startok_beep();
        Rp_betrag=0;
        FunktionKey=0;
        LCDClear();
        LCDUpdate();
  break;
  }
}
}
```

5.2 leser.c

```
/*****************************************************************************/
/*  Leser.c                              Letzte Änderung: 31.07.2008         */
/*                                                                           */
/*                          Diplomarbeit 2008                                */
/*                                                                           */
/*  Schule: Höhere Fachschule Uster (HFU)                                    */
/*  Autor: Philippe Villard (PVI)                                            */
/*                                                                           */
/*  RS232_INIT: Initialisiert die RS232 Schnittstelle                        */
/*  BNR_lesen: Liest die Benutzernummer eines TAG's                          */
/*  Betrag_lesen: Liest den vorhandenen Betrag eines TAG's                   */
/*  Betrag_schreiben: Schreibt einen Betrag auf einen TAG                    */
/*  init_reader: Synchronisiert mit dem Leser und gibt info auf Display aus  */
/*  speiloe_Leser: Löscht noch vorhandene Daten im Leserspeicher             */
/*                                                                           */
/*  History:                                                                 */
/*  V0.00    05.07.2008    PVI         RS232_INIT, BNR_lesen implementiert   */
/*  V0.01    06.07.2008    PVI         Betrag_lesen impl.                    */
/*  V0.02    13.07.2008    PVI         Betrag_schreiben und init_reader impl.*/
/*  V0.03    31.07.2008    PVI         speiloe_Leser                         */
/*****************************************************************************/
#include  "msp430x16x.h"            // Prozessorspezifische Definitionen
#include  "Nokia_lcd.h"             // Funktionen für die Nokia LCD-Ansteuerung
#include  "Div_Funk.h"              // Diverse Funktionen

char Einzelzeichen;                 // V für einzelne Zeichen (temp.)
int akt_Zeichen=0;                  // V für einzelne Zeichen in String
unsigned char c[20]="";             // String V für Gelesener Datenstring
unsigned char d[20]="";             // String V für Gelesener Datenstring


/*****************************************************************************/
/*  Funktion :  RS232_INIT                                                   */
/*  Die RS232 Schnittstelle wird mit einem Empfangsinterrupt initialisiert.  */
/*  RS232 Parameter: Bits Pro Sekunde: 9600; Datenbits: 8; Parität: Keine;   */
/*  Stopbit: 1; Flussteuerung: Keine                                         */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void RS232_INIT (void)
{
  WDTCTL = WDTPW + WDTHOLD;             // watchdog aus
  P3SEL = 0x30;                         // P3.4 und P3.5 als USART0 TXD/RXD

  ME1 |= UTXE0 + URXE0;                 // TX- und RX-modul Einschalten
  UCTL0 |= CHAR;                        // 8 Datnbits, 1 Stopbit, Keine Parität
  UTCTL0 |= SSEL0;                      // ACLK als UCLK festlegen
  UBR00 = 0x03;                         // 9600 baud (Seite 265 13-6)
  UBR10 = 0x00;                         // USART Baudrate 1
  UMCTL0 = 0x4A;                        // korrektur der division
  UCTL0 &= ~SWRST;                      // USART freigeben
  IE1 |= URXIE0;                        // Enable USART0 RX interrupt
  IFG1 &= ~UTXIFG0;                     // initales interrupt-flag loeschen
}
```

```
/*****************************************************************************/
/*  Funktion :  BNR_lesen                                                    */
/*  Die aktuelle Benutzernummer eines TAG's wird beim Leser angefragt.       */
/*  Ist eine Benutzernummer vorhanden wird diese direkt auf dem Display      */
/*  ausgegeben. Ansonsten kommt eine Fehlermeldung.                          */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void BNR_lesen(void)
{
// Funktionsvariablen
unsigned int BNR_low=0;              // V für 1er
unsigned int BNR_middle1=0;          // V für 16er
unsigned int BNR_middle2=0;          // V für 256er
unsigned int BNR_high=0;             // V für 4096er
int BNR=0;                           // V für BNR
akt_Zeichen=0;                       // auf 1. Zeichen im String zeigen (init)

RS232_INIT();                        // initialisierung RS232
LCDClear();                          // LCD löschen
LCDUpdate();                         // LCD Updaten

// Befehl BNR senden (0x5D 0x30 0x32 0x35 0x46 0x0D)

  TXBUF0 = 0x5D;                     // Befehl BNR Lesen
  while (!(IFG1 & UTXIFG0));
  TXBUF0 = 0x30;                     // Parameter 1.1 BNR lesen
  while (!(IFG1 & UTXIFG0));
  TXBUF0 = 0x32;                     // Parameter 1.2 BNR lesen
  while (!(IFG1 & UTXIFG0));
  TXBUF0 = 0x35;                     // Checksumme Byte 1 BNR lesen
  while (!(IFG1 & UTXIFG0));
  TXBUF0 = 0x46;                     // Checksumme Byte 2 BNR lesen
  while (!(IFG1 & UTXIFG0));
  TXBUF0 = 0x0D;                     // CR

  _BIS_SR(LPM3_bits + GIE);          // ab in den LPM3 mit interrupt ...

// Sobald aus LPM aus Interrupt erwacht
do
{
  Einzelzeichen=RXBUF0;              // Abholen vom Zeichen im Buffer
  if(Einzelzeichen!=0x0d)            // Wenn Einzelzeichen != Schlusszeichen
  {
  c[akt_Zeichen++]=Einzelzeichen;    // in String abfüllen
  _BIS_SR(LPM3_bits + GIE);          // in den LPM3 mit interrupt ...
  }
}
while (Einzelzeichen!=0x0d);         // Solange Einzelzeichen != Schlusszeichen
c[akt_Zeichen--];                    // Auf eine Stelle früher zeigen
c[akt_Zeichen]='\0';                 // Stringende einfügen

if (c[1]==0x30)             // wenn 2. Zeichen 0 ist (= Dat da)
{
if (c[10]<0x3A)             // Prüfen ob Hex wert eine numerische Zahl
{
BNR_low=(c[10]-0x30)*1;     // mit numerischer Zahl (1-9) rechnen
}
else
{
BNR_low=(c[10]-0x37)*1;     // sonst mit Buschstaben rechnen (A-F)
}
```

```
if (c[9]<0x3A)              // Prüfen ob Hex wert eine numerische Zahl
{
BNR_middle1=(c[9]-0x30)*16; // mit numerischer Zahl (1-9) rechnen
}
else
{
BNR_middle1=(c[9]-0x37)*16; // sonst mit Buschstaben rechnen (A-F)
}

if (c[8]<0x3A)              // Prüfen ob Hex wert eine numerische Zahl
{
BNR_middle2=(c[8]-0x30)*256;// mit numerischer Zahl (1-9) rechnen
}
else
{
BNR_middle2=(c[8]-0x37)*256;// sonst mit Buschstaben rechnen (A-F)
}
if (c[7]<0x3A)              // Prüfen ob Hex wert eine numerische Zahl
{
BNR_high=(c[7]-0x30)*4096;  // mit numerischer Zahl (1-9) rechnen
}
else
{
BNR_high=(c[7]-0x37)*4096;  // sonst mit Buschstaben rechnen (A-F)
}
BNR=0;
BNR=BNR_low+BNR_middle1+BNR_middle2+BNR_high;     // BNR zusammenrechnen
LCDStr (1,0, "BNR:");                             // Anzeige auf Display
LCDint (2,0,0, BNR);                              // Anzeige BNR 3. Zeile
LCDUpdate();                                      // LCD Update
}
else                        // Wenn 5. Zeichen etwas anderes als nicht D
{
LCDStr (1,0, "Kein TAG da...");                   // Fehler ausgeben
LCDUpdate();                                      //  LCD Update
}
warten(300000);           // Warten
LCDClear();               // LCD löschen
LCDUpdate();              // LCD Updaten
}


/*****************************************************************************/
/*  Funktion :  Betrag_lesen                                                 */
/*  Der aktuelle Betrag eines TAG's wird beim Leser angefragt.               */
/*  Ist ein Betrag vorhanden wird dieser zurückgegeben.                      */
/*  Ansonsten wird eine Fehlermeldung ausgegeben.                            */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  Betrag                                                */
/*****************************************************************************/
unsigned int Betrag_lesen(void)
{
akt_Zeichen=0;
unsigned int Betrag_low=0;          // V für Betrag 1er
unsigned int Betrag_middle1=0;      // V für Betrag 16er
unsigned int Betrag_middle2=0;      // V für Betrag 256er
unsigned int Betrag_high=0;         // V für Betrag 4096er
unsigned int Betrag=0;              // V für Betrag

RS232_INIT();                       // initialisierung RS232

// Befehl Betrag lesen senden (0x4F 0x34 0x46 0x0D)
  TXBUF0 = 0x4F;                    // Befehl für Betrag lesen
```

```
  while (!(IFG1 & UTXIFG0));
  TXBUF0 = 0x34;                    // Checksumme Byte 1 Betrag lesen
  while (!(IFG1 & UTXIFG0));
  TXBUF0 = 0x46;                    // Checksumme Byte 2 Betrag lesen
  while (!(IFG1 & UTXIFG0));
  TXBUF0 = 0x0D;                    // CR

  _BIS_SR(LPM3_bits + GIE);           // in den LPM3 mit interrupt ...

// Sobald aus LPM aus Interrupt erwacht
do
{
  Einzelzeichen=RXBUF0;               // Abholen vom Zeichen im Buffer
  if(Einzelzeichen!=0x0d)             // Wenn Einzelzeichen != Schlusszeichen
  {
  c[akt_Zeichen++]=Einzelzeichen;     // in String abfüllen
  _BIS_SR(LPM3_bits + GIE);           // in den LPM3 mit interrupt ...
  }
}
while (Einzelzeichen!=0x0d);          // Solange Einzelzeichen != Schlusszeichen
c[akt_Zeichen--];                     // Auf eine Stelle früher zeigen
c[akt_Zeichen]='\0';                  // Stringende einfügen

if (c[1]==0x30)                   // wenn 1. Zeichen nicht Dat verschwunden
{
if (c[6]<0x3A)                    // Prüfen ob Hex wert eine numerische Zahl
{
Betrag_low=(c[6]-0x30)*1;         // mit numerischer Zahl (1-9) rechnen
}
else
{
Betrag_low=(c[6]-0x37)*1;         // sonst mit Buschstaben rechnen (A-F)
}

if (c[5]<0x3A)
{
Betrag_middle1=(c[5]-0x30)*16;    // mit numerischer Zahl (1-9) rechnen
}
else
{
Betrag_middle1=(c[5]-0x37)*16;    // sonst mit Buschstaben rechnen (A-F)
}

if (c[4]<0x3A)
{
Betrag_middle2=(c[4]-0x30)*256;   // mit numerischer Zahl (1-9) rechnen
}
else
{
Betrag_middle2=(c[4]-0x37)*256;   // sonst mit Buschstaben rechnen (A-F)
}

if (c[3]<0x3A)
{
Betrag_high=(c[3]-0x30)*4096;     // mit numerischer Zahl (1-9) rechnen
}
else
{
Betrag_high=(c[3]-0x37)*4096;     // sonst mit Buschstaben rechnen (A-F)
}

Betrag=Betrag_low+Betrag_middle1+Betrag_middle2+Betrag_high; // Betrag rechnen
}
else                              // Wenn 5. Zeichen etwas anderes als nicht D
{
LCDStr (5,0, "Kein TAG da..."); // Fehler ausgeben
```

```
LCDUpdate();                    // LCD Update
}
return (Betrag);                // Betrag zurückgeben
}


/*****************************************************************************/
/*  Funktion :  Betrag_schreiben                                             */
/*  Ein der Funktion übergebener Parameter Ladebetrag wird auf den           */
/*  TAG geschrieben. Bevor diese Funktion aufgerufen wird, muss ein          */
/*  TAG im Feld sein (kann z.B. mit betrag_lesen geprüft werden)             */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  Betrag                                                */
/*                                                                           */
/*          Output  :  Errorflag (Dat verschwunden) 1=OK, 0=Fehler           */
/*****************************************************************************/
int Betrag_schreiben(unsigned int Ladebetrag)
{
int chks=0;                             // V für CRC (Cecksumme)
int _4096er=0;                          // V für 4096er
int _256er=0;                           // V für 256er
int _16er=0;                            // V für 16er
int _1er=0;                             // V für 1er
akt_Zeichen=0;                          // auf 1. Zeichen im String zeigen (init)

RS232_INIT();                           // initialisierung RS232
LCDClear();                             // LCD löschen
LCDUpdate();                            // LCD updaten

// Befehl Betrag lesen senden (0x4F 0x34 0x46 0x0D)
  TXBUF0 = 0x4F;                        // Befehl für Betrag lesen
  while (!(IFG1 & UTXIFG0));
  TXBUF0 = 0x34;                        // Checksumme Byte 1 Betrag lesen
  while (!(IFG1 & UTXIFG0));
  TXBUF0 = 0x46;                        // Checksumme Byte 2 Betrag lesen
  while (!(IFG1 & UTXIFG0));
  TXBUF0 = 0x0D;                        // CR

  _BIS_SR(LPM3_bits + GIE);             // in den LPM3 mit interrupt...

// Sobald aus LPM aus Interrupt erwacht
// Befehl in String abfüllen
c[0]=0x4E;                              // Befehl
c[1]=0x46;                              // Kennung (egal)
c[2]=0x46;                              // Kennung (egal)
c[3]=0x46;                              // Kennung (egal)
c[4]=0x46;                              // Kennung (egal)
c[5]=0x30;                              // Betrag
c[6]=0x30;                              // Betrag

// int Ladebetrag in String abfüllen
_4096er=Ladebetrag/0x1000;
_256er=(Ladebetrag-(_4096er*0x1000))/0x100;
_16er=(Ladebetrag-(_4096er*0x1000)-(_256er*0x100))/0x10;
_1er=(Ladebetrag-(_4096er*0x1000)-(_256er*0x100)-(_16er*0x10))/0x01;

if (_4096er<10)                         // Wenn eine Ziffer (0-9)
{
c[7]=_4096er+0x30;                      // Ziffer in ASCII umrechnen
}
else                                    // sonst (wenn A-F)
{
c[7]=_4096er+0x37;                      // Buchstabe in ASCII umrechnen
}
```

```
if(_256er<10)                          // Wenn eine Ziffer (0-9)
{
c[8]=_256er+0x30;                      // Ziffer in ASCII umrechnen
}
else
{
c[8]=_256er+0x37;                      // Buchstabe in ASCII umrechnen
}

if (_16er<10)                          // Wenn eine Ziffer (0-9)
{
c[9]=_16er+0x30;                       // Ziffer in ASCII umrechnen
}
else
{
c[9]=_16er+0x37;                      // Buchstabe in ASCII umrechnen
}

if (_1er<10)                           // Wenn eine Ziffer (0-9)
{
c[10]=_1er+0x30;                       // Ziffer in ASCII umrechnen
}
else
{
c[10]=_1er+0x37;                       // Buchstabe in ASCII umrechnen
}

// CRC (Checksumme) berechnen:
while (akt_Zeichen<11)                 // Solange Zeichen nicht an 12. Stelle
{
chks ^=c[akt_Zeichen] ;                // XOR Verknüpfung
akt_Zeichen++;                         // Auf nächstes Zeichen im String gehen
}
c[11]=(chks/16)+0x30;                 // 1. Byte Checksumme abfüllen in String
if((chks%16)<10)                      // Wenn eine Zahl (0-9)
{
c[12]=(chks%16)+0x30;                 // 2. Byte Checksumme abfüllen in String
}
else                                  // Sonst (wenn keine Zahl)
{
c[12]=(chks%16)+0x37;                 // 2. Byte Checksumme abfüllen in String
}

c[13]=0x0d;                           // CR

akt_Zeichen=0;                        // auf 1. Zeichen im String zeigen (init)

// Befehl Betrag schreiben senden (0x4E 0xFF 0xFF 0xXX 0xXX 0xXX 0xCS 0xCS 0x0D)
  while (akt_Zeichen<14)
  {
  TXBUF0 = c[akt_Zeichen];
  while (!(IFG1 & UTXIFG0));
  akt_Zeichen++;
  }

  _BIS_SR(LPM3_bits + GIE);           // ab in den LPM3 mit interrupt ...

// Sobald aus LPM aus Interrupt erwacht
akt_Zeichen=0;                        // auf 1. Zeichen im String zeigen (init)
// Antwort des Lesers in String abfüllen
do
{
  Einzelzeichen=RXBUF0;               // Abholen vom Zeichen im Buffer
  if(Einzelzeichen!=0x0d)             // Wenn Einzelzeichen != Schlusszeichen
  {
  d[akt_Zeichen++]=Einzelzeichen;     // in String abfüllen
```

```
  _BIS_SR(LPM3_bits + GIE);           // in den LPM3 mit interrupt ...
  }
}
while (Einzelzeichen!=0x0d);          // Solange Einzelzeichen != Schlusszeichen
d[akt_Zeichen--];                     // Auf eine Stelle früher zeigen
d[akt_Zeichen]='\0';                  // Stringende einfügen

// Wenn zweites Zeichen =4 (Betr. nicht erfolgreich geschrieben)
if (d[1]=='4')
{
d[1]=0;                               // Variable wieder zurückstellen
return(0);                            // ERROR (z.B. Dat weg)
}
else
{
d[1]=0;                               // Variable wieder zurückstellen
return(1);                            //OK
}
}


/*****************************************************************************/
/*  Funktion :  init_Leser                                                   */
/*  Diese Funktion sendet dem Leser den Synchronisationsbefehl. Der Leser    */
/*  antwortet, wenn er bereit ist, mit seiner SW Release Nummer              */
/*  (z.B. G20040204).                                                        */
/*  Diese wird am Display ausgegeben. Kommt keine Meldung zurück, so wird    */
/*  eine Fehlermeldung ausgegeben.                                           */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void init_Leser(void)
{
unsigned char boot[30]="LESER STARTET  "; // String-Variable für Anzeige
akt_Zeichen=0;                        // Zähler auf 0 stellen
RS232_INIT();                         // initialisierung RS232
LCDClear();                           // LCD löschen
LCDUpdate();                          // LCD Updaten
LCDStr ( 0,0, "Bitte warten..");      // Ausgabe auf 1. Zeile
LCDUpdate();                          // LCD Update

while(akt_Zeichen<14)                 // Solange Zähler kleiner als 14
{
  warten(15000);                      // Warteschlaufe
  boot[akt_Zeichen+14]='-';           // - in String schriben
  LCDStr ( 4,0, boot);                // Ausgabe des Strings
  akt_Zeichen++;                      // Zähler erhöhen
  LCDUpdate();                        // LCD Updaten
}

// Befehl Synchronisieren senden (0x47 0x34 0x37 0x0D)
  TXBUF0 = 0x47;                      // Befehl für Betrag lesen
  while (!(IFG1 & UTXIFG0));
  TXBUF0 = 0x34;                      // Checksumme Byte 1 Betrag lesen
  while (!(IFG1 & UTXIFG0));
  TXBUF0 = 0x37;                      // Checksumme Byte 2 Betrag lesen
  while (!(IFG1 & UTXIFG0));
  TXBUF0 = 0x0D;                      // cr

  _BIS_SR(LPM3_bits + GIE);           // in den LPM3 mit interrupt ...

// Sobald aus LPM durch Interrupt erwacht:
  akt_Zeichen=0;
```

```
// Empfangene Zeichen in String abfüllen
do
{
  Einzelzeichen=RXBUF0;                 // Abholen vom Zeichen im Buffer
  if(Einzelzeichen!=0x0d)               // Wenn Einzelzeichen != Schlusszeichen
  {
  c[akt_Zeichen++]=Einzelzeichen;       // in String abfüllen
  _BIS_SR(LPM3_bits + GIE);             // in den LPM3 mit interrupt ...
  }
}
while (Einzelzeichen!=0x0d);            // Solange Einzelzeichen != Schlusszeichen
c[akt_Zeichen--];                       // Auf eine Stelle früher zeigen
c[akt_Zeichen]='\0';                    // Stringende einfügen
if (c[0]=='G')                          // Wenn korrekte Antwort
{
  LCDClear();                           // LCD löschen
  LCDStr ( 4,0, "Starten OK");          // auf 5. Zeile ausgeben
  LCDStr ( 5,0, "SW: ");                // Auf 6. Zeile ausgeben
  LCDStr ( 5,3, c);                     // Auf 6. Zeile 4. Zeichen String ausgeben
}
else
{
LCDClear();                             // LCD löschen
LCDStr ( 4,0, "ERROR Starten");         // auf 5. Zeile ausgeben
LCDStr ( 5,0, c);                       // Ausgabe des Strings (Fehlerstring)
}
LCDUpdate();                            // Update des Displays
warten(170000);                         // Eine Weile warten
LCDClear();                             // LCD löschen
LCDUpdate();                            // LCD Updaten
}
```

```
/*****************************************************************************/
/*  Funktion :  speiloe_Leser                                                */
/*  Diese Funktion sendet dem Leser den Befehl BNR lesen.                    */
/*  Dieser Befehl wird benötigt, da der Leser evtl. noch alte Daten          */
/*  eines zum voraus gebrauchten TAG's im Speicher hat.                      */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void speiloe_Leser(void) // Leser Speicher löschen
{
// Befehl Betrag lesen senden (0x4F 0x34 0x46 0x0D)
  TXBUF0 = 0x4F;                    // Befehl für Betrag lesen
  while (!(IFG1 & UTXIFG0));
  TXBUF0 = 0x34;                    // Checksumme Byte 1 Betrag lesen
  while (!(IFG1 & UTXIFG0));
  TXBUF0 = 0x46;                    // Checksumme Byte 2 Betrag lesen
  while (!(IFG1 & UTXIFG0));
  TXBUF0 = 0x0D;                    // CR

  _BIS_SR(LPM3_bits + GIE);           // in den LPM3 mit interrupt ...

// Sobald aus LPM aus Interrupt erwacht
do
{
  Einzelzeichen=RXBUF0;               // Abholen vom Zeichen im Buffer
  if(Einzelzeichen!=0x0d)             // Wenn Einzelzeichen != Schlusszeichen
  {
  c[akt_Zeichen++]=Einzelzeichen;     // in String abfüllen
  _BIS_SR(LPM3_bits + GIE);           // in den LPM3 mit interrupt ...
  }
}
while (Einzelzeichen!=0x0d);          // Solange Einzelzeichen != Schlusszeichen
c[akt_Zeichen--];                     // Auf eine Stelle früher zeigen
c[akt_Zeichen]='\0';                  // Stringende einfügen
}

// Interrupt: Sobald ein Zeichen empfasngen, aus LPM 3 aufwachen
#pragma vector=UART0RX_VECTOR
__interrupt void usart0_rx (void)
{
  _BIC_SR_IRQ(LPM3_bits);                   // LPM3 bits löschen 0(SR)
}
```

5.3 leser.h

```
/*****************************************************************************/
/*  Leser.h                              Letzte Änderung: 31.07.2008         */
/*                                                                           */
/*                          Diplomarbeit 2008                                */
/*                                                                           */
/*  Schule: Höhere Fachschule Uster (HFU)                                    */
/*  Autor: Philippe Villard (PVI)                                            */
/*                                                                           */
/*  RS232_INIT: Initialisiert die RS232 Schnittstelle                        */
/*  BNR_lesen: Liest die Benutzernummer eines TAG's                          */
/*  Betrag_lesen: Liest den vorhandenen Betrag eines TAG's                   */
/*  Betrag_schreiben: Schreibt einen Betrag auf einen TAG                    */
/*  init_reader: Synchronisiert mit dem Leser und gibt info auf Display aus  */
/*  speiloe_Leser: Löscht noch vorhandene Daten im Leserspeicher             */
/*                                                                           */
/*  History:                                                                 */
/*  V0.00    05.07.2008    PVI         RS232_INIT, BNR_lesen implementiert   */
/*  V0.01    06.07.2008    PVI         Betrag_lesen impl.                    */
/*  V0.02    13.07.2008    PVI         Betrag_schreiben und init_reader impl.*/
/*  V0.03    31.07.2008    PVI         speiloe_Leser                         */
/*****************************************************************************/
#ifndef Leser_h
#define Leser_h

/*****************************************************************************/
/*  Funktion :  RS232_INIT                                                   */
/*  Die RS232 Schnittstelle wird mit einem Empfangsinterrupt initialisiert.  */
/*  RS232 Parameter: Bits Pro Sekunde: 9600; Datenbits: 8; Parität: Keine;   */
/*  Stopbit: 1; Flusssteuerung: Keine                                        */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void RS232_INIT (void);

/*****************************************************************************/
/*  Funktion :  BNR_lesen                                                    */
/*  Die aktuelle Benutzernummer eines TAG's wird beim Leser angefragt.       */
/*  Ist eine Benutzernummer vorhanden wird diese direkt auf dem Display      */
/*  ausgegeben. Ansonsten kommt eine Fehlermeldung.                          */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void BNR_lesen(void);

/*****************************************************************************/
/*  Funktion :  Betrag_lesen                                                 */
/*  Der aktuelle Betrag eines TAG's wird beim Leser angefragt.               */
/*  Ist ein Betrag vorhanden wird dieser zurückgegeben.                      */
/*  Ansonsten wird eine Fehlermeldung ausgegeben.                            */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  Betrag                                                */
/*****************************************************************************/
unsigned int Betrag_lesen(void);
```

```
/*****************************************************************************/
/*  Funktion :  init_Leser                                                   */
/*  Diese Funktion sendet dem Leser den Synchronisationsbefehl. Der Leser    */
/*  antwortet, wenn er bereit ist, mit seiner SW Release Nummer              */
/*  (z.B. G20040204).                                                        */
/*  Diese wird am Display ausgegeben. Kommt keine Meldung zurück, so wird    */
/*  eine Fehlermeldung ausgegeben.                                           */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void init_Leser(void);

/*****************************************************************************/
/*  Funktion :  Betrag_schreiben                                             */
/*  Ein der Funktion übergebener Parameter Ladebetrag wird auf den           */
/*  TAG geschrieben. Bevor diese Funktion aufgerufen wird, muss ein          */
/*  TAG im Feld sein (kann z.B. mit betrag_lesen geprüft werden)             */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  Betrag                                                */
/*                                                                           */
/*          Output  :  Errorflag (Dat verschwunden) 1=OK, 0=Fehler           */
/*****************************************************************************/
int Betrag_schreiben(unsigned int Ladebetrag);

/*****************************************************************************/
/*  Funktion :  speiloe_Leser                                                */
/*  Diese Funktion sendet dem Leser den Befehl BNR lesen.                    */
/*  Dieser Befehl wird benötigt, da der Leser evtl. noch alte Daten          */
/*  eines zum voraus gebrauchten TAG's im Speicher hat.                      */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void speiloe_Leser(void);
#endif // Leser_h
```

5.4 beep.c

```
/*****************************************************************************/
/*  beep.c                              Letzte Änderung: 22.07.2008          */
/*                                                                           */
/*                          Diplomarbeit 2008                                */
/*                                                                           */
/*  Schule: Höhere Fachschule Uster (HFU)                                    */
/*  Autor: Philippe Villard (PVI)                                            */
/*                                                                           */
/*  init_beep: Initialisiert den Port 2                                      */
/*  abwerten_beep: Tonfolge für das Abziehen eines Betrags                   */
/*  laden_beep: Tonfolge für das Laden eines Betrags                         */
/*  startok_beep: Tonfolge für Aufstartvorgang                               */
/*  tastendruck_beep: Tonfolge bei Tastendruck                               */
/*                                                                           */
/*  History:                                                                 */
/*  V0.00    26.06.2008    PVI         init_beep, abwerten_beep implementiert*/
/*  V0.01    27.06.2008    PVI         laden_beep, startok_beep              */
/*  V0.02    31.07.2008    PVI         tastendruck_beep                      */
/*  V0.03    01.08.2008    PVI         Diverse Anpassungen / Kommentare      */
/*****************************************************************************/
#include <msp430x16x.h>               // Prozessorspezifische Definitionen
#include  "Div_Funk.h"                // Diverse Funktionen

/*****************************************************************************/
/*  Funktion :  init_beep                                                    */
/*  Signalgeber und LED orange werden initialisiert (Port 2)                 */
/*                                                                           */
/*      Parameters                                                           */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void init_beep (void)
{
P2DIR=0xFF;              // Port 2 = Output (LED grün)
P2SEL=0x00;              // Verwendung des Pins: 0=I/O Port
P2IE=0x00;               // Interrupt ausgeschaltet
P2OUT=0x00;              // P2 auf 0
}

/*****************************************************************************/
/*  Funktion :  abwerten_beep                                                */
/*  Ein Betrag wird abgezogen (akustisches Signal und LED löscht lange)      */
/*                                                                           */
/*      Parameters                                                           */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void abwerten_beep (void)
{
  init_beep();       // Initialisierung des Port 2
  warten(3000);      // Warten
  P2OUT=P2OUT^0x01; // mit EXOR Verknüpfung nur P2.1 umschalten (LED)
  P2OUT=P2OUT^0x02; // mit EXOR Verknüpfung nur P2.2 umschalten (Beep)
  warten(60000);     // Warten
  P2OUT=P2OUT^0x01; // mit EXOR Verknüpfung nur P2.1 umschalten (LED)
  P2OUT=P2OUT^0x02; // mit EXOR Verknüpfung nur P2.2 umschalten (Beep)
}
```

```
/*****************************************************************************/
/*  Funktion :  laden_beep                                                   */
/*  Ein Betrag wird abgezogen (akustisches Signal und LED leuchtet           */
/*  3 mal auf)                                                               */
/*                                                                           */
/*      Parameters                                                           */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void  laden_beep (void)
{
  init_beep();
  P2OUT=P2OUT^0x01; // mit EXOR Verknüpfung umschalten (LED)
  P2OUT=P2OUT^0x02; // mit EXOR Verknüpfung nur P2.2 umschalten (Beep)
  warten(5000);     // Warten
  P2OUT=P2OUT^0x01; // mit EXOR Verknüpfung nur P2.1 umschalten (Beep)
  P2OUT=P2OUT^0x02; // mit EXOR Verknüpfung nur P2.2 umschalten (Beep)
  warten(5000);     // Warten
  P2OUT=P2OUT^0x01; // mit EXOR Verknüpfung nur P2.1 umschalten (Beep)
  P2OUT=P2OUT^0x02; // mit EXOR Verknüpfung nur P2.2 umschalten (Beep)
  warten(5000);     // Warten
  P2OUT=P2OUT^0x01; // mit EXOR Verknüpfung nur P2.1 umschalten (Beep)
  P2OUT=P2OUT^0x02; // mit EXOR Verknüpfung nur P2.2 umschalten (Beep)
  warten(5000);     // Warten
  P2OUT=P2OUT^0x01; // mit EXOR Verknüpfung nur P2.1 umschalten (Beep)
  P2OUT=P2OUT^0x02; // mit EXOR Verknüpfung nur P2.2 umschalten (Beep)
  warten(5000);
  P2OUT=P2OUT^0x01; // mit EXOR Verknüpfung nur P2.1 umschalten (Beep)
  P2OUT=P2OUT^0x02; // mit EXOR Verknüpfung nur P2.2 umschalten (Beep)
}


/*****************************************************************************/
/*  Funktion :  startok_beep                                                 */
/*  Akustisches Signal, wenn Leser fertig gestartet ist.                     */
/*                                                                           */
/*      Parameters                                                           */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void  startok_beep (void)
{
  init_beep();
  P2OUT=P2OUT^0x01; // mit EXOR Verknüpfung umschalten (LED)
  P2OUT=P2OUT^0x02; // mit EXOR Verknüpfung nur P2.2 umschalten (Beep)
  warten(2000);     // Warten
  P2OUT=P2OUT^0x01; // mit EXOR Verknüpfung nur P2.1 umschalten (Beep)
  P2OUT=P2OUT^0x02; // mit EXOR Verknüpfung nur P2.2 umschalten (Beep)
  warten(20000);     // Warten
  P2OUT=P2OUT^0x01; // mit EXOR Verknüpfung nur P2.1 umschalten (Beep)
  P2OUT=P2OUT^0x02; // mit EXOR Verknüpfung nur P2.2 umschalten (Beep)
  warten(2000);     // Warten
  P2OUT=P2OUT^0x01; // mit EXOR Verknüpfung nur P2.1 umschalten (Beep)
  P2OUT=P2OUT^0x02; // mit EXOR Verknüpfung nur P2.2 umschalten (Beep)
  warten(10000);     // Warten
  P2OUT=P2OUT^0x01; // mit EXOR Verknüpfung nur P2.1 umschalten (Beep)
  P2OUT=P2OUT^0x02; // mit EXOR Verknüpfung nur P2.2 umschalten (Beep)
    warten(2000);     // Warten
  P2OUT=P2OUT^0x01; // mit EXOR Verknüpfung umschalten (LED)
  P2OUT=P2OUT^0x02; // mit EXOR Verknüpfung nur P2.2 umschalten (Beep)
}
```

```
/*****************************************************************************/
/*  Funktion :  tastendruck_beep                                             */
/*  Kurzer Ton wenn Taste gedrückt, Status LED kurz aus                      */
/*                                                                           */
/*      Parameters                                                           */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void  tastendruck_beep (void)
{
  init_beep();      // Initialisierung des Port 2
  warten(1000);     // Warten
  P2OUT=P2OUT^0x01; // mit EXOR Verknüpfung nur P2.1 umschalten (LED)
  P2OUT=P2OUT^0x02; // mit EXOR Verknüpfung nur P2.2 umschalten (Beep)
  warten(1000);     // Warten
  P2OUT=P2OUT^0x01; // mit EXOR Verknüpfung nur P2.1 umschalten (LED)
  P2OUT=P2OUT^0x02; // mit EXOR Verknüpfung nur P2.2 umschalten (Beep)
}
```

5.5 beep.h

```
/*****************************************************************************/
/*  beep.h                              Letzte Änderung: 22.07.2008          */
/*                                                                           */
/*                          Diplomarbeit 2008                                */
/*                                                                           */
/*  Schule: Höhere Fachschule Uster (HFU)                                    */
/*  Autor: Philippe Villard (PVI)                                            */
/*                                                                           */
/*  init_beep: Initialisiert den Port 2                                      */
/*  abwerten_beep: Tonfolge für das Abziehen eines Betrags                   */
/*  laden_beep: Tonfolge für das Laden eines Betrags                         */
/*  startok_beep: Tonfolge für Aufstartvorgang                               */
/*  tastendruck_beep: Tonfolge bei Tastendruck                               */
/*                                                                           */
/*  History:                                                                 */
/*  V0.00    26.06.2008    PVI         init_beep, abwerten_beep implementiert*/
/*  V0.01    27.06.2008    PVI         laden_beep, startok_beep              */
/*  V0.02    31.07.2008    PVI         tastendruck_beep                      */
/*  V0.03    01.08.2008    PVI         Diverse Anpassungen / Kommentare      */
/*****************************************************************************/
#ifndef beep_h
#define beep_h

/*****************************************************************************/
/*  Funktion :  init_beep                                                    */
/*  Signalgeber und LED orange werden initialisiert (Port 2)                 */
/*                                                                           */
/*      Parameters                                                           */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void  init_beep (void);
```

```
/*****************************************************************************/
/*  Funktion :  abwerten_beep                                                */
/*  Ein Betrag wird abgezogen (akustisches Signal und LED löscht lange)      */
/*                                                                           */
/*      Parameters                                                           */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void  abwerten_beep (void);

/*****************************************************************************/
/*  Funktion :  laden_beep                                                   */
/*  Ein Betrag wird abgezogen (akustisches Signal und LED leuchtet           */
/*  3 mal auf)                                                               */
/*                                                                           */
/*      Parameters                                                           */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void  laden_beep(void);

/*****************************************************************************/
/*  Funktion :  startok_beep                                                 */
/*  Akustisches Signal, wenn Leser fertig gestartet ist.                     */
/*                                                                           */
/*      Parameters                                                           */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void  startok_beep (void);

/*****************************************************************************/
/*  Funktion :  tastendruck_beep                                             */
/*  Kurzer Ton wenn Taste gedrückt, Status LED kurz aus                      */
/*                                                                           */
/*      Parameters                                                           */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void  tastendruck_beep (void);

#endif // beep_h
```

5.6 Tastatur.c

```
/*****************************************************************************/
/*  Tastatur.c                          Letzte Änderung: 31.07.2008          */
/*                                                                           */
/*                          Diplomarbeit 2008                                */
/*                                                                           */
/*  Schule: Höhere Fachschule Uster (HFU)                                    */
/*  Autor: Philippe Villard (PVI)                                            */
/*                                                                           */
/*  Tastatur: Fragt permanent die Tastatur ab und gibt einen evtl. ein-      */
/*  gegebenen Betrag und den gedrückten FunktionKey (LADEN, ABWERTEN,        */
/*  ENTER, CLEAR, BNR Lesen und Betrag lesen) zurück.                        */
/*  Sobald ein FunctionKey gedrückt wird, werden die Werte zurück gegeben.   */
/*  Ein Betrag muss also vor dem Drücken eines FunktionKey                   */
/*  eingegeben werden. Sollen die einzelnen Tastendrück sichtbar sein,       */
/*  so muss Z_Ausgabe=1 sein.                                                */
/*                                                                           */
/*  History:                                                                 */
/*  V0.00    26.06.2008    PVI         Erste Tests mit der Tastatur          */
/*  V0.01    19.06.2008    PVI         Diverse Anpassungen                   */
/*  V0.02    20.07.2008    PVI         Diverse Anpassungen / Kommentare      */
/*  V0.02    31.07.2008    PVI         Diverse kleineAnpassungen / Kommentare */
/*                                                                           */
/*****************************************************************************/
#include <MSP430x16x.h>               // Processor specific definitions
#include "Nokia_lcd.h"                // Funktionen für die Nokia LCD-Ansteuerung
#include "Div_Funk.h"                 // Diverse Funktionen
#include "beep.h"                     // Funktionen für die Signalgeber-Ansteuerung

/*****************************************************************************/
/*  Function :  Tastatur                                                     */
/*  Fragt permanent die Tastatur ab und gibt einen evtl. ein-                */
/*  gegebenen Betrag und den gedrückten FunktionKey ( LADEN, ABWERTEN,       */
/*  ENTER, CLEAR, BNR Lesen und Betrag lesen) zurück.                        */
/*  Sobald ein FunctionKey gedrückt wird, werden die Werte zurück gegeben.   */
/*  Ein Betrag muss also vor dem Drücken eines FunktionKey                   */
/*  eingegeben werden. Sollen die einzelnen Tastendrücke sichtbar sein,       */
/*  so muss Z_Ausgabe=1 sein.                                                */
/*                                                                           */
/*      Parameter                                                            */
/*          Input   :  Z_Ausgabe                                             */
/*                                                                           */
/*          Output  :  int*Rp_betrag, int *FunktionKey                       */
/*****************************************************************************/
void Tastatur(int Z_Ausgabe, int *Rp_betrag, int *FunktionKey)
{
int Zeichen=0;                              // Zuweisung des Zeichens
int Port1=0;                                // "eingefangenes" Zeichen
int Betragseingabe [6]={0,0,0,0,0,0};       // Einzelne Tastendrücke
int countBE=0;                              // Zaehler Ziffern der Betragseingabe
int Entprellung=0;                          // V für Entprellung
int Zeile=0;                                // V für Zeile der Tastatur
int Einzzeichenausgabe=0;                   // V für Einzelzeichenausgabe

P1DIR=0x00;             // P1 = Input
P1SEL=0x00;             // Verwendung des Pins: 0=I/O Port
P1IE=0x00;              // Interrupt ausgeschaltet

P4DIR=0xFF;             // P4 = Output
P4SEL=0x00;             // Verwendung des Pins: 0=I/O Port
P4OUT=0xFF;             // Alle P4 auf 1

while(Zeichen<10)       // Solange kein FunktionKey gedrückt
```

```
{
Port1=0;                // Variable Port1 auf 0 setzen
Zeichen=0;              // Variable Zeichen auf 0 setzen

// Entprellvorang: Solange nicht X mal hintereinander die gleiche Taste
// gedrückt diese Schlaufe durchführen
for (Entprellung=0; Entprellung<3; Entprellung++) // 9 ext Speisung, 110 ohne...
{
while (P1IN==0x00)      // solange an P1 nichts kommt
{
  //P2OUT=0x02;         // Summer=aktiv
}
Port1=P1IN;             // Einlesen des P1 in Port1 variable
warten(30);
  if (Port1==Zeile)     // Wenn Port1 gleich ist wie Zeile
  {
  Entprellung++;        // Entprellung inkrementieren
  }
  else                  // sonst
  {
  Entprellung=0;        // Entprellung wider 0 setzen (von vorne beginnen)
  Zeile=Port1;          // Zeile ist Port1 (1-4)
  }
}

// Springe zur gedrückten Zeile
  switch (Port1)
  {
    case 1: // eine Taste in der 4. Zeile wurde gedrückt

    P4OUT=1;
    warten(30);
    while (P1IN==0x01){Zeichen=10;}  // BNR lesen wurde gedrückt
    P4OUT=2;
    warten(9);
    while (P1IN==0x01){Zeichen=0;}  // 0 wurde gedrückt
    P4OUT=4;
    warten(9);
    while (P1IN==0x01){Zeichen=11;} // Betrag lesen wurde gedrückt
    P4OUT=8;
    warten(9);
    while (P1IN==0x01){Zeichen=12;}  // ENTER wurde gedrückt
    break;

   case 2: // eine Taste in der 3. Zeile wurde gedrückt

    P4OUT=1;
    warten(30);
    while (P1IN==0x02){Zeichen=7;}  // 7 wurde gedrückt
    P4OUT=2;
    warten(9);
    while (P1IN==0x02){Zeichen=8;}  // 8 wurde gedrückt
    P4OUT=4;
    warten(9);
    while (P1IN==0x02){Zeichen=9;}  // 9 wurde gedrückt
    P4OUT=8;
    warten(9);
    while (P1IN==0x02){Zeichen=13;} // Clear wurde gedrückt
    break;

   case 4: // eine Taste in der 1. Zeile wurde gedrückt

    P4OUT=1;
    warten(30);
    while (P1IN==0x04){Zeichen=1;}  // 1 wurde gedrückt
    P4OUT=2;
```

```
    warten(9);
    while (P1IN==0x04){Zeichen=2;}  // 2 wurde gedrückt
    P4OUT=4;
    warten(9);
    while (P1IN==0x04){Zeichen=3;}  // 3 wurde gedrückt
    P4OUT=8;
    warten(9);
    while (P1IN==0x04){Zeichen=14;} // Laden wurde gedrückt
    break;

   case 8: // eine Taste in der 2. Zeile wurde gedrückt

    P4OUT=1;
    warten(30);
    while (P1IN==0x08){Zeichen=4;}  // 4 wurde gedrückt
    P4OUT=2;
    warten(9);
    while (P1IN==0x08){Zeichen=5;}  // 5 wurde gedrückt
    P4OUT=4;
    warten(9);
    while (P1IN==0x08){Zeichen=6;}  // 6 wurde gedrückt
    P4OUT=8;
    warten(9);
    while (P1IN==0x08){Zeichen=15;} // Abwerten wurde gedrückt
    break;

    default: LCDStr ( 5,0, "ERROR_TASTATUR");
    break;
  }

if (Z_Ausgabe==1)                   // Wenn Zeichen jeweils auf LCD sollen
{
LCDEinzZeich (4, Einzzeichenausgabe, Zeichen);
Einzzeichenausgabe++;
LCDUpdate();
}
tastendruck_beep();                 // Pipsen, wenn Taste gedrückt

if (Zeichen<10)                     // Wenn  eine normale Zifer gedrückt (0-9)
{
Betragseingabe[countBE]=Zeichen;    // in aktuelles Arrayindex Zeichn schreiben
countBE++;                          // Arrayindex erhöhen
}

if (countBE>5)                      // Prüfung, ob mehr als 5 Ziffern gedrückt
{
*Rp_betrag=0;                       // Rappenbetrag auf 0 stellen
countBE=0;                          // Zähler Ziffern auf 0 stellen
LCDStr ( 5,0, "ERROR_EINGABE");     // Anzeige auf Display
LCDUpdate();                        // LCD Updaten
warten(200000);                     // Warten
}
}


*FunktionKey=Zeichen;               // Zeichen in Funktion Key zurückgeben


// Berechung / Zusammensetzung des Rappenbetrags
if ((countBE-1)>=0){*Rp_betrag=*Rp_betrag+(Betragseingabe[countBE-1]*1);}
if ((countBE-2)>=0){*Rp_betrag=*Rp_betrag+(Betragseingabe[countBE-2]*10);}
if ((countBE-3)>=0){*Rp_betrag=*Rp_betrag+(Betragseingabe[countBE-3]*100);}
if ((countBE-4)>=0){*Rp_betrag=*Rp_betrag+(Betragseingabe[countBE-4]*1000);}
if ((countBE-5)>=0){*Rp_betrag=*Rp_betrag+(Betragseingabe[countBE-5]*10000);}
}
```

5.7 Tastatur.h

```
/*****************************************************************************/
/*  Tastatur.h                          Letzte Änderung: 31.07.2008          */
/*                                                                           */
/*                          Diplomarbeit 2008                                */
/*                                                                           */
/*  Schule: Höhere Fachschule Uster (HFU)                                    */
/*  Autor: Philippe Villard (PVI)                                            */
/*                                                                           */
/*  Tastatur: Fragt permanent die Tastatur ab und gibt einen evtl. ein-      */
/*  gegebenen Betrag und den gedrückten FunktionKey (LADEN, ABWERTEN,        */
/*  ENTER, CLEAR, BNR Lesen und Betrag lesen) zurück.                        */
/*  Sobald ein FunctionKey gedrückt wird, werden die Werte zurück gegeben.   */
/*  Ein Betrag muss also vor dem Drücken eines FunktionKey                   */
/*  eingegeben werden. Sollen die einzelnen Tastendrück sichtbar sein,       */
/*  so muss Z_Ausgabe=1 sein.                                                */
/*                                                                           */
/*  History:                                                                 */
/*  V0.00    26.06.2008    PVI         Erste Tests mit der Tastatur          */
/*  V0.01    19.06.2008    PVI         Diverse Anpassungen                   */
/*  V0.02    20.07.2008    PVI         Diverse Anpassungen / Kommentare      */
/*  V0.02    31.07.2008    PVI         Diverse kleine Anpassungen / Kommentare*/
/*                                                                           */
/*****************************************************************************/
#ifndef Tastatur_h
#define Tastatur_h

/*****************************************************************************/
/*  Function :  Tastatur                                                     */
/*  Fragt permanent die Tastatur ab und gibt einen evtl. ein-                */
/*  gegebenen Betrag und den gedrückten FunktionKey ( LADEN, ABWERTEN,       */
/*  ENTER, CLEAR, BNR Lesen und Betrag lesen) zurück.                        */
/*  Sobald ein FunctionKey gedrückt wird, werden die Werte zurück gegeben.   */
/*  Ein Betrag muss also vor dem Drücken eines FunktionKey                   */
/*  eingegeben werden. Sollen die einzelnen Tastendrück sichtbar sein,       */
/*  so muss Z_Ausgabe=1 sein.                                                */
/*                                                                           */
/*      Parameter                                                            */
/*          Input   :  Z_Ausgabe                                             */
/*                                                                           */
/*          Output  :  int*Rp_betrag, int *FunktionKey                       */
/*****************************************************************************/
void Tastatur(int Z_Ausgabe, int *Rp_betrag, int *FunktionKey);

#endif // Tastatur_h
```

5.8 Nokia_lcd.c

```
/*****************************************************************************/
/*  Nokia_lcd.c                          Letzte Änderung: 26.07.2008         */
/*                                                                           */
/*                         Diplomarbeit 2008                                 */
/*                                                                           */
/*  Schule: Höhere Fachschule Uster (HFU)                                    */
/*  Autor: Philippe Villard (PVI)                                            */
/*                                                                           */
/*  Funktionen für die Ansteuerung eines Nokia 3310  Displays                */
/*  (mit PCD8544 48 * 84 pixels matrix LCD controller von Philips)           */
/*                                                                           */
/*  LCDInit: Initiapisiert UART1 für SPI Betrieb                             */
/*  LCDSend: Senden eines Zeichens                                           */
/*  LCDUpdate: Aktualisierung des Displayinhalts                             */
/*  LCDClear: löscht gesammten Displayinhalt                                 */
/*  LCDPixel: ein einzelnes Pixel auf Displayausgeben                        */
/*  LCDChrXY: Aufbereiten eines Zeichens                                     */
/*                                                                           */
/*                                                                           */
/*  History:                                                                 */
/*  V0.00    08.06.2008    PVI         Erste Tests mit LCD Display          */
/*  V0.01    15.06.2008    PVI         Diverse Funktionen implementiert     */
/*  V0.02    14.06.2008    PVI         Diverse Anpassungen / Funktionen     */
/*  V0.03    20.07.2008    PVI         Diverse Anpassungen / Kommentare     */
/*****************************************************************************/

#include <io430x16x.h>                // Prozessorspezifische Definitionen
#include "Nokia_lcd.h"                // für Lookuptable
#include "Div_Funk.h"                 // Diverse Funktionen

// LCD memory index
unsigned int  LcdMemIdx;

// LCD matrix
unsigned char  LcdMemory[LCD_CACHE_SIZE];


/*****************************************************************************/
/*  Funktion :  LCD_Init                                                     */
/*  Initialisiert die SPI Schnittstelle für das LCD Display                  */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void LCDInit(void)
{

  //  Pull-up auf reset Pin
  P5OUT_bit.P5OUT_4 = 1;

  // Pin Konfiguration (alles outputs)
  P5DIR_bit.P5DIR_0 = 1;  // STE0
  P5DIR_bit.P5DIR_1 = 1;  // SIMO0
  P5DIR_bit.P5DIR_2 = 1;  // SOMI0
  P5DIR_bit.P5DIR_3 = 1;  // ULCK0
  P5DIR_bit.P5DIR_4 = 1;  // RES

  // Pin Zuordnung
  P5SEL_bit.P5SEL_0 = 0;  // STE0
  P5SEL_bit.P5SEL_1 = 1;  // SIMO0
  P5SEL_bit.P5SEL_2 = 0;  // SOMI0 (D/S)
```

```
  P5SEL_bit.P5SEL_3 = 1;  // ULCK


  //  Toggeln des display Resetpin
  P5OUT_bit.P5OUT_4 = 0;
  warten(10000);
  P5OUT_bit.P5OUT_4 = 1;

  // Init SPI (S292 in User's Guide)
  U1CTL   = 0x16;   // SPI Mode, 8bit, Master mode
  U1TCTL  = 0xB2;   // 3pin Mode, clock->SMCLK, kein CKPL (poliarity), kein CKPH

  U1BR0   = 0x02; // Baudrate 14-7 in User Guide
  U1BR1   = 0x00; // 14-17 in User Guide
  UMCTL1  = 0x00; // nicht benötigt in SPI mode

  ME2     = 0x10;   // Enable SPI1=0x10 (für SPI0:0x01)

  // Display Kontroller Ausschalten
  P5OUT_bit.P5OUT_0 = 1;  // STE0

  warten(100);

  // LCD Komandos senden (PCD8544 S14 Instruction set)
  LCDSend( 0x21, SEND_CMD );  // LCD Extended Commands.
  LCDSend( 0xC8, SEND_CMD );  // Set LCD Vop (Contrast).
  LCDSend( 0x06, SEND_CMD );  // Set Temp coefficent.
  LCDSend( 0x13, SEND_CMD );  // LCD bias mode 1:48 (seite 16)
  LCDSend( 0x20, SEND_CMD );  // LCD Standard, horizontaler Addressierungsmodus
  LCDSend( 0x08, SEND_CMD );  // LCD blank
  LCDSend( 0x0C, SEND_CMD );  // LCD in normal mode stellen

  // Clear und Update
  LCDClear();
  LCDUpdate();
}


/*****************************************************************************/
/*  Funktion :  LCDSend                                                      */
/*  Sendet ein Zeichen ans LCD Display                                       */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  data und  SEND_CHR oder SEND_CMD                      */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void LCDSend(unsigned char data, unsigned char cd)
{

  // Display Kontroller einschalten (active low).
  P5OUT_bit.P5OUT_0 = 0;  // STE0 (CE)

  // Kommando oder Daten
  if(cd == SEND_CHR)
  {
    P5OUT_bit.P5OUT_2 = 1;  // SOMI0 (D/S)
  }
  else
  {
    P5OUT_bit.P5OUT_2 = 0;                  // SOMI0 (D/S)
  }

// Auf SPI geben
  U1TXBUF = data;                         // Daten senden
  while((U1TCTL & TXEPT) == 0);           //Warten auf U0TXBUF bereit
```

```
  // Display Kontroller ausschalten (active low).
  P5OUT_bit.P5OUT_0 = 1;                    // STE0 (CE)

}


/*****************************************************************************/
/*  Funktion :  LCDUpdate                                                    */
/*  Updatet den Display Speicher                                             */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input    :  -                                                    */
/*                                                                           */
/*          Output   :  -                                                    */
/*****************************************************************************/
void LCDUpdate ( void )
{
  int i;

  //  Setze Basisadresse X=0 Y=0
  LCDSend(0x80, SEND_CMD );
  LCDSend(0x40, SEND_CMD );

  //  Buffer senden
  for (i=0; i<LCD_CACHE_SIZE; i++)
  {
    LCDSend( LcdMemory[i], SEND_CHR );
  }
}

/*****************************************************************************/
/*  Funktion :  LCDClear                                                     */
/*  Löscht das LCD Display                                                   */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input    :  -                                                    */
/*                                                                           */
/*          Output   :  -                                                    */
/*****************************************************************************/
void LCDClear(void)
{
  int i;

  // Alles auf 0 setzen
  for (i=0; i<LCD_CACHE_SIZE; i++)
  {
     LcdMemory[i] = 0;
  }

}


/*****************************************************************************/
/*  Funktion :  LCDPixel                                                     */
/*  Einzelnes Pixel kann auf Display angezeigt werden                        */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input    :  x-Koordinate, y- Koordinate, mode                    */
/*                                                                           */
/*          Output   :  -                                                    */
/*****************************************************************************/
void LCDPixel (unsigned char x, unsigned char y, unsigned char mode )
{
    unsigned int    index   = 0;
    unsigned char   offset  = 0;
    unsigned char   data    = 0;
```

```
    // Prüfen ob inerhalb des Displaybereichs
    if ( x > LCD_X_RES )
    {
      return;
    }
    if ( y > LCD_Y_RES )
    {
      return;
    }

    index = ((y / 8) * 84) + x;
    offset  = y - ((y / 8) * 8);

    data = LcdMemory[index];

    if ( mode == PIXEL_OFF )
    {
        data &= (~(0x01 << offset));
    }
    else if ( mode == PIXEL_ON )
    {
        data |= (0x01 << offset);
    }
    else if ( mode  == PIXEL_XOR )
    {
        data ^= (0x01 << offset);
    }

    LcdMemory[index] = data;

}


/*************************************************************************/
/*  Funktion :  LCDChrXY                                                 */
/*  Einzelnes Zeichen auf Display ausgeben                               */
/*                                                                       */
/*         Parameter:                                                    */
/*          Input   :  x-Koordinate, y- Koordinate, mode                 */
/*                                                                       */
/*          Output  :  -                                                 */
/*************************************************************************/
void LCDChrXY (unsigned char x, unsigned char y, unsigned char ch )
{
    unsigned int    index   = 0;
    unsigned int    i       = 0;

    // Prüfen ob inerhalb des Displaybereichs
    if ( x > LCD_X_RES )
    {
      return;
    }
    if ( y > LCD_Y_RES )
    {
      return;
    }

    index = (x*48 + y*48*14)/8 ;

    for ( i = 0; i < 5; i++ )
    {
      LcdMemory[index] = FontLookup[ch - 32][i] << 1;
      index++;
    }

}
```

```
/*****************************************************************************/
/*  Funktion :  LCDContrast                                                  */
/*  Kontrast des LCD Displays einstellen                                     */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  contrast                                              */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void LCDContrast(unsigned char contrast)
{
    //  Erweitertes Komando (Function set)
    LCDSend( 0x21, SEND_CMD );

    // LCD Vop setzen (Contrast)
    LCDSend( 0x80 | contrast, SEND_CMD );

    // LCD Standard, horizontaler Addressierungsmodus
    LCDSend( 0x20, SEND_CMD );
}


/*****************************************************************************/
/*  Funktion :  LCDStr                                                       */
/*  String an LCD senden                                                     */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  row (Zeile), sp (Spalte), *dataPtr (String)           */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void LCDStr(unsigned char row, unsigned char sp, unsigned char *dataPtr )
{

  // Schleife bis ans Ende des Strings
  while (*dataPtr)
  {
    LCDChrXY( sp, row, (*dataPtr));
    sp++;
    dataPtr++;
  }
}

/*****************************************************************************/
/*  Funktion :  LCDint                                                       */
/*  Integer an LCD senden                                                    */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  x, y, wertart (mit SFr. oder ohne), betrag            */
/*                     x=Spalte, y=Zeile, betrag=Betrag in Währung*100       */
/*          Output  :  -                                                     */
/*****************************************************************************/
void LCDint (unsigned char y, unsigned char x, int wertart, unsigned int betrag)
{
    unsigned int    xAchse  = 0;      // Variable fuer x Achse
    unsigned int    spalte  = 0;      // Variable fuer welche Spalte
    unsigned int    _10000er= 0;      // Variable fuer 10000er
    unsigned int    _1000er = 0;      // Variable fuer 1000er
    unsigned int    _100er  = 0;      // Variable fuer 100er
    unsigned int    _10er   = 0;      // Variable fuer 10er
    unsigned int    _1er    = 0;      // Variable fuer 1er

    // Pruefung fuer Ausserhalb Display
    if ( x > LCD_X_RES ) return;
    if ( y > LCD_Y_RES ) return;
```

```
    //Umrechung der einzelenen Ziffern in 1000er, 100er etc.
    _10000er=betrag/10000;
    _1000er=(betrag-(_10000er*10000))/1000;
    _100er=(betrag-((_10000er*10000)+(_1000er*1000)))/100;
    _10er=(betrag-((_10000er*10000)+(_1000er*1000)+(_100er*100)))/10;
    _1er=(betrag-((_10000er*10000)+(_1000er*1000)+(_100er*100)+(_10er*10)))/1;

    // Berechnung welche Spalte
    xAchse = (x*48 + y*48*14)/8 ;

    // Ausgabe auf Display, 5 Durchgaenge (siehe FontLookup)
    for ( spalte = 0; spalte < 5; spalte++ )
    {
      LcdMemory[xAchse]     = FontLookup[_10000er+16][spalte] << 1;
      LcdMemory[xAchse+6]   = FontLookup[_1000er+16][spalte] << 1;
      LcdMemory[xAchse+12]  = FontLookup[_100er+16][spalte] << 1;
      LcdMemory[xAchse+21]  = FontLookup[_10er+16][spalte] << 1;
      LcdMemory[xAchse+27]  = FontLookup[_1er+16][spalte] << 1;
      xAchse++;
    }
    if (wertart==1)
    {
    //Ausgabe für Punkt (Trennung Fr./ Rp.)
      LCDPixel(18, 23, 1);
      LCDPixel(19, 23, 1);
    // Waehrungsangabe
      LCDStr ( 2,6, "SFr.");
    }
}


/*************************************************************************/
/*  Funktion :  LCDEinzZeich                                             */
/*  Zeigt eine einzelne int Ziffer an                                    */
/*                                                                       */
/*         Parameter:                                                    */
/*          Input   :  x, y, Ziffer                                      */
/*                     x=Spalte, y=Zeile, Ziffer                         */
/*          Output  :  -                                                 */
/*************************************************************************/
void LCDEinzZeich (unsigned char y, unsigned char x, unsigned int Ziffer)
{
    unsigned int    xAchse  = 0;      // Variable fuer x Achse
    unsigned int    spalte  = 0;      // Variable fuer welche Spalte

    // Pruefung fuer Ausserhalb Display
    if ( x > LCD_X_RES ) return;
    if ( y > LCD_Y_RES ) return;

    // Berechnung welche Spalte
    xAchse = (x*48 + y*48*14)/8 ;

    // Ausgabe auf Display, 5 Durchgaenge (siehe FontLookup)
 for ( spalte = 0; spalte < 5; spalte++ )
    {
      LcdMemory[xAchse]=FontLookup[Ziffer+16][spalte] << 1;//25
      xAchse++;
    }
}
```

5.9 Nokia_lcd.h

```
/*****************************************************************************/
/*  Nokia_lcd.h                          Letzte Änderung: 26.07.2008          */
/*                                                                           */
/*                          Diplomarbeit 2008                                */
/*                                                                           */
/*  Schule: Höhere Fachschule Uster (HFU)                                    */
/*  Autor: Philippe Villard (PVI)                                            */
/*                                                                           */
/*  Funktionen für die Ansteuerung eines Nokia 3310  Displays                */
/*  (mit PCD8544 48 * 84 pixels matrix LCD controller von Philips)           */
/*                                                                           */
/*  LCDInit: Initiapisiert UART1 für SPI Betrieb                             */
/*  LCDSend: Senden eines Zeichens                                           */
/*  LCDUpdate: Aktualisierung des Displayinhalts                             */
/*  LCDClear: löscht gesammten Displayinhalt                                 */
/*  LCDPixel: ein einzelnes Pixel auf Displayausgeben                        */
/*  LCDChrXY: Aufbereiten eines Zeichens                                     */
/*                                                                           */
/*                                                                           */
/*  History:                                                                 */
/*  V0.00    08.06.2008    PVI         Erste Tests mit LCD Display           */
/*  V0.01    15.06.2008    PVI         Diverse Funktionen implementiert      */
/*  V0.02    14.06.2008    PVI         Diverse Anpassungen / Funktionen      */
/*  V0.03    20.07.2008    PVI         Diverse Anpassungen / Kommentare      */
/*****************************************************************************/
#ifndef lcd_def
#define lcd_def

// Definitionen
#define NULL                        0
#define FALSE                       0
#define TRUE                        1

#define SEND_CMD                    0
#define SEND_CHR                    1

#define LCD_X_RES                   84
#define LCD_Y_RES                   48

#define PIXEL_OFF                   0
#define PIXEL_ON                    1
#define PIXEL_XOR                   2

#define FONT_1X                     1
#define FONT_2X                     2


#define LCD_CACHE_SIZE              ((LCD_X_RES * LCD_Y_RES) / 8)

/*****************************************************************************/
/*  Funktion :  LCD_Init                                                     */
/*  Initialisiert die SPI Schnittstelle für das LCD Display                  */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
```

```
void LCDInit(void);

/*****************************************************************************/
/*  Funktion :  LCDSend                                                      */
/*  Sendet ein Zeichen ans LCD Display                                       */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  data und  SEND_CHR oder SEND_CMD                      */
/*                                                                           */
/*          Output  :  -                                                    */
/*****************************************************************************/
void LCDSend(unsigned char data, unsigned char cd);

/*****************************************************************************/
/*  Funktion :  LCDUpdate                                                    */
/*  Updatet den Display Speicher                                             */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                    */
/*****************************************************************************/
void LCDUpdate ( void );

/*****************************************************************************/
/*  Funktion :  LCDClear                                                     */
/*  Löscht das LCD Display                                                   */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  -                                                     */
/*                                                                           */
/*          Output  :  -                                                    */
/*****************************************************************************/
void LCDClear(void);


/*****************************************************************************/
/*  Funktion :  LCDPixel                                                     */
/*  Einzelnes Pixel kann auf Display angezeigt werden                        */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  x-Koordinate, y- Koordinate, mode                     */
/*                                                                           */
/*          Output  :  -                                                    */
/*****************************************************************************/
void LCDPixel (unsigned char x, unsigned char y, unsigned char mode );


/*****************************************************************************/
/*  Funktion :  LCDChrXY                                                     */
/*  Einzelnes Zeichen auf Display ausgeben                                   */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  x-Koordinate, y- Koordinate, mode                     */
/*                                                                           */
/*          Output  :  -                                                    */
/*****************************************************************************/
```

```
void LCDChrXY (unsigned char x, unsigned char y, unsigned char ch );

/*****************************************************************************/
/*  Funktion :  LCDContrast                                                  */
/*  Kontrast des LCD Displays einstellen                                     */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  contrast                                              */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void LCDContrast(unsigned char contrast);

/*****************************************************************************/
/*  Funktion :  LCDStr                                                       */
/*  String an LCD senden                                                     */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  row (Zeile), sp (Spalte), *dataPtr (String)           */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void LCDStr(unsigned char row, unsigned char sp, unsigned char *dataPtr );


static const unsigned char FontLookup [][5] =
{
    { 0x00, 0x00, 0x00, 0x00, 0x00 },  // sp
    { 0x00, 0x00, 0x2f, 0x00, 0x00 },   // !
    { 0x00, 0x07, 0x00, 0x07, 0x00 },   // "
    { 0x14, 0x7f, 0x14, 0x7f, 0x14 },   // #
    { 0x24, 0x2a, 0x7f, 0x2a, 0x12 },   // $
    { 0xc4, 0xc8, 0x10, 0x26, 0x46 },   // %
    { 0x36, 0x49, 0x55, 0x22, 0x50 },   // &
    { 0x00, 0x05, 0x03, 0x00, 0x00 },   // '
    { 0x00, 0x1c, 0x22, 0x41, 0x00 },   // (
    { 0x00, 0x41, 0x22, 0x1c, 0x00 },   // )
    { 0x14, 0x08, 0x3E, 0x08, 0x14 },   // *
    { 0x08, 0x08, 0x3E, 0x08, 0x08 },   // +
    { 0x00, 0x00, 0x50, 0x30, 0x00 },   // ,
    { 0x10, 0x10, 0x10, 0x10, 0x10 },   // -
    { 0x00, 0x60, 0x60, 0x00, 0x00 },   // .
    { 0x20, 0x10, 0x08, 0x04, 0x02 },   // /
    { 0x3E, 0x51, 0x49, 0x45, 0x3E },   // 0
    { 0x00, 0x42, 0x7F, 0x40, 0x00 },   // 1
    { 0x42, 0x61, 0x51, 0x49, 0x46 },   // 2
    { 0x21, 0x41, 0x45, 0x4B, 0x31 },   // 3
    { 0x18, 0x14, 0x12, 0x7F, 0x10 },   // 4
    { 0x27, 0x45, 0x45, 0x45, 0x39 },   // 5
    { 0x3C, 0x4A, 0x49, 0x49, 0x30 },   // 6
    { 0x01, 0x71, 0x09, 0x05, 0x03 },   // 7
    { 0x36, 0x49, 0x49, 0x49, 0x36 },   // 8
    { 0x06, 0x49, 0x49, 0x29, 0x1E },   // 9
    { 0x00, 0x36, 0x36, 0x00, 0x00 },   // :
    { 0x00, 0x56, 0x36, 0x00, 0x00 },   // ;
    { 0x08, 0x14, 0x22, 0x41, 0x00 },   // <
    { 0x14, 0x14, 0x14, 0x14, 0x14 },   // =
    { 0x00, 0x41, 0x22, 0x14, 0x08 },   // >
    { 0x02, 0x01, 0x51, 0x09, 0x06 },   // ?
    { 0x32, 0x49, 0x59, 0x51, 0x3E },   // @
    { 0x7E, 0x11, 0x11, 0x11, 0x7E },   // A
    { 0x7F, 0x49, 0x49, 0x49, 0x36 },   // B
    { 0x3E, 0x41, 0x41, 0x41, 0x22 },   // C
    { 0x7F, 0x41, 0x41, 0x22, 0x1C },   // D
    { 0x7F, 0x49, 0x49, 0x49, 0x41 },   // E
    { 0x7F, 0x09, 0x09, 0x09, 0x01 },   // F
```

```
    { 0x3E, 0x41, 0x49, 0x49, 0x7A },    // G
    { 0x7F, 0x08, 0x08, 0x08, 0x7F },    // H
    { 0x00, 0x41, 0x7F, 0x41, 0x00 },    // I
    { 0x20, 0x40, 0x41, 0x3F, 0x01 },    // J
    { 0x7F, 0x08, 0x14, 0x22, 0x41 },    // K
    { 0x7F, 0x40, 0x40, 0x40, 0x40 },    // L
    { 0x7F, 0x02, 0x0C, 0x02, 0x7F },    // M
    { 0x7F, 0x04, 0x08, 0x10, 0x7F },    // N
    { 0x3E, 0x41, 0x41, 0x41, 0x3E },    // O
    { 0x7F, 0x09, 0x09, 0x09, 0x06 },    // P
    { 0x3E, 0x41, 0x51, 0x21, 0x5E },    // Q
    { 0x7F, 0x09, 0x19, 0x29, 0x46 },    // R
    { 0x46, 0x49, 0x49, 0x49, 0x31 },    // S
    { 0x01, 0x01, 0x7F, 0x01, 0x01 },    // T
    { 0x3F, 0x40, 0x40, 0x40, 0x3F },    // U
    { 0x1F, 0x20, 0x40, 0x20, 0x1F },    // V
    { 0x3F, 0x40, 0x38, 0x40, 0x3F },    // W
    { 0x63, 0x14, 0x08, 0x14, 0x63 },    // X
    { 0x07, 0x08, 0x70, 0x08, 0x07 },    // Y
    { 0x61, 0x51, 0x49, 0x45, 0x43 },    // Z
    { 0x00, 0x7F, 0x41, 0x41, 0x00 },    // [
    { 0x55, 0x2A, 0x55, 0x2A, 0x55 },    // 55
    { 0x00, 0x41, 0x41, 0x7F, 0x00 },    // ]
    { 0x04, 0x02, 0x01, 0x02, 0x04 },    // ^
    { 0x40, 0x40, 0x40, 0x40, 0x40 },    // _
    { 0x00, 0x01, 0x02, 0x04, 0x00 },    // '
    { 0x20, 0x54, 0x54, 0x54, 0x78 },    // a
    { 0x7F, 0x48, 0x44, 0x44, 0x38 },    // b
    { 0x38, 0x44, 0x44, 0x44, 0x20 },    // c
    { 0x38, 0x44, 0x44, 0x48, 0x7F },    // d
    { 0x38, 0x54, 0x54, 0x54, 0x18 },    // e
    { 0x08, 0x7E, 0x09, 0x01, 0x02 },    // f
    { 0x0C, 0x52, 0x52, 0x52, 0x3E },    // g
    { 0x7F, 0x08, 0x04, 0x04, 0x78 },    // h
    { 0x00, 0x44, 0x7D, 0x40, 0x00 },    // i
    { 0x20, 0x40, 0x44, 0x3D, 0x00 },    // j
    { 0x7F, 0x10, 0x28, 0x44, 0x00 },    // k
    { 0x00, 0x41, 0x7F, 0x40, 0x00 },    // l
    { 0x7C, 0x04, 0x18, 0x04, 0x78 },    // m
    { 0x7C, 0x08, 0x04, 0x04, 0x78 },    // n
    { 0x38, 0x44, 0x44, 0x44, 0x38 },    // o
    { 0x7C, 0x14, 0x14, 0x14, 0x08 },    // p
    { 0x08, 0x14, 0x14, 0x18, 0x7C },    // q
    { 0x7C, 0x08, 0x04, 0x04, 0x08 },    // r
    { 0x48, 0x54, 0x54, 0x54, 0x20 },    // s
    { 0x04, 0x3F, 0x44, 0x40, 0x20 },    // t
    { 0x3C, 0x40, 0x40, 0x20, 0x7C },    // u
    { 0x1C, 0x20, 0x40, 0x20, 0x1C },    // v
    { 0x3C, 0x40, 0x30, 0x40, 0x3C },    // w
    { 0x44, 0x28, 0x10, 0x28, 0x44 },    // x
    { 0x0C, 0x50, 0x50, 0x50, 0x3C },    // y
    { 0x44, 0x64, 0x54, 0x4C, 0x44 },    // z
    { 0x08, 0x6C, 0x6A, 0x19, 0x08 },    // { (gramotevichka)
    { 0x30, 0x4E, 0x61, 0x4E, 0x30 },    // | (urche)
    { 0x7E, 0x7E, 0x7E, 0x7E, 0x7E }     // kvadratche
};
```

```
static const unsigned char BitmapSmall[][18]  = {
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
  0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
  0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0,
  0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0,
  0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
  0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
  0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
  0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
};

/*****************************************************************************/
/*  Funktion :  LCDint                                                       */
/*  Integer an LCD senden                                                    */
/*                                                                           */
/*          Parameter:                                                       */
/*           Input   :  x, y, wertart (mit SFr. oder ohne), betrag           */
/*                      x=Spalte, y=Zeile, betrag=Betrag in Währung*100      */
/*           Output  :  -                                                    */
/*****************************************************************************/
void LCDint (unsigned char y, unsigned char x, int wertart, unsigned int betrag);


/*****************************************************************************/
/*  Funktion :  LCDEinzZeich                                                 */
/*  Zeigt eine einzelne int Ziffer an                                        */
/*                                                                           */
/*          Parameter:                                                       */
/*           Input   :  x, y, Ziffer                                         */
/*                      x=Spalte, y=Zeile, Ziffer                            */
/*           Output  :  -                                                    */
/*****************************************************************************/
void LCDEinzZeich (unsigned char y, unsigned char x, unsigned int Ziffer);

#endif // Nokia_lcd
```

5.10 Div_Funk.c

```
/*****************************************************************************/
/*  Div_Funk.c                          Letzte Änderung: 01.08.2008          */
/*                                                                           */
/*                          Diplomarbeit 2008                                */
/*                                                                           */
/*  Schule: Höhere Fachschule Uster (HFU)                                    */
/*  Autor: Philippe Villard (PVI)                                            */
/*                                                                           */
/*  warten: einfach Warteschalufe                                            */
/*  Aufstartvorgang: Ausgabe der Aufstartsequenz                             */
/*  gag: Test LED, Display und Smmer                                         */
/*                                                                           */
/*                                                                           */
/*  History:                                                                 */
/*  V0.00    08.06.2008    PVI         Warteschlaufe implementiert           */
/*  V0.01    19.07.2008    PVI         Aufstartvorgang implementiert         */
/*  V0.02    01.08.2008    PVI         gag implementiert                     */
/*****************************************************************************/

#include "Nokia_lcd.h"              // Funktionen für die Nokia LCD-Ansteuerung
#include "Leser.h"                  // Funktionen für die Leser-Ansteuerung
#include "beep.h"                   // Funktionen für die Signalgeber-Ansteuerung

/*****************************************************************************/
/*  Funktion :  warten                                                       */
/*  Einfache Warteschlaufe                                                   */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  Wartezeit                                             */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void warten(unsigned long a)
{
  while (--a!=0);
}


/*****************************************************************************/
/*  Funktion :  Aufstartvorgang                                              */
/*  Aufstartvorgang (LED, Display, Leser etc.)                               */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  Wartezeit                                             */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void Aufstartvorgang(void)
{
LCDInit();
LCDContrast(0x45);
LCDClear();
LCDStr (1,1, "Diplomarbeit");
LCDStr (2,5, "2008");
LCDStr (3,3, "HF USTER");
LCDStr (4,2, "Ph. Villard");
LCDUpdate();
warten(400000);
init_Leser();
LCDStr (3,2, "Willkommen");
LCDUpdate();
warten(100000);
LCDClear();
```

```
startok_beep();
}


/*****************************************************************************/
/*  Funktion :  gag                                                          */
/*  Test Display, Summer, LED                                                */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  Wartezeit                                             */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void gag(void)
{
      LCDClear();
      LCDStr (1,0, "FLORIDA!");
      LCDUpdate();
      abwerten_beep();
      warten(100000);
      LCDClear();
      LCDStr (2,0, "FLORIDA!");
      LCDUpdate();
      laden_beep();
      warten(100000);
      LCDClear();
      LCDStr (3,0, "FLORIDA!");
      LCDUpdate();
      startok_beep();
      warten(100000);
      LCDClear();
      LCDStr (4,0, "FLORIDA!");
      LCDUpdate();
      warten(100000);
}
```

5.11 Div_Funk.h

```
/*****************************************************************************/
/*  Div_Funk.h                          Letzte Änderung: 01.08.2008          */
/*                                                                           */
/*                          Diplomarbeit 2008                                */
/*                                                                           */
/*  Schule: Höhere Fachschule Uster (HFU)                                    */
/*  Autor: Philippe Villard (PVI)                                            */
/*                                                                           */
/*  warten: einfach Warteschalufe                                            */
/*  Aufstartvorgang: Ausgabe der Aufstartsequenz                             */
/*  gag: Test LED, Display und Smmer                                         */
/*                                                                           */
/*                                                                           */
/*  History:                                                                 */
/*  V0.00    08.06.2008    PVI         Warteschlaufe implementiert           */
/*  V0.01    19.07.2008    PVI         Aufstartvorgang implementiert         */
/*  V0.02    01.08.2008    PVI         gag implementiert                     */
/*****************************************************************************/
#ifndef Div_Funk_h
#define Div_Funk_h
/*****************************************************************************/
/*  Funktion :  warten                                                       */
/*  Einfache Warteschlaufe                                                   */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  Wartezeit                                             */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void warten(unsigned long a);

/*****************************************************************************/
/*  Funktion :  Aufstartvorgang                                              */
/*  Aufstartvorgang (LED, Display, Leser etc.)                               */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  Wartezeit                                             */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void Aufstartvorgang(void);

/*****************************************************************************/
/*  Funktion :  gag                                                          */
/*  Test Display, Summer, LED                                                */
/*                                                                           */
/*         Parameter:                                                        */
/*          Input   :  Wartezeit                                             */
/*                                                                           */
/*          Output  :  -                                                     */
/*****************************************************************************/
void gag(void);

#endif // Div_Funk_h
```