Werner-Jaeger-Gymnasium Nettetal

Facharbeit Stufe 12

1. Halbjahr

Ralf Wilke

Informatik

Entwicklung und Realisation der Hard- und Software für ein mikrocontrollergesteuertes Thermometer mit Langzeitmesswertspeicher, alphanumerischem LC-Display, Benutzermenüführung und Interface zum PC zwecks Auswertung und Weiterverarbeitung der gesammelten Daten

Inhaltsverzeichnis

Grobe Gliederung

Diese Gliederung entstand zu Beginn der Arbeit und fasst den Arbeitsablauf mit seinen Teilaufgaben in der notwendigen Reihenfolge zusammen.

1. Entwurf und Entwicklung des Schaltungskonzeptes / Design der Schaltung
2. Realisierung auf Hardwareebene: Platine entwerfen (EAGLE-Layoutprogramm), ätzen, bestücken, mechanische Arbeiten am Gehäuse
3. Entwicklung des Programms für den Mikrocontroller in modularer Form in Assembler mit einer Entwicklungsumgebung für Windows:
   a) Programmierung der hardwarenahen Routinen zum Betreiben der einzelnen Komponenten (Speicher, LCD, ser. Schnittstelle, Temperatursensor, Eingabetaster)
   b) Programmierung von Hilfsroutinen, die bei der weiteren Entwicklung nützlich/wichtig sind
   c) Programmierung des Menüsystems zur einfachen Benutzerführung
   d) Festlegung einer Datenstruktur des Langzeitspeichers
      - Effiziente Ausnutzung des Speicherplatzes (evtl. Komprimierung)
      - Speicherung von Zeitintervall, Anfangszeit, Größe des Datensatzes,...
      - Speicherverwaltung in Hinblick auf Löschen, Umsortieren (Defragmentieren) der Daten, Ermittlung des freien Speicherplatzes
   e) Programmierung der Routinen, die über das Menüsystem auswählbar sind
4. Entwicklung eines VBA-Makros mit Microsoft Excel, dass die Daten über die ser. Schnittstelle einliest und sowohl tabellarisch als auch grafisch darstellt.

Vorwort

Zu Beginn des Jahres 2001 begann ich mich mit Mikrocontrollern zu beschäftigen. Unter www.batronix.com fand ich eine Web-Seite, die eine Assembler-Entwicklungsumgebung für Windows, Layouts für ein passendes Programmiergerät und eine Experimentierplatine sowie einen umfangreichen Fundus von Datenblättern anbietet. Beim Elektronik-Versandhandel Reichelt[1] bestellte ich mir die benötigten elektronischen Bauteile, das Basisplatinenmaterial[2] und stellte die Platinen durch Belichtung mit UV-Licht und anschließender Ätzung her.
Mit diesen Equipment und dem „Mikrocontroller Handbuch“ [1] arbeitete ich mich in die Programmierung von Mikrocontrollern ein und schreib einfache Programme wie Lauflichtsteuerung, Tonerzeugung und andere Spielereien. Mit dieser Erfahrung beginne ich jetzt dieses Projekt, das in seinem Umfang und seiner Komplexität das Vorherige weit übertrifft:
Ein mikrocontrollergesteuertes Thermometer mit 4kB E²PROM Langzeitspeicher (Datenerhalt ohne Versorgungsspannung min. 10 Jahre), LC-Display, einfacher und kompakter Bedienung durch Menüführung sowie seriellem PC-Interface, um die gesammelten Daten (grafisch) auszuwerten.
Die Aufgabenstellung beschränkt sich nun nicht mehr auf die reine Software-programmierung; vielmehr gerät auch die externe Hardware und die Mechanik in den Vordergrund. Wissen über die Handhabung der einzelnen Komponenten steht mir durch Bücher und Datenblätter des Herstellers zur Verfügung.

Zur Form

- Unterpunkte, in denen entweder besonders schwierige Probleme zu lösen waren oder die trickreiche Strategien enthalten, werden ausführlich erläutert.
- Besonders in Texten, die auf dem LCD dargestellt werden, wird die deutsche Sprache - teils mit der zu diesem Zeitpunkt alten Rechtschreibung - mit der englischen kombiniert angewandt, da der Anzeigeplatz im LCD auf 16 Zeichen pro Zeile begrenzt ist und nur durch diese Kombination das Gemeinte auf dem stark begrenzten Platz verständlich darzustellen ist. Ähnliches gilt für die Benennung von Labels und RAMs im Assembler-Quelltext, wobei hier der Vorteil der kürzeren englischen Ausdrücke in den Vordergrund tritt.
- Auf den im Anhang abgedruckten Assembler- und VBA-Quellcode wird durch die Zeilenangaben in Klammern Bezug genommen.
- Signale oder Leitungen, die mit invertierter Logik[3] arbeiten, werden durch ein Apostroph ' markiert.
- Das Wort Mikrocontroller wird wegen seines häufigen Vorkommens und seiner zentralen Bedeutung durch MC abgekürzt.
- Hexadezimalzahlen werden durch Anhängen von h, Binärzahlen durch b gekennzeichnet.
- Bezeichnungen wie R01 stellen die Register R0 und R1 dar, wobei diese als 16Bit-Wert zu interpretieren sind. Das erstgenannte Register enthält das HighByte.

---

1 Reichelt Elektronik www.reichelt.de Elektronikring 1 D- 26452 Sande

2 Basisplatinenmaterial: Platinen, die ein- oder beidseitig mit einer dünnen Kupferschicht und evtl. einem lichtempfindlichen Fotolack bedeck sind. Sie dienen beim Herstellen der Leiterbahnzüge durch Ätzen oder Fräsen das Grundmaterial dar.

3 Invertierte Logik: Im Gegensatz zu Active High-Eingängen, die bei hoher Spannung (High = H = 5 Volt) ihre Funktion ausführen, werden Active Low-Eingänge bei niedriger Spannung (Low = L = 0 Volt) aktiv.

Die Grundstruktur des Projektes

Dieses Kapitel beschreibt kurz die einzelnen Bauteile und das daraus resultierende Schaltungskonzept

Das Herzstück - Der Mikrocontroller

Das Herzstück des Projektes ist der Mikrocontroller AT89C4051 der Firma ATMEL[1]. Es handelt sich hierbei um ein Derivat des weit verbreiteten 8051 mit folgenden Merkmalen:

- 4kB wiederbeschreibbaren Flash-Programmspeicher PEROM
- 15 Input/Output (I/O) Leitungen
- 128 Bytes interner RAM
- 2 16 Bit Timer/Zähler
- 5 Interrupts mit 2 verschiedenen Prioritäten
- vollduplexfähige[2] serielle Schnittstelle „on chip“
- Taktgeschwindigkeit bedingt durch einen ext. Quarz bis 24MHz
- kompatibel mit dem Industriestandard MCS-51™
- 20 Pin PDIP[3]-Gehäuse

Der interne RAM des MC ist in vier Segmente unterteilt.

000h-01Fh: Registerbänke 0-3

Der AT89C4051 hat 4 Registerbänke mit je 8 Registern R0 bis R7. Die Auswahl der einzelnen Bank erfolgt über 2 Bits im PSW; hierzu später mehr.

020h-02Fh: Bitadressierbarer Bereich

Hier stehen 16 Byte zur Verfügung, deren Bits über die Bit-Befehle zusätzlich über die Adressen 00h-7Fh angesprochen werden können.

030h-07Fh: Daten-RAM

Dieser Bereich ist direkt byte-adressierbar und steht dem Programmierer als normaler RAM zur Verfügung.

080h-0FFh: Special Function Register-Bereich

In diesem Bereich sind die sogenannten Special Function Register (SFR) ansprechbar. Die Register sind Speicherstellen mit besonderen Funktionen wie zum Beispiel der Akku, Steuerbytes für die Interrupt-Verwaltung, die ser. Schnittstelle, die Timer, usw. Ebenfalls werden über zwei SFR die 15 I/O-Leitungen des MC gesteuert. Diese sind zu zwei Bytes zusammengefasst, und werden als Port 1 (P1) und Port 3 (P3) bezeichnet. Ausgaben von Spannungswerten erfolgen durch Schreibbefehle auf die jeweiligen Adressen; zum Einlesen müssen zuvor die entsprechenden Bits auf 1 gesetzt werden.

---

1 Europe Atmel U.K., Ltd. Coliseum Business Centre Riverside Way Camberley, Surrey GU15 3YL England TEL (44) 1276-686-677 FAX (44) 1276-686-697 e-mail literature@atmel.com Web Site http://www.atmel.com

2 Vollduplexfähig: gleichzeitiges Senden und Empfangen ist möglich

3 PDIP20: Plasic Dual-In-Line-Gehäuse mit 20 Pins

LC-Anzeige

Die aktuelle Anzeige der Temperatur und anderer Daten, sowie die Menüdarstellung übernimmt das 2-zeilige LC-Display mit 16 Zeichen pro Zeile LCD 162a von Displaytech[1]. Zur Ansteuerung der 5x7 Punktmatrix hat es einen Controller vom Typ HD44780 o.ä. bereits auf seiner Platine integriert, der einen relativ einfachen Betrieb ermöglicht. Neben zwei Anschlüssen für die LED-Hintergrundbeleuchtung (Pins 15-16) besitzt es einen 8-Bit breiten bidirektionalen[2] Datenbus (Pins 7-14), einen Enable-Eingang, der den Datentransfer auslöst, einen R/'W-Eingang, der die Richtung des Datenflusses festlegt, einen RS-Eingang, mit dem zwischen Daten- und Kommandobytes umgeschaltet werden kann, einen Kontrasteingang, sowie natürlich Anschlüsse für die Stromversorgung.

Die Softkeys

Unter dem Display sind symmetrisch 4 Taster angeordnet, deren Bedeutung abhängig vom aktuellen Menü in die untere Zeile des LCD geschrieben wird. Daher die Bezeichnung Softkeys.

Der Langzeitdatenspeicher

Um die Messwerte auch ohne Stromversorgung sicher zu speichern, werden 2 M24C16 E²PROMs[3] von STMicroelectronics[4] verwendet. Diese ICs im PDIP8-Gehäuse haben eine Speicherkapazität von je 16 kBit mit einer Organisation von 2048 x 8 Bit, also 2kByte. Sie kommunizieren über den seriellen, bidirektionalen I²C-Bus (Details im entsprechenden Kapitel) mit dem MC. Anfangs war nur ein E²PROM vorgesehen; das Hinzufügen eines Weiteren war jedoch einfach, so dass der zusätzliche Speicherplatz entweder für die angezeigten ASCII-Texte oder auch als Messwertspeicher benutzt werden kann. Diese Entscheidung kann aber erst zum Ende der Softwareprogrammierung getroffen werden, da erst dann der verbleibende freie Programmspeicherplatz im MC feststeht.

Die serielle Schnittstelle

Die ser. Schnittstelle ist onChip im MC integriert. Zum Betrieb mit einem PC ist jedoch eine Spannungswandlung nötig: Der MC arbeitet mit 0 oder 5 Volt, die ser. Schnittstelle des PCs nach dem RS232-Standard aber mit -12 bis +12 Volt. Diese Spannungswandlung nimmt das IC MAX 232 von MAXIM[5] vor.

Der Temperatursensor

Als Temperatursensor verwende ich den DS1620 von Dallas[6]. In diesem IC im 8Pin-PDIP-Gehäuse ist ein Temp.-Sensor mit A/D-Wandler integriert.

---

1 Displaytech Ltd, Email : sales@dispalytech.com.hk Fax: +852 2722 6998

2 bidirektional: In beide Richtungen benutzbar; hier: Daten können zum Display geschrieben, oder vom Display gelesen werden.

3 E²PROM: Electric Erasable Programable Read Only Memory

4 STMicroelectronics GROUP OF COMANIES www.st.com

5 MAXIM: Maxim Integrated Products, 120 San Gabriel Drive, Sunnyvale, CA 94086 (408) 737-7600

6 Dallas Semiconductor, www.dalsemi.com

Das Schaltungskonzept

Nachdem nun alle wichtigen Bauteile feststehen, gilt es, einen Schaltplan zu entwickeln, der mit möglichst wenig zusätzlichen Bauteilen die einzelnen Hardwarekomponenten zum Zusammenspiel bewegt.

Von den 15 bidirektionalen Leitungen des MC sind in Port 1 acht zusammengefasst, die restlichen 7 in Port 3. Die 16. Leitung (Bit 6 von P 3, auch P3.6 genannt) ist intern mit dem Ausgang eines analogen Komparators[1] verbunden und kann deshalb hier nicht genutzt werden. Insgesamt müssen min. 23 ext. Bauteilanschlüsse bedient werden: 11 beim LCD, 3 für die E²PROMs, 4 für die Taster, 3 für den Temp.-Sensor und 2 für die ser. Schnittstelle. Da dieses mit 15 Leitungen direkt nicht möglich ist, muss eine Porterweiterung vorgenommen werden. Hierzu gibt es mehrere Möglichkeiten:

1. Benutzung von I²C-Bus-Porterweiterungsbausteinen wie dem PCF 8574
   - Vorteil: Sie können direkt an den I²C-Bus, der schon für die E²PROMs benötigt wird, angeschlossen werden und besitzen 8 I/O-Anschlüsse sowie eine Interruptleitung, die Änderungen an den Eingängen anzeigt und über den Interrupt-Eingang des MC ausgewertet werden kann. (vgl. [1]: 109, 113).
   - Nachteil: Wegen des I²C-Protokolls ist die Ansteuerung nur langsam möglich, der Preis ist rel. hoch[2] und die Interruptprogrammierung macht das Programm komplexer und fehleranfälliger.
2. Benutzung von Schieberegistern
   - Vorteil: Mit jeweils 3 Leitungen können durch Kaskadierung von Ser. In/Par. Out-Schieberegistern die benötigten Ausgänge bzw. mit Par. In/Ser. Out-Schieberegistern Eingänge zur Verfügung gestellt werden.
   - Nachteil: 6 Leitungen werden benötigt, der Vorteil der Kaskadierbarkeit kann nicht ausgenutzt werden, da nicht so viele Ein- und Ausgänge benötigt werden.
3. Daten-Adressbus-System
   - Vorteil: Es ist in der MC-Technik üblich, z.B. zum Betreiben von ext. parallelen Daten- und Programmspeichern wie EPROMs und RAMs; es hat keinen Protokollaufwand, ist schnell und läßt sich mit dem Anschluss des LCD gut kombinieren. Zusätzlich stehen mir viele Informationen hierzu in [1] zur Verfügung.
   - Nachteil: Ein Port ist als Datenbus belegt, die verbleibenden 7 Leitungen müssen für den Adressbus und alle anderen Funktionen wie dem I²C-Bus ausreichen.

Nach Abwägung der Vor- und Nachteile habe ich mich für das Daten-Adressbus-System entschieden, weil es den Anforderungen genügt und die geringsten Schwierigkeiten bei der praktischen Umsetzung zu erwarten sind.

1 Komparator: Vergleicher zweier Spannungen, hier zwischen P1.0 und P1.1

2 Vgl. Reichelt Elektronik Katalog 10/2001, Seite 354 PCF 8574: 5,50DM

Die Realisierung

Port 1wird als Datenport mit seinen 8 bidirektionalen Leitungen benutzt, an die andere Bauteile wie das LCD einfach angeschlossen werden. Die Tasterzustände werden über ein Latch[1] auf den Datenbus gelegt und die Steuersignale für das LCD und den Temp.-Sensor über ein Flip-Flop mit Takteingang[2] vom Bus übernommen.

Nun stellt sich aber das Problem, woher jedes Bauteil des Datenbusses weiß, wann es Daten senden oder empfangen soll. Hierzu nutze ich einen 2 Bit breiten Adressbus, angeschlossen an Port 3:
00b: Keines der Bauteile ist aktiv
01b: LCD
10b: Flip-Flop für Statusausgänge
11b: Latch für Tasterzustände

Als Leitungen werden P3.4 und P3.5 benutzt und dem 74 HC 138[3] zugeführt. Dieses IC hat Active Low-Ausgänge, deshalb müssen mit dem 74 HC 14 zwei Signale invertiert werden.
Das Latch 74 HC 541 ist 8 Bit breit, von denen aber nur 4 Bit für die 4 Taster benötigt werden. Das Widerstandsarray RN1(siehe Schaltplan) sorgt für definierte Spannungszustände auf den Tasterleitungen. Das Flip-Flop 74 HC 574 arbeitet nur in Ausgaberichtung, man kann mit ihm also nur die Steuereingänge des LCD und des Temp.-Sensors bedienen.

Der I²C-Bus für die E²PROMs besteht aus einer Taktleitung, der „serial clock“ (SCL). In ihrem Takt werden über die „serial data“ (SDA)-Leitung Daten übertragen. Glücklicherweise lassen es die Spezifikationen des I²C-Busses zu, die SDA-Leitungen der beiden E²PROMs zu verbinden und mit dem Flip-Flop lediglich getrennte Taktsignale zu erzeugen.[4]

Der Temperatursensor wird über ein 3-Draht-Interface angesteuert. Eine ChipSelect-Leitung aktiviert das IC und eine Taktleitung definiert die Geschwindigkeit, mit der über die bidirektionale Datenleitung DQ die Daten transportiert werden.[5]

Grob betrachtet sind also 3 Bus-Systeme kombiniert; ein vielleicht verbesserungsfähiges Konzept, das aber auf jeden Fall Abwechslung und Herausforderung birgt.

---

1 Latch: Bauteil aus der Logikreihe, das Spannungspegel auf Befehl durchlässt.

2 Flip-Flop mit Takteingang: Bauteil aus der Logikreihe, das auf einen Takt-Impuls hin Signalzustände speichert.

3 74 HC 138: 3/8 Binär-Dezimalwandler, das je nach Binärzahl am Eingang einen entsprechenden Ausgang aktiviert.

4 STMicroelectronics, Datenblatt „16/8/4/2/1 Kbit Serial I²C BUS EEPROM“ 2000, Seite 4, Figure 4: I²C Bus Protocol

5 DALLAS SEMICONDUCTOR, Datenblatt DS1620 „Digital Thermometer and Thermostat“ : Seiten 6, 9f.

Aufbau und Mechanik

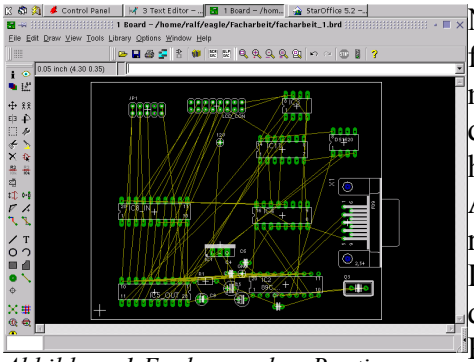

Abbildung 1 Eagle, vor dem Routing

Nachdem der Schaltplan feststeht, muss dieser für das Ätzen layoutet werden. Hierzu wird er mit dem Layoutprogramm EAGLE von CadSoft, dessen Light Edition von der Webseite kostenlos heruntergeladen werden kann, gezeichnet. Anschließend werden alle Bauteile mit ihren realen Abmaßen und den vorher festgelegten Leitungen im Layoutbereich angezeigt. Wegen der durch die Light Edition beschränkten Platinengröße kann auf ein zweiseitiges Routen nicht verzichtet werden. Dieses hat für die spätere Platinenherstellung den großen Nachteil, dass die beiden Folien für Ober- und Unterseite zum Belichten auf min. 0,5 mm genau auf der kupferkaschierten und mit Fotolack überzogenen Platine liegen müssen, denn die Bohrungen für die Bauteile und Durchkontaktierungen[1] müssen natürlich auf beiden Seiten mit dem Verlauf der Kupferbahnen übereinstimmen. Besonders beim Umdrehen zum Belichten der zweiten Seite der Platine ist ein Verrutschen fast nicht zu vermeiden. Der erste Versuch scheiterte, wenngleich zu dem Positionierungs-problem noch die Erschwerung durch eine nicht deckende Kopiererschwärze kam. Diese brachte als Resultat stark durchlöcherte bis nicht mehr vorhandene Kupferbahnen, so dass die Platine nicht mehr zu gebrauchen war.
Der zweite Versuch mit Qualitäts-Platinenmaterial von Bungard, dass in Bezug auf die Belichtungszeit sehr tolerant ist, gelang jedoch. Das Problem des Verrutschens wurde auf folgende Art gelöst: Eine Folie wurde auf die überstehende Platine mit Tesafilm geklebt, markante Lötaugen durchbohrt, mit diesen auf der anderen Seite die Folie ausgerichtet und ebenfalls mit Tesafilm fixiert. Glücklicherweise war sowohl die Auflösung des Kopierers, als auch die des Fotolacks groß genug, dass auch die feinen Leiterbahnen zwischen den IC-Anschlüssen durchgehend und ohne Kurzschlüsse entstanden sind. Nach dem Belichten, Ätzen, Zurechtsägen, Bohren der 223 Löcher und Beschichten mit Lötlack lag die fertige Platine zur Bestückung bereit.

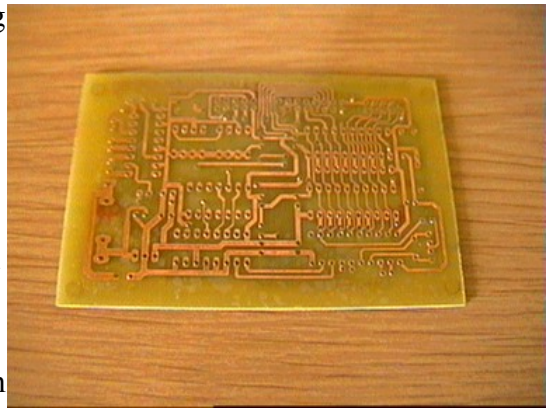

Abbildung 2 Die geätzte Platine

Das Projekt findet in einem 160x95x60 mm großen Kunststoffgehäuse mit Aludeckplatte von TEKO Platz. Auf 5mm-Abstandshaltern steht die doppelseitige Hauptplatine, auf die alle Bauteile bis auf das LCD, die 4 Taster, den Temp.-Sensor und den Einschaltknopf gelötet sind. Alle ICs sind gesockelt, um bei evtl. Defekten durch Unfälle während der Entwicklung einen unproblematischen Austausch möglich zu machen. Besonders die Fassung des MC wird durch die häufige Umprogrammierung während der Entwicklung im ext. Programmiergerät stark beansprucht. Hier hätte sich auch die Möglichkeit der Verwendung eines MCs mit InCircuit-Programmierbarkeit[2] angeboten; dies hätte aber einen anderen MC mit

---

1Durchkontaktierungen: Verbindungen der Leiterbahnen zwischen der Ober- und Unterseite
2InCircuit-Programmierbarkeit: Möglichkeit, einen MC in der Zielschaltung mit wenigen Leitungen seriell zu programmieren, so dass dieser in der Programmentwicklung zum Umprogrammieren und Testen nicht aus der Fassung genommen werden muss.

anderem Gehäuse sowie ein neues Programmiergerät zur Folge gehabt, dessen Aufwand und Umstellung ich scheute.
Die Leitungen für das LCD, die Taster und den Temp.-Sensor sind über doppelreihige Stiftleisten erreichbar. Auf diese werden übliche Flachkabelstecker gesteckt, um so die Verbindungen zwischen abnehmbarer Frontplatte und Hauptplatine herzustellen. Die 4 Taster sind auf eine Trägerplatine gelötet, die ebenfalls eine solche Stiftleiste trägt. Zusammen mit dem LC-Display, das oberhalb der Taster in der dafür ausgefeilten und ausgekleideten, rechteckigen Aussparung der Deckplatte seinen Platz findet, ist diese über Kunststoffabstandsbolzen mit einer weiteren Trägerplatine verschraubt. Auf ihr sind zusätzlich ein Poti[1] zur Kontrast-eeinstellung, ein Kondensator von 100µF zur Spannungsstabilisierung und die 16polige Stiftleiste für den Anschluss des LCD vorhanden. Sinn der 2. Trägerplatine ist weiterhin die Beschränkung auf 4 Senkkopfschrauben, die durch die metallene Deckplatte hindurch die Taster und das LCD befestigen.

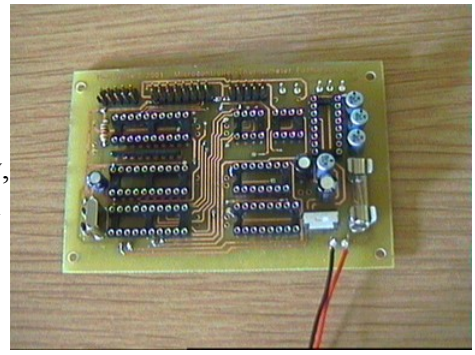
Abbildung 3 Die fast fertig bestückte Platine

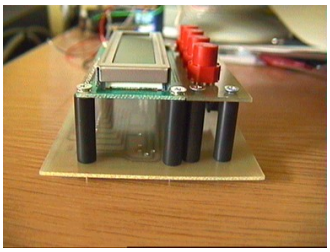
Abbildung 4 Bedienteil,links

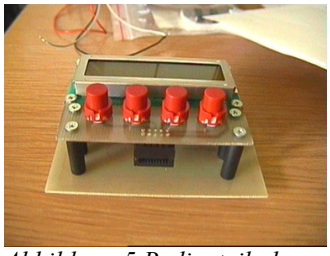
Abbildung 5 Bedienteil,oben

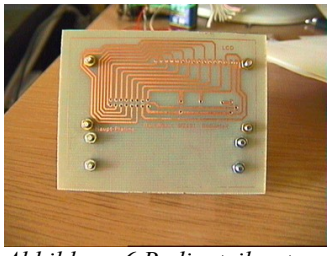
Abbildung 6 Bedienteil,unten

Die Anordnung der Bedienelemente, zu denen zusätzlich noch das Poti mit Schalter zur Einstellung der Hintergrundbeleuchtung zählt, wurde am PC entschieden. Mit einem Bildbearbeitungsprogramm und einer Videokamera erstellte ich Einzelbilder der Bauteile und plazierte sie dann am Bildschirm, so dass ein ästhetisch ausgewogenes Bild entstand.


Abbildung 7 Gehäuse, Frontplatte

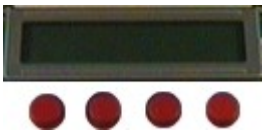
Abbildung 8 LCD, Taster


Abbildung 9 Drehknopf

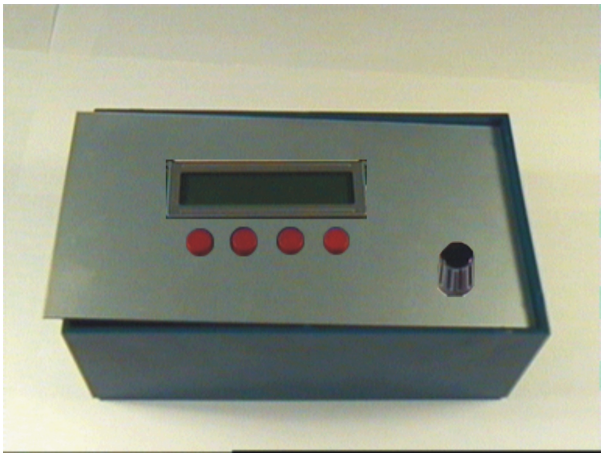
Abbildung 10 Mit dem PC designte Frontplatte

1 Poti: Abkürzung für Po

Hardwareroutinen

Taster

Die Hardwareansteuerung des Taster-Latches übernimmt die Routine RD_Taster in Zeilen 2047-2079. Zuerst wird das Interrupt-Lock-Bit gesetzt, um Interrupts der ser. Schnittstelle während des Einlesens zu unterbinden. Dann werden die Enable-Leitungen auf Port 3 für das Latch gesetzt und über den vor mit FFh beschriebenen, und damit als Eingang nutzbar gemachten Datenport P1 die Tasterzustände in das RAM INPUT_TASTER geMOVt (2069). Abschließend werden die Enable-Leitungen wieder deaktiviert und ein evtl. eingetroffener Interrupt nachträglich „manuell“ ausgelöst (2074-2078).
Hierauf baut die Routine “Wait_for_Taster“ (1994 bis 2044) auf, der aber aufgrund ihrer Funktion als Hauptwarteschleife noch die Aufgabe der Auslösung über bestimmte Bits der Temperaturmessung und des Blinkens verschiedener Zeichen im LC-Display zukommt. Eine durch die Bauart der Taster bedingte Entprellung wird realisiert wie folgt: Die aktuellen Tasterzustände werden eingelesen und mit dem Wert „ Kein Taster gedrückt = 00h“ verglichen. Wird ein Taster gedrückt, so werden die o.g. Nebenfunktionen der Schleife übersprungen, 100 Mikrosekunden gewartet und anschließend das vorher auf den Wert 255 gesetzte Zählregister R6 dekrementiert. Ist es nicht Null, so werden diese Aktionen wiederholt. Nach der 255-maligen Detektierung dieses Tasterzustands kann der Einschaltvorgang als hinreichend entprellt angenommen werden und die Schleife (2032-2038) beginnt, die die Taster wieder abfragt und erst bei Lösen der Taste verlassen wird. Anschließend wird auch dieses in der o.g. Weise entprellt.

I²C-Bus

Der I²C-Bus ist ein zweiadriger Bus bestehend aus der Taktleitung SCL, in deren Takt Daten bidirektional über die Datenleitung SDA vom und zum E²PROM übertragen werden. Jede Übertragung beginnt mit der Startbedingung gefolgt vom „Device Selektion“ Byte, das zur Adressierung und Unterscheidung mehrerer Bausteine an einem Bus dient. Anschließend werden die Nutzdaten gefolgt von der Stoppbedingung, die den Datentransfer beendet, übertragen.
Die Implementierung des I²C-Bus erfolgt deshalb in modularer Aufbauweise. Die Routine „I2C_Start“ (3215-3231) generiert die Startbedingung; entsprechendes gilt für die Stopproutine (3243-3251). Als Besonderheit bei dieser Art von E²PROMs mit 2kB Speicherplatz werden zur Adressierung 11 Bits benötigt. Das Device Selektion Byte wird deshalb teilweise für die Übertragung der 3 MSBs[1] mitbenutzt. Dies regelt die Routine I2C_Dev_Sel (3254-3270). Durch den dadurch bedingten Wegfall dieser Bits zur o.g. Unterscheidung mehrerer I²C-Bausteine würden zur Ansteuerung der beiden E²PROMs zwei unterschiedliche I²C-Busse benötigt. Die Spezifikation erlauben jedoch laut Datenblatt[2] die Verknüpfung der beiden SDA-Leitungen zu einer. Wird jeweils nur ein E²PROM angesprochen und die jeweils andere SCL-Leitung auf Low gehalten, dann kann das nicht ausgewählte E²PROM nie eine Startbedingung empfangen und bleibt somit im Standby-Mode. Dies ist wichtig, damit die gemeinsame SDA-Leitung nicht gestört wird.
Die serielle Ein- und Ausgabe von Byte-Werten sind in den Routinen I2C_Byte_write sowie I2C_Byte_read (3272-3336) implementiert.

1 MSB: Most Significant Bit

2 DS1620-Datenblatt „DS1620 Digital Thermometer and Thermostat“, Seite 4

E²PROM

Um einen vollständigen Datentransfer zu ermöglichen und den Programmierer bei der späteren Verwendung der E²PROMs von der Hardware freizuhalten, müssen die Teilroutinen zusammengefasst werden. Dies stellen die Routinen EEPROM_Byte_write (3337-3308) und EEPROM_Byte_read (3309-3336) dar.

Die Adressierung des E²PROM-Speichers geschieht über den Datenpointer des AT89C4051. Diese Vorgehensweise hat mehrere Gründe:
- Der Datenpointer ist das einzige 16-Bit-Register im MC.
- Der Befehl INC DPTR ist der einzige 16 Bit übergreifende INC-Befehl.

Da der Gesamtspeicher der beiden E²PROMs 4 kB beträgt, sind zur Adressierung 12 Bits nötig. Würde man diese auf 2 Byte RAM einteilen, würde ein Übertrag bei INC- oder DEC-Befehlen nicht automatisch weitergeleitet - eine Mathematik-Routine müsste an vielen Stellen eingefügt oder aufgerufen werden, sodass das Ziel der effizienten Ausnutzung des nur begrenzt zur Verfügung stehenden Programm-speicherplatzes sowie einer Laufzeitoptimierung nicht erreicht werden kann.

DS1620

Der Temperatursensor DS 1620 wird über ein serielles 3-Draht-Interface, bestehend aus einer Enable-Leitung ('RST), einer Taktleitung (CLK) und einer bidirektionalen Datenleitung (DQ), angesteuert. Für eine Datenübertragung wird zuerst die 'RST-Leitung auf High gesetzt, dann im Takt der CLK-Leitung ein Protokollbyte zur Auswahl der verschiedenen Funktionen[1] gesendet, sodass in den folgenden 8-9 Takten die Nutzdaten übertragen werden und abschließend 'RST durch Low-Pegel das IC in den Standbymode bringt[2].
Die Protokollausgabe wird durch die Routine Proto_DS1620 (3408-3448), das Senden und Empfangen von Daten durch WR_ und RD_DS1620 beschrieben.

LCD

Der auf der Platine des LCDs integrierten Kontroller KS0070B o.ä. verwaltet einen eigenen Speicher, der sich in den Display Data (DD)- und den Character Generator (CG)-RAM aufteilt. Das Display zeigt in der ersten Zeile die Zeichen des Adressbereichs 00h-0Fh, in der zweiten die Zeichen des Adressbereich 40h-4Fh an. In den CG-RAM kann der Programmierer 8 Zeichen mit der verwendeten 5x7 Dot-Matrix selbst kreieren. Diese sind dann unter dem ASCII-Wert 00h bis 07h ansprechbar.
Vor Benutzung des LC-Displays muss dieses nach dem Datenblatt [2] Seite 17-23 erst auf den gewünschten Betriebsmodus (2 Zeilen, 5x7 Dot-Matrix Display on, Cursor off, Blink off, Increment Mode[3], Entire Shift off[4]) initialisiert werden (167-217).
Zur einfachen Verwendung stehen verschiedene Hilfsroutinen zur Verfügung. Auch diese sind modular von der Hardware- zur Softwareebene aufgebaut. Grundlegend sind hier die Routinen LCD_WR und LCD_RD (1950-1978), die jeweils ein Byte über den Datenbus zum LCD schreiben oder von ihm lesen.

---

1 DS1620-Datenblatt, Seiten 6-7

2 DS1620-Datenblatt, Seiten 9

3 Increment Mode: Nach dem Schreiben eines Zeichen in den DD-RAM wird der Cursor automatisch eine Position weiter gesetzt.

4 Entire Shift off : Der Inhalt des Displays wird nicht synchron zum Cursor verschoben.

LCD_READY (1980-1990) wird hierbei aufgerufen, um durch Polling[5]-Betrieb das Ende der Abarbeitung eines an das Display geschickten Befehls zu erkennen. Hierauf aufbauend setzt die Routine LCD_ADD_SET(1922-1953) den Cursor im LCD auf die im RAM LCD_ADDRESS benannte Stelle; LCD_S_CHAR (1893-1906) schreibt ein Byte in das DD-oder CG-RAM.

ASCII-Text-Speicherung

Die angezeigten Texte werden am Ende des Programmcodes als ASCII-Werte mit in den Programmspeicher übertragen. Um eine effiziente Ausnutzung des Programmspeichers zu erreichen, wird bei Zeichenkettenlängen unter 15 Bytes das Trennzeichen A6h angefügt, das im LCD ein chinesisches Zeichen ergeben würde und deshalb hier genutzt werden kann. Das in der PC-Programmierung häufig verwendete Tennzeichen 00h (Nullterminierte Strings)kann hier nicht benutzt werden, da gerade der ASCII-Bereich 00h-07h die vom Programmierer frei definierbaren Zeichen im LCD darstellt; in diesem Fall 00h = „Pfeil nach oben“.

Ausgabe von Text

Die Ausgabe von Texten auf das LCD geschieht durch die Routine Line2LCD (1859-1889). Die jeweilige Zeile wird durch das Bit LCD_Zeile ausgewählt; der Datenpointer muss auf die Anfangsadresse eines -wie oben beschrieben- existierenden Textfeldes zeigen. Der Cursor im LCD wird entsprechend auf den Anfang der 1. oder 2. Zeile gesetzt. Eine Schleife liest ab der im Datenpointer gegebenen Adresse max. 16 Byte aus und schreibt diese auf das LCD, wenn sie nicht zuvor durch das Trennzeichen A6h abgebrochen wird.

Ausgabe von Zahlen

Bei der Ausgabe von Zahlenwerten auf das LCD ist zuvor eine Umwandlung von der hexadezimalen Übergabe der darzustellenden Zahl im Akku in eine dezimale Form notwendig. Des Weiteren müssen Zehner- und Einerstelle getrennt als „Text“ ausgegeben werden.
Zuerst wird die übergebene Zahl zur dezimalen Auftrennung durch 10 geteilt (1773-1775). Die Zehnerstelle bleibt als ganzzahliger Teil des Quotienten im Akku, die Einerstelle als Rest im Hilfsakku B. Nun läßt sich geschickt ausnutzen, dass in der ASCII-Tabelle (s. Anhang) die arabischen Zahlen mit Null beginnend von 30h bis 39h fortlaufend festgelegt sind. Durch Addition des Offsets von 30h ergibt sich so automatisch der richtige ASCII-Wert der darzustellenden Zahl, der als Text einfach ausgegeben wird.

Serielle Schnittstelle / Interrupt

Die auf dem MC integrierte serielle Schnittstelle muss vor ihrer Benutzung auf die gewünschte Geschwindigkeit eingestellt werden. Dies erfolgt über den Timer 1, der in dem hier verwendeten Modus als doppeltes 8Bit-Register verwendet wird. Das LowByte wird im Systemtakt inkrementiert und nach einem Überlauf von 255 auf 0 mit dem im HighByte gespeicherten Reload-Wert neu gesetzt. Die Überlaufrate dieses Timers bestimmt dann die Geschwindigkeit der seriellen Schnittstelle, hier 4800 Baud (224-234). Zusätzlich muss, damit das Programm ein empfangenes Zeichen automatisch bemerkt, der Interrupt der Schnittstelle in Zeile 245 freigegeben werden.
Die im MC festgelegte Einsprungadresse dieses Interrupts ist 0023h. An dieser Stelle (154) befindet sich ein LJMP-Befehl auf die Interrupt-Service-Routine Ser_Int

5 Polling: Wiederholtes Abfragen zum Testen auf einen bestimmten Zustand

(2600-2656). Diese Routine muss am Anfang unterscheiden, ob der Interrupt durch ein empfangenes Zeichen oder durch das Ende einer Aussendung ausgelöst wurde.

Empfangen

Wie es sich für eine Interruptroutine gehört, werden zuerst die Register, die garantiert geändert werden, auf den Stack gePUSHt, um am Ende des Interrupts den Ausgangszustand für die unterbrochene Routine wieder herzustellen. Anschließend wird durch das Bit Ser_Int_lock überprüft, ob die unterbrochene Routine nicht gestört werden darf. In diesem Falle wird durch das Bit Ser_Int_arrived das Eintreffen vorgemerkt und die Routine verlassen. Anderenfalls wird das empfangene Zeichen aus dem Puffer SBUF in den Akku übertragen, um die Art der angeforderten Aktion erkennen. Dies geschieht durch die CJNE-Befehle (2626-2633), die die jeweiligen Routinen aufrufen.

Senden

Beim Aussenden von Zeichen ist es wichtig, ein neues Zeichen erst dann in den Sendepuffer SBUF zu schreiben, wenn das alte Zeichen vollständig gesendet wurde. Die Hauptfunktion der Senderoutine ist die Ermittlung dieses Zeitpunktes sowie die Unterscheidung, ob die Routine aus dem Hauptprogramm oder aus einer bereits laufenden Empfangsinterruptroutine aufgerufen wurde.
Wurde der Interrupt durch das Ende einer Aussendung ausgelöst, dann wird das Bit Ser_ready gesetzt, um dies nach außen anzuzeigen. Zusätzlich muss das entsprechende Interrupt-Flag TI zurückgesetzt werden, damit der Interrupt nicht sofort wieder aufgerufen wird.
Ist, wie in der Diagnose-Routine, das Senden durch den Interrupt eines empfangenen Zeichens ausgelöst, so kann man das Ende der Aussendung nicht wie oben beschrieben feststellen. Da ein Interrupt niemals von einem neuen Interrupt gleicher Art - wie es am Ende der Aussendung des betreffenden Zeichens geschehen würde - unterbrochen werden kann, muss durch die separate Routine Ser_Send_while_Int das TI-Flag in einer Schleife abgefragt werden.

Softwareroutinen

Diagnose-Routine

Eine ganz entscheidende Hilfe während der Programmentwicklung war die Diagnoseroutine SNAP (2906-3024). Auf Anforderung über die serielle Schnittstelle wird über diese zu einem auf dem angeschlossenen PC laufenden Terminal-Programm der Inhalt des internen RAMs sowie interessante Teile des E²PROM -Speichers in tabellarischer Form hexadezimal ausgegeben. Hierdurch konnte man einen Eindruck über die aktuellen Zustände der Speicherzellen erhalten und Programmierfehler erkennen und beheben.

Menüsystem

Zur einfachen und kompakten Bedienung aller für den Benutzer interessanten Funktionen wird zur Steuerung ein Menüsystem verwendet:

1 Hauptmenü
  1.1 Messwerterfassung
    1.1.1 Zeitintervall eingeben
    1.1.2 Startdatum eingeben - Tag und Monat
    1.1.3 Startuhrzeit eingeben - Stunde und Minute
    1.1.4 Anzahl der Messungen eingeben
      1.1.4.1 Test auf nicht ausreichenden Speicherplatz
      1.1.4.2 Ermittlung der max. Anzahl von Messungen
    1.1.5 Anzeige des Fortschritts der Messwertaufnahme mit aktueller Temperatur
  1.2 Datenverwaltung
    1.2.1 Einzelne Datensätze löschen
    1.2.2 Gesamten Speicher löschen
  1.3 Statistik
    1.3.1 Auflistung der vorhandenen Datensätze mit Startzeit
    1.3.2 freier Speicher
2 Direktsprung nach 1.1.1 möglich
3 Ausschaltsequenz

Der programmiertechnische Aufbau der einzelnen Menüsegmente ist grundsätzlich gleich: Zuerst wird der jeweilige Titel in die erste LCD-Zeile geschrieben, dann mittels der Routine LCD_write_Menü (1806-1820) in die zweite Zeile die Tastenbeschriftung „OK ESC Up Down“ zur Navigation ausgegeben. Es beginnt eine Schleife, die über die Routine Wait_for_Taster auf Tastendruck wartet. Mit JB-Befehlen wird anhand eines Tastendrucks zu einem weiteren Menüsegment, oder zu der gewählten Routine gesprungen. Ist das Sprunglabel weiter als 127 Byte Opcode[1] von dem JB-Befehl entfernt, so wird nach dem Druck des entsprechenden Tasters ein hier notwendiger AJMP- oder LJMP-Befehl durch jetzt verwendeten JNB-Befehle nicht übersprungen.

Die Datenstruktur und ihre Routinen

Datenstruktur

Die gewählte Datenstruktur erlaubt variable Datensatzlänge. Dadurch bedingt müssen zu Beginn des Datensatzes Verwaltungsdaten wie eine fortlaufende Nummerierung und die 1. Adresse nach diesem Datensatz gespeichert werden.

Adressenoffset	**Bedeutung**
00h	fortlaufende Nummerierung
01h	1. Adresse nach diesem Datensatz - HighByte
02h	1. Adresse nach diesem Datensatz - LowByte
03h	Start-Tag der Messung
04h	Start-Monat der Messung
05h	Start-Stunde der Messung
06h	Start-Minute der Messung

1 Opcode: Binäre Anweisungsfolge, die die CPU verarbeiten kann

Adressenoffset	Bedeutung
07h	Messintervall-Länge - Stunde
08h	Messintervall-Länge - Minute
09h	Messintervall-Länge - Sekunde
0Ah	1. Temperaturmessung - HighByte
0Bh	1. Temperaturmessung - LowByte
0Ch	2. Temperaturmessung - HighByte
0Dh	2. Temperaturmessung - LowByte
...	...

Datensatz / Letzten Datensatz finden

Um die Adresse eines Datensatzes zu finden, wird dessen Nummer im RAM DSNummer übergeben. Die Routine setzt den Datenpointer und den RAM DS_Zähler, der die Anzahl der überprüften Datensätze speichert, auf den Anfangswert Null; das Kontrollbit DS_gefunden wird gelöscht(1693-1696). Nun folgt eine Schleife, die jeweils an der aktuellen Adresse ein Byte liest und seinen Inhalt vergleicht. Ist dieser FFh, so folgen keine weiteren Datensätze und die Schleife wird verlassen.
Im anderen Falle (1711) wird DS_Zähler inkrementiert (1712) und jetzt ein Vergleich mit der gewünschten Datensatznummer vorgenommen. Fällt dieser positiv aus, so wird die Schleife verlassen und das Bit DS_gefunden zur Meldung an die aufrufende Routine gesetzt. Der Datenpointer steht in diesem Fall auf der Anfangsadresse des gesuchten Datensatzes.
Fällt dieser negativ aus, so werden die folgenden 2 Bytes, die laut oben stehender Tabelle die erste Adresse nach diesem Datensatz enthalten, ausgelesen und der Datenpointer auf diesen Wert (1728-1738) gesetzt. Anschließend beginnt die Überprüfung der Schleife von neuem.
Um den letzten Datensatz zu finden (1667-1680) wird DSNummer auf FFh gesetzt, da dies garantiert die letzte Datensatznummer ist. Die Anzahl der vorhandenen Datensätze wird in das RAM Anzahl_DS übertragen.

Datensätze hinzufügen

Diese Subroutine sucht wie oben mit der oben beschriebenen Routine die erste freie Adresse nach dem letzten Datensatz und schreibt folgend die restlichen 9 Verwaltungsbytes nach der obengenannten Tabelle. Der Datenpointer steht hiernach auf der 1. Adresse nach den Verwaltungsbytes, so dass die Temperaturdaten direkt angefügt werden können.

Freier Speicher

Der verbleibende freie Speicher wird durch einfache Subtraktion der letzten benutzten Adresse vom Gesamtspeicher mit 4096 Byte ermittelt (1742-1764). Die Rückgabe des Wertes erfolgt in R67.

Anzeige der vorhandenen Datensätze mit Startzeit

Die Routine Show_DataSet_Data (1408-1509) zeigt die vorhandenen Datensätze mit ihrer Startzeit an und wird als Auswählroutine zum Löschen einzelner Datensätze, sowie im Statistik-Menüpunkt verwendet.

Die gesammte Routine besteht aus einer Schleife, deren weiterer Verlauf sich durch Betätigen der Tasters bestimmt. Zuerst wird die Anzahl der vorhandenen Datensätze, sowie die Anfangsadresse des mit dem RAM DS_Nummer gewählten Datensatzes in den Datenpointer übertragen (1416-1420). Wurde durch die aufgerufene Routine Find_Nummer nicht mindestens der vorher als Initialwert gesetzte Datensatz Nummer 1 gefunden, und damit das Bit DS_gefunden gesetzt (1423), so ist der Speicher leer. Dieses wird durch eine Meldung auf dem LCD bekannt gegeben und eine kleine Schleife, die auf den Tastendruck „OK“ wartet, verlässt nach Setzten des Meldungsbits SDSD_Back diese Routine.
Sind Datensätze vorhanden (1441), so wird die Nummer des Datensatzes mit Startdatum und Uhrzeit ausgegeben (1442-1472). Die nun folgende innere Schleife (1474-1507) wartet auf einen Tastendruck. Vor einem De- oder Inkrementieren des RAMs DSNummer wird über CJNE-Befehle die aktuelle Nummer mit 00h für das unteren Ende, und Anzahl_DS für das obere Ende vergleichen, so dass sich DSNummer (bei vorausgesetzter richtiger Initialisierung) nur zwischen 01h und der Nummer des letzten Datensatzes bewegen kann (1474-1505). Anschließend wird zum Anfang der Routine gesprungen (1492, 1503). Wurde der Taster OK gedrückt, so wird die Schleife verlassen (1506-1509).

Alles löschen

Zuerst wird in Zeile 1519 der Interrupt für die serielle Schnittstelle über das betreffende Bit im SRF-Register IE abgestellt, denn der Löschvorgang darf nicht durch Zugriffe auf den Speicher durch eine Interruptroutine gestört werden. Die Ergebnisse wären unsinnig; außerdem würde der Datenpointer verstellt werden und die Löschung damit nicht vollständig ermöglichen.
Während des Löschens soll in der unteren Zeile des LCD-Display ein Fortschrittsbalken angezeigt werden, dessen Stand sich nach der aktuellen Löschposition richtet. Zuerst wird in den Zeilen 1523 bis 1529 das Display gelöscht und die erste Zeile mit dem Anzeigentext beschrieben. Der Datenpointer wird auf die Anfangsadresse 0000h, der „Strich“Zähler R0 auf Null gesetzt. Es folgt in den Zeilen 1535 bis 1569 eine Schleife, die jeweils ein Byte(FFh) schreibt, und damit ein Zelle löscht. Unter der Berücksichtigung von 2 Zeichen für den Text OK bleiben bei Fertigstellung 14 Zeichen à 5 Striche (=70) in der 2. LCD-Zeile zur Darstellung der Balkenanzeige. Der Gesamtspeicher der beiden E²PROMs beträgt 4096 Byte; ein ganzzahliges Teilungsverhältnis ist also nur auf Umwegen möglich. Hierzu nutze ich alle 64 Bytes einen Strich, sodass ich für 4kB 12 4/5 Kästchen (=64 Striche) zur Anzeige zu Verfügung habe. Das Teilungsverhältnis von 64 wird in den Zeilen 1537-1539 durch Maskieren der unteren 6 Bits des DPL-Registers in Verbindung mit einem CJNE-Befehl realisiert, der den Graphikteil überspringt, wenn nicht eine durch 63 teilbare Adresse gelöscht wurde. Wenn doch, so wird R0 inkrementiert und durch 5 geteilt. Der im Akkumulator enthaltene ganzzahlige Anteil bestimmt die Anzahl der vollen Kästchen; der im Hilfsakku B enthaltene Rest kann sofort als ASCII-Zeichen an das LCD gesendet werden, da sein Wert mit der Anzahl der darzustellenden Striche übereinstimmt (siehe Deklaration in Zeilen 197 bis 215). Nach Löschung der gesamten 4 kB sind also 12 4/5 Kästchen beschrieben. Der Nutzer könnte sich jetzt wundern, dass der letzte Strich zum vollen 13. Kästchen nicht erscheint und einen Programmfehler vermuten. Deshalb wird in Zeilen 1574 bis 1596 eine Wartezeit von 2 Sekunden eingebaut (die in etwa dem Zeitbedarf des Löschens von 64 Bytes entspricht) und das 13. Kästchen ausgefüllt, sowie die Tastenbezeichnung OK ausgegeben. Eine kleine Schleife wartet auf diesen Tastendruck und vor dem Rücksprung wird der Interrupt der seriellen Schnittstelle wieder freigegeben.

Datensatz einzeln löschen

Dem Benutzer soll die Möglichkeit gegeben werden, Datensätze einzeln löschen zu können. Hierzu ist nach der Auswahl des zu löschenden Datensatzes der benötigte Löschalgorithmus sehr komplex, da

1. die fortlaufende Nummerierung geändert werden muss,
2. die Verkettung der Datensätze durch die Angabe der 1. Adresse nach jedem Datensatz (s. Kap. Datenstruktur) angeglichen werden muss, und
3. die evtl. nachfolgenden Datensätze im Speicher nach vorne verschoben werden müssen.

Diese Aufgaben löst die Routine TeilDatenbestand_löschen (1094-1306). Hier werden natürlich oft die Lese- und Schreibroutinen für den E²PROM-Speicher benötigt. Ein ACALL-Befehl benötigt nur 2 Byte Programmspeicher, kann aber auch nur Unterroutinen im selben 2kB-Programmspeicher-Segment aufrufen. Ein LCALL-Befehl hat diese Einschränkung nicht, benötigt aber 3 Byte. Aufgrund der Lage der E²PROM-Routinen im Programmspeicher wären bei allen E²PROM-Aktionen in dieser Routine LCALLs nötig. Um Speicherplatz zu sparen und ACALLs benutzten zu können, sind am Ende der Routine Hilfslabel eingebaut, die mit LJMP-Befehlen zu den Unterroutinen weiterspringen. Diese Hilfslabel können nun von der Löschroutinen mit ACALLs angesprungen werden, so dass sich trotz dem durch die LJMP-Befehle benötigten 12 Bytes eine Einsparung ergab.
Am Anfang der Routine steht die Auswahl des zu löschenden Datensatzes (1100-1111). Wird ein Datensatz ausgewählt, so stellt das Programm noch eine Sicherheitabfrage mit Anzeige der gewählten Nummer (1114-1129). Wenn diese Frage mit „Ja“ beantwortet wird, so beginnt die eigentliche Löschroutine damit, das getestet wird, ob der gewählte Datensatz der Letzte ist (1132-1135). Wenn ja, dann wird dieser einfach mit FFh überschrieben, da weitere Änderungen oder Verschiebungen nicht notwendig sind. Um festzustellen, wann de Datensatz vollständig überschrieben ist, wir nach jedem gelöschten Byte 2 Überprüfungen gemacht. Als erstet wird getestet, ob das eben gelöschte Byte an der letzten Adresse im Speicher steht (1145-1151). Ein Inkrementieren des Datenpointer würde dann die 12 LSBs wieder auf 0 springen lassen und damit das erste Byte im Speicher adressieren, dessen Löschung aber natürlich nicht gewünscht ist und die Datenstruktur zerstören würde. Ist dies nicht der Fall, so wird das nächste Byte eingelesen und getestet, ob es schon FFh enthält. In diesem Fall kann schon vermutet werden, dass der Datensatz vollständig überschrieben wurde und der Datenpointer in einen freien Bereich des E²PROM zeigt. Da aber Fehlinterpretationen durch einzelne FFhs in den Datensätzen, die z.B. in den Temperaturdaten vorkommen können, auszuschließen, wird das vorher auf 10 initialisierte Register R1 dekrementiert. Diese Register zählt nun rückwärts die aufeinanderfolgenden FFhs; beim Finden eines „Nicht-FFhs“ wird dieses gelöscht (1156, 1136-1143) und das Register wieder auf 10 gesetzt. Nach dem Lesen von 10 aufeinanderfolgenden FFhs ist der nachfolgende Speicher als leer zu anzusehen und die Routinen wird verlassen (1167).
Ist der gewählte Datensatz nicht der letzte (1171), so sind einige Änderungen, Berechnungen und Verschiebungen notwendig, um die Datenstruktur konsistent zu erhalten. Wegen der Rechenoperationen wird zu Beginn über das PSW die Registerbank 3 ausgewählt (1173-1174). Hier können die Register ohne Folgen für andere Routinen verändert werden. Die notwendigen Aktionen sind:

1. Die nachfolgenden Datensatz an die erste Adresse diese Datensatzes kopieren
2. Dabei die fortlaufenden Nummerierung und die Verkettungsadresse richtig ändern
3. Den noch beschriebenen Bereich hinter dem letzten Datensatz löschen.

Der Aufbau der Routine ist in 2 verschachtelte Schleifen gegliedert: Die innere

kümmert sich um das Verschieben der einzelnen Bytes eines Datensatzes, die äußere betrachtet den Datensatz als Ganzes und sorgt für die Änderungen an Nummerierung und Verknüpfung. Die Bedeutung der verwendeten Register sei hier zum besseren Verständnis kurz dargestellt.

R10	enthält die zu überschreibende Adresse
R32	enthält die zu lesende Adresse
R54	enthält die erste Adresse nach dem gerade gelesenen Datensatz
R76	enthält den Adressenversatz, um den die Bytes verschoben werden

Zu Beginn wird die Anfangsadresse des zu löschenden Datensatzes in R10 gesichert (1175-1176), dann die Anfangsadresse des folgenden, zu verschiebenden Datensatzes in R32 geschrieben. Es beginnt die äußere Schleife, die zuerst das Nummerierungsbyte des zu verschiebenden Datensatzes einliest, es dekrementiert und an die in R10 gespeicherte Adresse schreibt (1184-1190). Es folgt die Änderung der Verkettung, denn die folgenden 2 Byte müssen an die neue Position im Speicher angepasst werden. Hierzu wird zuerst der Versatz (Offset) berechnet, um die der gesammte Datensatz verschoben wird, und dieses dann von dem alten Verkettungszeiger subtrahiert wird.
Der Offset ergibt sich auf der Differenz zwischen R32 und R10, die ja die Lese- und Schreibadresse enthalten, und wird in R76 gespeichert (1201-1209). Aus dem zu verschiebenden Datensatz wird der „alte“ Verkettungszeiger gelesen und in R54 gespeichert (1211-1216). Der Datenpointer wird gleich auf die Adresse der ersten „richtigen“ Information, dem Start_Tag, inkrementiert und in R32 gespeichert (1218-1222).
Nun erfolgt die Berechnung des neuen Endes des zu verschiebenden Datensatzes durch die Subtraktion
alte Position (=R54) - Offset (=R76) = neue 1. Adresse nach diesem Datensatz (1231-1241). Zur Zwischenspeicherung des LowBytes, das zwar zuerst berechnet werden muss, aber erst als zweites nach dem HighByte geschrieben werden darf, wird hier geschickt der Stack ausgenutzt (1234, 1240).
Anschließend beginnt die innere Schleife, die Byte für Byte die Nutzdaten verschiebt und deren Abbruchbedingung R32 = R54 ist, also das Erreichen der letzten Leseadresse (1267-1273). Nach Beendigung der inneren Schleife wird geprüft, ob noch ein weiterer Datensatz folgt (1275-1279). Wenn ja, wird die äußere Schleife mit ihren Änderungen und Verschiebungsaktionen erneut gestartet (1279), sonst muss nur noch der Rest des letzten Datensatzes, der nicht durch seine neue Position überschrieben wurde, gelöscht werden. Dies geschieht in einer kleinen Schleife, die solange FFhs schreibt, bis sie am ehemaligen Ende des Datensatzes mit der in R54 gespeicherten Adresse angekommen ist (1287-1293). Zum Abschluss wird das ursprüngliche PSW vom Stack zurückgePOPt, so dass die Aufruf aktive Registerbank wieder ausgewählt, und die veränderten Flags wie z.B. das Carry wiederhergestellt sind.

Menü Messwerterfassung

Der Menüpunkt Messwerterfassung (409-1009)ist der komplexeste. Er vereint die Funktionen, die die Hauptaufgabe des MC-Thermometers darstellen: Temperaturdaten in regelmäßigen Abständen automatisch aufzunehmen und diese abzuspeichern.

Eingabe der Daten

Zuerst fragen die Messwert-Erfassungsmenüs 1-3 die Intervallzeit, das Startdatum sowie die Startuhrzeit ab. Die Eingabe erfolgt durch zwei Taster, die mit einem Pfeil nach oben und einem Pfeil nach rechts beschriftet sind. Die Eingabe der Daten, wie auch ihre Speicherung im RAM, erfolgt jeweils Stellenweise . Die aktuelle Stelle, die gerade geändert werden kann, blinkt dabei im Takt von 0,5 s. Der Taster mit dem Pfeil nach oben inkrementiert diese Stelle. Kommt es dadurch zu einem Wert, der nicht mehr sinnvoll ist (z.B. 64 Sekunden), so wird diese Stelle entsprechend richtig neuinitialisiert. Der Pfeil nach rechts verschiebt die aktuelle Änderungsstelle eine Position nach rechts. Wie diese Aufgabenstellung gelöst wurde, soll jetzt beispielhaft an der Eingabe des Zeitintervalls (408-517) dargestellt werden.
Zuerst werden die RAMs auf sinnvolle Werte vorinitialisiert (416ff. : RAM_Init), und der Text „Abstand“ in die 1. LCD-Zeile, sowie die Tasterbezeichnungen „OK Zurück ↑ → in die 2. Zeile geschrieben (422-427). Die Auswahl, welche Stelle jeweils geändert werden soll und blinkt, erfolgt über die Bits X_OnChange in den RAMs DIVERS2 und DIVERS3, die mit ihren Adressen 23h und 24h (66-85) im bitadressierbaren Bereich (siehe Seite 5) des MC liegen. Mit der Änderung der Stunden wird begonnen (428). Es folgt eine Schleife, die die aktuellen RAM-Inhalte auf dem LCD ausgibt (430-444). Nun wird auf einen Tastendruck gewartet und dieser mit JB- und JNB-Befehlen in schon geschriebener Weise ausgewertet. Bei der Inkrementierung wird zunächst über Die X_OnChange-Bits zur richtigen Stelle gesprungen, an der dann auf Überlauf (z.B. Sekunden = 60) getestet wird (z.B. 485-493). Abschließend erfolgt der Rücksprung in die Schleife (493). Vor dem Weiterspringen mit „OK“ in das nächste Dateneingabemenü erfolgt noch ein Test, ob die Intervalllänge 0s beträgt. Hierzu werden die RAMs im Akku addiert, und auf 0 getestet (449-452). Eine Fehl-Interpretation aufgrund eines Überlauf des Akkus, der dann auch 00h enthalten würde, brauch nicht z.B. durch das Carry-Bit berücksichtigt werden, da der maximale Wert 59+59+99=217 beträgt.

Überprüfung, ob Speicherplatz ausreicht

Eine Besonderheit stellt die Eingabe der gewünschten Anzahl der aufzunehmenden Messwerte dar. Bevor mit Druck auf „OK“ die Eingabe akzeptiert wird, muss überprüft werden, ob noch genügend Platz vorhanden ist. Die hierfür notwendigen mathematischen Schritte bestehen aus allen Grundrechenarten. Da aber jeweils 16-Bit-Zahlen verrechnet werden, der MC aber nur 8-Bit Mathematik-Befehle kennt, müssen die Operationen jeweils Byte für Byte ausgeführt werden. Dies produziert einen recht umfangreichen Programmcode, mit vielen Möglichkeiten. Die auskommentierten „SNAP“-Aufrufe zeugen noch von intensiver Fehlersuche. Der Assembler-Text (741-885) ist ausführlich kommentiert, auf eine nochmalige Beschreibung wird deshalb verzichtet.

Timer 0 und sein Interrupt

Zum exakten Auslösen der Messwertaufnahme wird der noch unbenutzte Timer 0 benutzt. In dem hier verwendetet Modus 1 (s. Datenblatt [3] Seite 10) besteht er aus zwei verknüpften 8 Bit-Registern, die im Systemtakt inkrementiert werden und löst beim Überlauf einen Interrupt aus. An der Einsprungadresse 0023h (154) steht ein LJMP-Befehl, der auf die Interrupt-Serviceroutine zeigt.
Das Meßwert_Menü5 (935-1009) beinhaltet den Programmcode, der während der Messwertaufnahme ausgeführt wird. Zuerst wird der Interrupt der ser. Schnittstelle deaktiviert, da eine Unterbrechung der Messungen nicht möglich sein soll (942). Nach dem Beschriften des LCD und Erstellen eines neuen Datensatzes (950) wird ein

24Bit-Zähler R345 mit „create_24Bit_Counter“ aus den Eingaben über die Intervall-Länge erstellt (956). Weil hierdurch die Register R3 bis R5 der aktuellen Registerbank 0 nicht verändert werden sollen, wird über das PSW auf Bank 1 umgeschaltet (952-965). Dieses Vorgehen hat den Vorteil, dass man diese 24Bit-Zahl durch einfache Subtraktion dekrementieren kann. Würde man die in den Bytes Intervall_H bis Intervall_S gespeicherten Werte benutzten, so wären aufwendige Unterscheidungen und Tests nötig, um einen Überlauf (bzw. Unterlauf) z.B. der Stunden auf die anderen Bytes weiterzureichen. Anschließend die erste Messung gemacht (967) und der Timer 0 sowie sein Interrupt freigegeben (969). Es folgt eine Schleife, die auf einen „Stop“-Tastendruck wartet. Die zwei Timerbytes werden nun fortlaufend inkrementiert, laufen nach 50 ms über und die Interruptserviceroutine Timer_Int (2734-2830) wird aufgerufen. Zuerst wird der Timer mit den Startwerten, in Gegensatz zum Timer1 für die ser. Schnittstelle, manuell nachgeladen (2744f.). Dann wird auf Registerbank 1 umgeschaltet und das Register R7 heruntergezählt und bei Bedarf mit dem Wert 20 nachgeladen (2758f). Hierdurch ergibt sich, dass bei der eingestellten Interruptrate von $\frac{1}{50}\frac{Interrupt}{ms}=20\frac{Interrupts}{s}$ jede Sekunde die eigentliche Serviceroutine ausgeführt wird. In diesem Sekundentakt wird nun der 24Bit-Sekundenzähler R345 dekrementiert und so die Intervallzeit abgewartet (2785-2798). Ist der Inhalt von R345 = 0 (2800-2806), ist diese abgelaufen, wird der Zähler R345 neu gesetzt (2809) und eine Messung ausgelöst. Dieses Auslösen geschieht über das Bit „mach_Messung“, dass von der aktuellen Warteschleife „Wait_for_Taster laufend abgefragt wird (2006). Ist es von der Interruptroutine gesetzt, so wird es wieder gelöscht und die Routine „Temp_Aufnahme“ aufgerufen.

Messwerterfassung

Zur Aufnahme der Messwerte werden zuerst die Anzahl-Bytes dekrementiert (2138-2160) und bei einem Stand von 0000h eine Fertig-Meldung ausgegeben. Sonst wird vom DS1620 die aktuelle Temperatur eingelesen (2163-2177). Die Überprüfung des Bits „Show_Temp“ ist nötig, da auch im Ruhezustand die Temperatur aktualisiert werden soll. Dann ist dieses Bit gesetzt, und der Timer ebenfalls aktiviert, aber durch dieses Bit wird die Speicherung, die dann nicht nötig ist, übersprungen. Dies soll aber während der Messwertaufnahme gerade nicht geschehen, deshalb wird ab dem Label „Save_Temp“ (2186) weitergemacht. Hier werden die Temperaturdaten an die richtige Stelle im E²PROM geschrieben, und die nächste Adresse nach diesem Datensatz berichtigt gespeichert (2193-2222).

Excel - Visual Basic for Applications

DynamicLinkLibrary-Aufrufe

Mit dem Buch [4] wird auf CD eine DLL mitgeliefert, die Funktionen und Prozeduren (in Visual Basic „Sub“ genannt) zum Ansprechen der seriellen Schnittstelle beinhaltet. Diese Routinen müssen vor ihrer Verwendung in VBA mit dem declare-Befehl erst eingebunden werden, dann können sie wir normale Routinen verwendet werden.

Das Programm

Das VBA-Makro teilt sich in mehrere Unterroutinen auf. Anfangs werden über die Sub DiagrammeLöschen evtl. vorhandene Diagramme entfernt. Die Sub ReadoutData öffnet die Schnittstelle (79), sendet das Anforderungsbyte „M“ aus (81) und empfängt den gesamten Speicherinhalt und sortiert ihn richtig in die Tabelle „Roh-Daten“ ein (82-110). Anschließend werden diese umgewandelt und die Zeitpunkte der Messungen anhand der Startzeit und der Intervalllänge berechnet. Diese werden dann in die Tabelle „Temperaturen“ zusammen mit der Temperatur in °C geschrieben (162-186). Dann wird für jeden Datensatz ein Diagramm erstellt (188-232), bei dem die Einteilung der Zeitachse durch den ersten und letzten Messzeitpunkt bestimmt wird (221-222).

Nachwort

Nach Beendigung der Arbeit möchte ich hier ein kleines Resumé geben. Sowohl das Konzipieren, Berechnen, Tüfteln und Conrad- und Reichelt-Kataloge wälzen im Hardwarebereich, als auch die Softwareentwicklung in Assembler, die den zeitintensivsten Part darstellte, machte Spaß. Das Damoklesschwert der 4096 Byte Programmspeicher, die bei diesem MC auch nicht mit Tricks überwunden werden können, forderte bei jeder Routine, bei jeder Zeile die Frage: Geht das auch kürzer ? Dennoch war gerade das Austüfteln trickreicher Sparmaßnahmen ein zusätzlicher Ansporn und machte Spaß. Glücklicherweise passten auch die ASCII-Texte für das LCD vollständig in den Programmspeicher und mussten nicht in Teile des E²PROM-Speichers ausgelagert werden. Doch fast keine Routine lief auf Anhieb, meist folgte dem Programmieren die unumgängliche Fehlersuche, die je nach Fehler auch mal eine Stunde in Anspruch nahm. Besonders „gefährlich“ waren Änderungen der Register in Bank 0. Teilweise wirkten die sich dann quer durch das Programm aus, so das eine Routine, an der man überhaupt nichts geändert hatte, plötzlich ganz tolle Dinge anstellte... Als Erfahrung für nächste Projekte habe ich viel zum Umgang mit MCs und der Programmierung in Assembler gelernt. So werde ich in Zukunft, wie auch schon Ansatzweise in der SNAP-Routine, kurze Labelbezeichnungen verwenden, um die Lesbarkeit erhöhen. Auch durch Aufführung der verwendeten Register zu Anfang jeder Routine hoffe ich einen besseren Durchblick besonders in Registerbank 0 zu bekommen und die Register, auch aus Gründen des kürzeren Programmcodes an Stelle der vielen PUSH A und POP A häufiger zu Nutzen.

Anhang

Schaltplan

Layout

Oberseite:

Unterseite:

Assembler-Quellcode

```
1  ; ------------------------------------------------------------------------------
2  ; -                                                                            -
3  ; - Mikrocontroller gesteuertes Thermometer mit Langzeit-Meßwerterfassung,    -
4  ; - LCD, menügesteuerter Benutzerführung, Interface zum PC zur (grafischen)   -
5  ; - Auswertung der Daten in Microsoft Excel mit Visual Basic for Applications -
6  ; -                                                                          -
7  ; ------------------------------------------------------------------------------
8
9  ; Facharbeit von Ralf Wilke am Werner-Jaeger-Gymnasium Nettetal  Stufe 12
10 ; Informatik
11
12
13 ; ------------------------------------------------------------------------------
14 ; ------------------------------------------------------------------------------
15 ; ------------------------------------------------------------------------------
16
17
18 ; Facharbeit-Teil Definitionen
19 ; Ralf Wilke
20 ;
21
22 INCLUDE 89C4051.mc                  ; Die SFR-Adressen des AT89C4051 einbinden
23
24   ; Port 3
25
26         SDA              BIT   P3.2 ; I2C-Bus-Leitung SDA an Port 3 definieren
27         DS_DQ            BIT   P3.3 ; DS1620-DQ-Leitung an Port 3 definieren
28         ENBL_SEL0        BIT   P3.5 ; Die Enable-Leitungen für den 138 definieren
29         ENBL_SEL1        BIT   P3.4
30
31
32   ; RAM INPUT_TASTER und seine Bits deklarieren
33
34         TASTER0          BIT   04h
35         TASTER1          BIT   05h
36         TASTER2          BIT   06h
37         TASTER3          BIT   07h
38         INPUT0           BIT   01h
39         INPUT1           BIT   02h
40         INPUT2           BIT   03h
41         INPUT3           BIT   04h
42         INPUT_TASTER     EQU   20h
43
44       ; RAM OUTPUT_STATUS und seine Bits deklarieren
45         SCL1             BIT   08h
46         SCL2             BIT   09h
47         DS_CLK           BIT   0Ah
48         DS_RST           BIT   0Bh
49         LCD_RS           BIT   0Ch
50         LCD_RW           BIT   0Dh
51         ;                BIT   0Eh
52         ;                BIT   0Fh
53         OUTPUT_STATUS    EQU   21h
54
55   ; RAM DIVERS1 und seine Bits deklarieren
56         2nd_row          BIT   10h
57         Ser_ready        BIT   11h
58         Ser_Int_arrived  BIT   12h
59         Ser_Int_Lock     BIT   13h
60         I2C_NoAck        BIT   14h
61         LCD_Zeile        BIT   15h
62         DPTR_is_I2C            BIT   16h
63         SCL_NS           BIT   17h   ; SCL-No IC Selection
64         DIVERS1          EQU   22h
65
66   ; RAM DIVERS2 und seine Bits deklarieren
67         Int_is_running         BIT   18h
68         Intervall_H_OnChange   BIT   19h
69         Intervall_M_OnChange   BIT   1Ah
70         Intervall_S_OnChange   BIT   1Bh
71         Start_Day_OnChange     BIT   1Ch
72         Start_Month_OnChange   BIT   1Dh
73         Start_Hour_OnChange    BIT   1Eh
74         Start_Minute_OnChange  BIT   1Fh
75         DIVERS2                EQU   23h
76
```

```
77    ; RAM DIVERS3 und seine Bits deklarieren
78          Anzahl_L_OnChange            BIT   20h
79          Anzahl_H_OnChange            BIT   21h
80          blink                  BIT   22h
81          mach_Messung           BIT   23h
82          Show_Temp              BIT   25h
83          DS_gefunden                  BIT   26h
84          SDSD_back              BIT   27h
85          DIVERS3                EQU   24h
86
87
88
89  ; RAM deklarieren
90
91          wait_time              EQU   30h
92          wait_time1             EQU   31h
93          wait_time_025s         EQU   32h
94          LCD_ADDRESS                  EQU   33h
95          DEV_SEL                EQU   34h
96          I2C_W_Byte             EQU   35h
97          I2C_R_Byte             EQU   36h
98          DSNummer               EQU   37h
99
100         Start_Day              EQU   38h
101         Start_Month                  EQU   39h
102         Start_Hour             EQU   3Ah
103         Start_Minute           EQU   3Bh
104         Intervall_H                  EQU   3Ch
105         Intervall_M                  EQU   3Dh
106         Intervall_S                  EQU   3Eh
107         Anzahl_L               EQU   3Fh
108
109         Anzahl_H               EQU   40h
110         Anzahl_L_ges           EQU   41h
111         Anzahl_H_ges           EQU   42h
112         DPH_Save               EQU   43h
113         DPL_Save               EQU   44h
114         DPH_next_Add           EQU   45h
115         DPL_next_Add           EQU   46h
116         Temp_L                 EQU   47h
117         Temp_H                 EQU   48h
118         Blink1                 EQU   49h
119         Blink2                 EQU   4Ah
120         Anzahl_DS              EQU   4Bh
121         DS_Zähler              EQU   4Ch
122 ;                              EQU   4Dh
123 ;                              EQU   4Eh
124 ;                              EQU   4Fh
125 ;!!!!!!! 50h ist die 1. Stackadresse !!!!!!!!!
126
127
128
129 (0000h):      ; Init-Teil
130         ANL P3, #00001011b           ; Auf Port 3 alles auf 0, nur SDA auf 1,
131         ORL P3, #00001000b           ; da dann inaktiv
132         MOV SP, #4Fh                 ; Stack-Pointer auf 4Fh
133         SJMP (000Eh)                 ; Überspringen der Interruptvektoradressen
134 (000Bh):      LJMP Timer_Int                   ; Sprung zur Interrupt-Routine
135                                      ; Timer 0
136
137 (000Eh):      MOV R0, #7fh                     ; Hier ist zwischen den Interrupteinsprung-
138                                      ; Adressen noch etwas Programmspeicherplatz,
139                                      ; der ausgenutzt werden kann, da die hier
140                                      ; liegenden Interrupts nicht aktiviert werden
141
142 Del_RAM_WDH:MOV @R0, #00h                    ; Den ganzen RAM löschen
143         DJNZ R0, Del_RAM_WDH
144
145         SETB DS_DQ                   ; DS1620-Leitungen in den Ausgangs-
146         CLR DS_RST                   ; zustand bringen.
147         SETB DS_CLK
148
149         MOV DPTR, #0000h
150         LCALL WR_Status_Ausgang            ; Ausgeben
151
152         SJMP (0026h)
153
154 (0023h):      LJMP Ser_Int                     ; Sprung zur Interrupt-Routine
155                                      ; Ser.-Schnittstelle
```

```
156
157
158 ;-------------------------------------------------------------------------------
159 (0026h):                                       ; Beginn des Programms, erste Adresse nach
160                                   ; nach der Ser.-Interruptvektoradresse
161         LCALL I2C_STOP
162         MOV DPH, 00001000b
163         LCALL I2C_STOP
164
165
166
167 ;-------------------------------------------------------------------------------
168 ; Facharbeit-Teil    LCD initialisieren
169 ;
170 ; Ralf Wilke
171
172 LCD_Init:     MOV A, #00111000b                ; Function set: 2 line, 5x7 dots
173         CLR LCD_RS
174         LCALL LCD_WR                ; Kommando ausgeben
175         MOV A, #00001100b                ; Display ON/OFF Control:
176                                   ; Display on, cursor off, blink off
177         LCALL LCD_WR                ; LCD_RS und LCD_RW sind noch 0
178         MOV A, #00000001b                ; Clear display
179         LCALL LCD_WR
180         MOV A, #00000110b                ; Entry Mode Set:
181         LCALL LCD_WR                ; increment mode, entire shift off
182         MOV A, #00000010b                ; Return Home
183         LCALL LCD_WR
184
185         MOV LCD_ADDRESS, #01000000b  ; "Pfeil nach oben" in den Charakter-
186         LCALL LCD_ADD_CG_SET         ; Generator-RAM schreiben
187         CLR A                        ; ASCII-Code 00h
188         LCALL LCD_S_Char
189         MOV A, #00000100b
190         LCALL LCD_S_Char
191         MOV A, #00001110b
192         LCALL LCD_S_Char
193         MOV A, #00010101b
194         LCALL LCD_S_Char
195         MOV R2, #4
196         MOV A, #00000100b
197 Pfeil_oben_WDH:
198         LCALL LCD_S_Char
199         DJNZ R2, Pfeil_oben_WDH
200                                   ; Zeichen für einen Vorschrittsbalken
201         MOV R0, #4                   ; 4 Zeichen auf diese Weise erstellen
202         MOV A, #11110000b                ; Bit 4 stellt die 1. Spalte dar
203
204         MOV LCD_ADDRESS, #01001000b  ; Adresse der 1. linken Spalte auswählen
205 Spalte_WDH:   LCALL LCD_ADD_CG_SET         ; Adresse setzen
206         MOV R2, #8                   ; 8 Punkte = 8 Zeilen pro Spalte setzen
207
208 Zeichen_WDH:LCALL LCD_S_Char
209         DJNZ R2, Zeichen_WDH
210         XCH A, LCD_ADDRESS           ; Akku und LCD_ADDRESS tauschen, um durch
211         ADD A, #00001000b                ; Addition von 1000b das nächste Zeichen
212         XCH A, LCD_ADDRESS           ; zu Adressieren, dann zurücktauschen
213         SETB C                       ; Carry setzten, um mit dem RotateRightCarry
214         RRC A                        ; die nächste Spalte zusätzlich zu füllen
215         DJNZ R0, Spalte_WDH
216
217
218 ;-------------------------------------------------------------------------------
219 ; Facharbeit-Teil    Serielle Schnittstelle initialisieren
220 ;
221 ; Ralf Wilke
222 ;
223
224 Ser_Init:     ORL PCON, #10000000b              ; SMOD = 1    = > Takt/16
225         ORL TMOD, #21h               ; Timer 1 als Baudratengenerator nutzen, Modus2
226                                   ; Timer 0 als Zeitgeber nutzen, Modus1
227         MOV TH0, #3Ch                ; alle 50ms einen Interrupt aufrufen
228         MOV TL0, #B0h
229         MOV TH1, #243                ; Preload Wert = 4807 Baud
230         MOV TL1, #243
231
232         CLR TCON.4
233         SETB TCON.6                      ; Timer 1 starten
234         MOV SCON, #01010000b         ; Ser. Control:
```

```
235                                     ; Mode 1, REN = 1, TI = 0, RI = 0
236
237 ;-------------------------------------------------------------------------------
238 ; Facharbeit-Teil    Interrupts aktivieren
239 ;
240 ; Ralf Wilke
241 ;
242
243
244
245 Int_Init:     ORL IE, #10010000b                ; Interrupt generell und für serielle
246                                     ; Schnittst. freigeben
247
248         LCALL Pwr_on_Msg            ; PowerOn-Meldung ausgeben
249
250
251
252 ;       MOV DPTR, #0000h
253 ;init_WDH2:   LCALL EEPROM_Byte_Write
254 ;       INC I2c_W_BYte
255 ;       MOV A, #47
256 ;init_WDH:    INC DPTR
257 ;       DJNZ A, INIT_WDH
258 ;       MOV A, DPH
259 ;       CJNE A, #00010000b, init_WDh2
260
261 ;-------------------------------------------------------------------------------
262 ; Facharbeit-Teil    Standard-Ausgabe mit aktueller Temperatur
263 ;
264 ; Ralf Wilke
265 ;
266
267
268 Main:         MOV LCD_ADDRESS, #00        ; 1. LCD Zeile löschen
269         LCALL LCD_ADD_SET
270         MOV R7, #16
271         MOV A, #" "
272 Main_WDH:     LCALL LCD_S_Char
273         DJNZ R7, Main_WDH
274
275
276
277
278
279         SETB LCD_Zeile              ; Die 2. Zeile beschreiben
280         MOV DPTR, #Main_DB          ; mit dem Menü-Zugang, Direkt-Speicherung,
281         LCALL Line2LCD
282
283 Main_Loop:    SETB Show_Temp                    ; Die Temperaturanzeige aktivieren
284         LCALL Wait_for_Taster       ; Auf Tastendruck warten
285         CLR Show_Temp               ; Die Temp.Anzeige deaktivieren, weil evtl.
286                                     ; in ein Menü gesprungen wird und dort Text
287                                     ; Überschrieben würde
288
289         JB Taster0, Main_Menü1      ; An den Anfang des Menüs springen
290         JNB Taster1, Main_not_Save
291         AJMP Meßwert_Menü1          ; Direkt zur Messwertaufnahme
292 Main_not_Save:
293         JB Taster3, Sure_End        ; Taster 3 springt zum Sure_End-Menü
294         SJMP Main_Loop
295
296
297 ;-------------------------------------------------------------------------------
298 ; Facharbeit-Teil    Main_Menü1 - Messwerterfassung
299 ;
300 ; Ralf Wilke
301 ;
302
303 Main_Menü1:   MOV DPTR, #Main_Menü1_DB    ; Anfangsadresse auswählen
304         LCALL LCD_write_Menü        ; Menü ausgeben
305 Main_Menü1_Loop:
306         LCALL Wait_for_Taster       ; Auf Tastendruck warten
307         JNB Taster0, Main_Menü1_weiter; Wenn nicht Taster 0 gedrückt wurde den
308                                     ; LJMP-Befehl überspringen
309         LJMP Meßwert_Menü1          ; Zur Meßwerterfassung springen
310 Main_Menü1_weiter:
311         JB Taster1, Main            ; Zum Standard-Ausgabe springen
312         JB Taster2, Main_Menü2      ; Taster 3 "Up" springt zum Menü 2
313         SJMP Main_Menü1_Loop
```

```
314
315 ;-------------------------------------------------------------------------------
316 ; Facharbeit-Teil    Main_Menü2 - Datenverwaltung
317 ;
318 ; Ralf Wilke
319 ;
320
321 Main_Menü2:   MOV DPTR, #Main_Menü2_DB     ; Anfangsadresse auswählen
322         LCALL LCD_write_Menü         ; Menü ausgeben
323 Main_Menü2_Loop:
324         LCALL Wait_For_Taster        ; Auf Tastendruck warten
325         JNB Taster0, Main_Menü2_weiter; Wenn nicht Taster 0 gedrückt wurde den
326                                      ; LJMP-Befehl überspringen
327         LJMP Datenverw._Menü1        ; Zur Datenverwaltung springen
328 Main_Menü2_weiter:
329         JB Taster1, Main_Back        ; Zur Standard-Ausgabe springen
330         JB Taster2, Main_Menü3       ; Taster 3 "Up" springt zum Menü 3
331         JB Taster3, Main_Menü1       ; Taster 4 "Down" springt zum Menü 1
332         SJMP Main_Menü2_Loop
333
334 ;-------------------------------------------------------------------------------
335 ; Facharbeit-Teil    Main_Menü3 - Statistik
336 ;
337 ; Ralf Wilke
338 ;
339
340 Main_Menü3:   MOV DPTR, #Main_Menü3_DB     ; Anfangsadresse auswählen
341         LCALL LCD_write_Menü         ; Menü ausgeben
342 Main_Menü3_Loop:
343         LCALL Wait_For_Taster        ; Auf Tastendruck warten
344         JNB Taster0, Main_Menü3_weiter; Wenn nicht Taster 0 gedrückt wurde den
345                                      ; LJMP-Befehl überspringen
346         LJMP Statistik_Menü1         ; Zur Statistik springen
347 Main_Menü3_weiter:
348         JB Taster1, Main_Back        ; Zur Standard-Ausgabe springen
349         JB Taster3, Main_Menü2       ; Taster 4 "Down" springt zum Menü 2
350         SJMP Main_Menü3_Loop
351
352 Main_back:    AJMP Main
353
354 ;-------------------------------------------------------------------------------
355 ; Facharbeit-Teil    "Wirklich ausschalten?"-Menü
356 ;
357 ; Ralf Wilke
358 ;
359
360 Sure_End:
361         MOV DPTR, #Sure_End_DB       ; DPTR mit Anfangsadresse beschreiben
362         LCALL 2Lines2LCD             ; 2 Zeilen ausgeben
363 Sure_End_Loop:
364         LCALL Wait_for_Taster        ; Auf Tastendruck warten
365         JB Taster2, Power_Off        ; Zur Power_Off-Routine springen
366         JNB Taster3, Sure_End_Loop   ; Wenn nicht Taster 3 gedrückt wurde, nochmal
367         LJMP Main                    ; sonst springe zum Main-Menü
368
369
370
371
372 ;-------------------------------------------------------------------------------
373 ; Facharbeit-Teil    Befehle, die vor dem Abschalten der Stromversorgung ausgeführt
374 ; werden müssen
375 ;
376 ; Ralf Wilke
377 ;
378
379 Power_Off:  ; < Befehle >
380
381             MOV A, #"O"                  ; Meldung OFF ausgeben
382         LCALL Ser_Send
383         MOV A, #"F"
384         LCALL Ser_Send
385         LCALL Ser_Send
386         LCALL CRLF
387
388         MOV DPTR, #Power_Off_DB          ; DPTR mit Anfangsadresse der Letzten
389                                      ; Meldung beschreiben
390         LCALL 2Lines2LCD
391         LJMP The_End
392
```

```
393
394 ;-------------------------------------------------------------------------------
395 ; Facharbeit-Teil    Meßwert_Menü - Rücksprung"routine"
396 ;
397 ; Ralf Wilke
398 ;
399
400 Main_Menü1_back:
401         ANL DIVERS2, #00000001b           ; Intervall_X_OnChange deaktivieren
402         LJMP Main_Menü1             ; Innerhalb des Menüs Datenverwaltung werden
403                                     ; JB-Befehle benötigt, die nur +-128 Byte
404                                     ; springen können. Diese Befehle springen hierher,
405                                     ; und von hier geht's dann weiter
406
407
408 ;-------------------------------------------------------------------------------
409 ; Facharbeit-Teil    Meßwert_Menü1 - Zeitintervall
410 ;
411 ; Ralf Wilke
412 ;
413
414 Meßwert_Menü1:
415
416 RAM_Init:           MOV A, #1                   ; Einige RAMs vorbelegen
417         MOV Start_Day, A
418         MOV Start_Month, A
419         MOV Intervall_S, A
420         MOV Anzahl_L, A
421
422         MOV DPTR, #Meßwert_Menü1_DB  ; Anfangsadresse auswählen
423         CLR LCD_Zeile                ; 1. Zeile auswälen
424         LCALL Line2LCD               ; Zeile ausgeben
425         MOV DPTR, #Meßwert_Eingabe   ; OK, Back, Up, - > Anfangsadresse auswählen
426         SETB LCD_Zeile               ; 2. Zeile auswählen
427         LCALL Line2LCD               ; Zeile ausgeben
428         SETB Intervall_H_OnChange    ; Stunden als Änderung auswählen
429
430 Meßwert_Menü1_Loop:
431         MOV LCD_ADDRESS, #7          ; LCD-Cursor auf Adresse 7 setzten
432         LCALL LCD_ADD_SET
433         MOV A, Intervall_H           ; Stunden_Intervall ausgeben
434         LCALL LCD_Send_Zahl
435         MOV A, #"h"                       ; h = hour
436         LCALL LCD_S_Char
437         MOV A, Intervall_M           ; Minuten_Intervall ausgeben
438         LCALL LCD_Send_Zahl
439         MOV A, #"m"                       ; m = minute
440         LCALL LCD_S_Char
441         MOV A, Intervall_S           ; Sekunden_Intervall ausgeben
442         LCALL LCD_Send_Zahl
443         MOV A, #"s"                       ; s = second
444         LCALL LCD_S_Char
445
446         LCALL Wait_For_Taster        ; Auf Tastendruck warten
447         JNB Taster0, not_Meßwert_Menü2; Wenn Taster 0 = OK nicht gedrückt wurde,
448                                     ; überspringen
449         MOV A, Intervall_H           ; Feststellen, ob alle Werte 0 sind
450         ADD A, Intervall_M           ; Dazu alle addieren und wenn dann
451         ADD A, Intervall_S
452         JZ not_Meßwert_Menü2         ; der Akku 0 ist, den Menüaufruf überspringen
453
454         ANL DIVERS2, #11110001b           ; sonst das Blinken abstellen und
455         AJMP Meßwert_Menü2           ; ein Menü weiterspringen
456 not_Meßwert_Menü2:
457         JB Taster1, Main_Menü1_back  ; Taster 1 = Back : Zur Main-Ebene springen
458         JNB Taster2, Meßwert_Menü1_not_up
459                                     ; Wenn Taster 3 = Up nicht gedrückt wurde, die
460                                     ; Inkrementierung überspringen
461         JNB Intervall_H_OnChange, Meßwert_Menü1_Inc_Minute
462                                     ; Wenn nicht das Bit gesetzt ist, überspringen
463         INC Intervall_H              ; sonst die Stunden inkrementieren
464         PUSH A                       ; Akku retten
465         MOV A, Intervall_H           ; Intervall_H in den Akku
466         CJNE A, #23, Meßwert_Menü1_not_overflow_H
467                                     ; Wenn durch die Erhöhung nicht 60 erreicht wurde,
468         MOV Intervall_H, #0          ; das Rückstellen überspringen
469 Meßwert_Menü1_not_overflow_H:
470         POP A
471 Meßwert_Menü1_Inc_Minute:
```

```
472         JNB Intervall_M_OnChange, Meßwert_Menü1_Inc_Sekunde
473                                     ; Wenn nicht das Bit gesetzt ist, überspringen
474         INC Intervall_M             ; sonst die Minuten inkrementieren
475         PUSH A                      ; Akku retten
476         MOV A, Intervall_m          ; Intervall_M in den Akku
477         CJNE A, #60, Meßwert_Menü1_not_overflow_M
478                                     ; Wenn durch die Erhöhung nicht 60 erreicht wurde,
479         MOV Intervall_M, #0         ; das Rückstellen überspringen
480 Meßwert_Menü1_not_overflow_M:
481         POP A
482 Meßwert_Menü1_Inc_Sekunde:
483         JNB Intervall_S_OnChange , Meßwert_Menü1_Loop
484                                     ; Wenn nicht das Bit hesetzt ist, überspringen
485         INC Intervall_S             ; sonst die Sekunden inkrementieren
486         PUSH A                      ; Akku retten
487         MOV A, Intervall_S          ; Intervall_S in den Akku
488         CJNE A, #60, Meßwert_Menü1_not_overflow_S
489                                     ; Wenn durch die Erhöhung nicht 60 erreicht wurde,
490         MOV Intervall_S, #0         ; das Rückstellen überspringen
491 Meßwert_Menü1_not_overflow_S:
492         POP A
493         SJMP Meßwert_Menü1_Loop           ; Tastendruck fertig abgearbeitet, zurück
494                                     ; in die Schleife
495 Meßwert_Menü1_not_up:
496         JNB Taster3, Meßwert_Menü1_Loop
497                                     ; Wenn der Pfeil nach rechts gewählt wurde,
498                                     ; also eine Stelle weiter rechts geändert
499                                     ; werden soll...
500         JNB Intervall_H_OnChange, Meßwert_Menü1_Minute
501                                     ; Wenn das Bit gesetzt ist, überspringen
502         CLR Intervall_H_OnChange    ; sonst die Minuten auswählen
503         SETB Intervall_M_OnChange
504         SJMP Meßwert_Menü1_Loop           ; Tastendruck ist abgearbeitet, zurück in den Loop
505
506 Meßwert_Menü1_Minute:
507         JNB Intervall_M_OnChange, Meßwert_Menü1_Sekunde
508                                     ; Wenn das Bit nicht gesetz ist, überspringen
509         CLR Intervall_M_OnChange    ; sonst die Sekunden auswählen
510         SETB Intervall_S_OnChange
511         AJMP Meßwert_Menü1_Loop           ; Tastendruck ist abgearbeitet, zurück in den Loop
512 Meßwert_Menü1_Sekunde:
513         CLR Intervall_S_OnChange
514         SETB Intervall_H_OnChange   ; Die Stunde wieder auswählen
515         AJMP Meßwert_Menü1_Loop           ; Tastendruck ist abgearbeitet, zurück in den Loop
516
517
518
519 ;-------------------------------------------------------------------------------
520 ; Facharbeit-Teil   Meßwert_Menü2 - Startzeit eingeben Tag und Monat
521 ;
522 ; Ralf Wilke
523 ;
524
525 Meßwert_Menü2:
526         MOV DPTR, #Meßwert_Menü2_DB  ; Anfangsadresse auswählen
527         CLR LCD_Zeile               ; 1. Zeile auswälen
528         LCALL Line2LCD              ; Zeile ausgeben
529         MOV DPTR, #Meßwert_Eingabe  ; OK, Back, Up, - > Anfangsadresse auswählen
530         SETB LCD_Zeile              ; 2. Zeile auswählen
531         LCALL Line2LCD              ; Zeile ausgeben
532         SETB Start_Day_OnChange           ; Tag als Änderung auswählen
533
534 Meßwert_Menü2_Loop:
535         MOV LCD_ADDRESS, #10        ; LCD-Cursor auf Adresse 10 setzten
536         LCALL LCD_ADD_SET
537         PUSH A                      ; Akku retten
538         MOV A, Start_Day            ; Start_Day ausgeben
539         LCALL LCD_Send_Zahl
540         MOV A, #"."
541         LCALL LCD_S_Char
542         MOV A, Start_Month          ; Start_Month ausgeben
543         LCALL LCD_Send_Zahl
544         MOV A, #"."
545         LCALL LCD_S_Char
546
547         POP A                       ; Ursprungsakku wiederherstellen
548         LCALL Wait_For_Taster       ; Auf Tastendruck warten
549         JNB Taster0, not_Meßwert_Menü3; Wenn Taster 0 = OK nicht gedrückt wurde,
550                                     ; überspringen
```

```
551         ANL DIVERS2, #11001111b           ; sonst das Blinken abstellen und
552         AJMP Meßwert_Menü3          ; ein Menü weiterspringen
553 not_Meßwert_Menü3:
554         JB Taster1, Meßwert_Menü1_back; Taster 1 = Back : Zum Meßwert_Menü1 springen
555         JNB Taster2, Meßwert_Menü2_not_up
556                                     ; Wenn Taster 3 = Up nicht gedrückt wurde, die
557                                     ; Inkrementierung überspringen
558         JNB Start_Day_OnChange, Meßwert_Menü2_Inc_Month
559                                     ; Wenn nicht das Bit gesetzt ist, überspringen
560         INC Start_Day               ; sonst den Tag inkrementieren
561         PUSH A                      ; Akku retten
562         MOV A, Start_Day            ; Start_Day in den Akku
563         CJNE A, #32, Meßwert_Menü2_not_overflow_Day
564                                     ; Wenn durch die Erhöhung nicht 32 erreicht wurde,
565         MOV Start_Day, #1                ; das Rückstellen überspringen
566 Meßwert_Menü2_not_overflow_Day:
567         POP A
568 Meßwert_Menü2_Inc_Month:
569         JNB Start_Month_OnChange, Meßwert_Menü2_Loop
570                                     ; Wenn nicht das Bit gesetzt ist, überspringen
571         INC Start_Month             ; sonst die Monate inkrementieren
572         PUSH A                      ; Akku retten
573         MOV A, Start_Month          ; Start_Month in den Akku
574         CJNE A, #13, Meßwert_Menü2_not_overflow_Month
575                                     ; Wenn durch die Erhöhung nicht 13 erreicht wurde,
576         MOV Start_Month, #1         ; das Rückstellen überspringen
577 Meßwert_Menü2_not_overflow_Month:
578         POP A
579         SJMP Meßwert_Menü2_Loop           ; Tastendruck fertig abgearbeitet, zurück
580                                     ; in die Schleife
581 Meßwert_Menü2_not_up:
582         JNB Taster3, Meßwert_Menü2_Loop
583                                     ; Wenn der Pfeil nach rechts gewählt wurde,
584                                     ; also eine Stelle weiter rechts geändert
585                                     ; werden soll...
586         JNB Start_Day_OnChange, Meßwert_Menü2_Month
587                                     ; Wenn das Bit gesetzt ist, überspringen
588         CLR Start_Day_OnChange      ; sonst die Monate auswählen
589         SETB Start_Month_OnChange
590         SJMP Meßwert_Menü2_Loop           ; Tastendruck ist abgearbeitet, zurück in den Loop
591
592 Meßwert_Menü2_Month:
593         CLR Start_Month_OnChange
594         SETB Start_Day_OnChange           ; Den Tag wieder auswählen
595         AJMP Meßwert_Menü2_Loop           ; Tastendruck ist abgearbeitet, zurück in den Loop
596
597
598 ;-------------------------------------------------------------------------------
599 ; Facharbeit-Teil    Meßwert_Menü1 - Rücksprung"routine"
600 ;
601 ; Ralf Wilke
602 ;
603
604 Meßwert_Menü1_back:
605         ANL DIVERS2, #00000001b           ; Blinken abstellen
606         AJMP Meßwert_Menü1          ; Innerhalb des Menüs Meßwert_Menü2 werden
607                                     ; JB-Befehle benötigt, die nur +-128 Byte
608                                     ; springen können. Diese Befehle springen hierher,
609                                     ; und von hier geht's dann weiter
610
611
612
613 ;-------------------------------------------------------------------------------
614 ; Facharbeit-Teil    Meßwert_Menü3 - Startzeit eingeben Stunde und Minute
615 ;
616 ; Ralf Wilke
617 ;
618
619 Meßwert_Menü3:
620         MOV DPTR, #Meßwert_Menü3_DB  ; Anfangsadresse auswählen
621         CLR LCD_Zeile               ; 1. Zeile auswälen
622         LCALL Line2LCD              ; Zeile ausgeben
623         MOV DPTR, #Meßwert_Eingabe   ; OK, Back, ^ , - > Anfangsadresse auswählen
624         SETB LCD_Zeile              ; 2. Zeile auswählen
625         LCALL Line2LCD              ; Zeile ausgeben
626         SETB Start_Hour_OnChange    ; Stunde als Änderung auswählen
627
628 Meßwert_Menü3_Loop:
629         MOV LCD_ADDRESS, #10        ; LCD-Cursor auf Adresse 10 setzten
```

```
630         LCALL LCD_ADD_SET
631         PUSH A                      ; Akku retten
632         MOV A, Start_Hour                ; Start_Hour ausgeben
633         LCALL LCD_Send_Zahl
634         MOV A, #"h"                      ; h = hour
635         LCALL LCD_S_Char
636         MOV A, Start_Minute         ; Start_Minute ausgeben
637         LCALL LCD_Send_Zahl
638         MOV A, #"m"                      ; m = minute
639         LCALL LCD_S_Char
640
641         POP A                       ; Ursprungsakku wiederherstellen
642         LCALL Wait_For_Taster       ; Auf Tastendruck warten
643         JNB Taster0, not_Meßwert_Menü4; Wenn Taster 0 = OK nicht gedrückt wurde,
644                                     ; überspringen
645         ANL DIVERS2, #00111111b          ; sonst das Blinken abstellen und
646         AJMP Meßwert_Menü4          ; ein Menü weiterspringen
647 not_Meßwert_Menü4:
648         JB Taster1, Meßwert_Menü2_back; Taster 1 = Back : Zum Meßwert_Menü2 springen
649         JNB Taster2, Meßwert_Menü3_not_up
650                                     ; Wenn Taster 3 = Up nicht gedrückt wurde, die
651                                     ; Inkrementierung überspringen
652         JNB Start_Hour_OnChange, Meßwert_Menü3_Inc_Minute
653                                     ; Wenn nicht das Bit gesetzt ist, überspringen
654         INC Start_Hour              ; sonst die Stunde inkrementieren
655         PUSH A                      ; Akku retten
656         MOV A, Start_Hour                ; Start_Hour in den Akku
657         CJNE A, #24, Meßwert_Menü3_not_overflow_Hour
658                                     ; Wenn durch die Erhöhung nicht 24 erreicht wurde,
659         MOV Start_Hour, #0          ; das Rückstellen überspringen
660 Meßwert_Menü3_not_overflow_Hour:
661         POP A
662 Meßwert_Menü3_Inc_Minute:
663         JNB Start_Minute_OnChange, Meßwert_Menü3_Loop
664                                     ; Wenn nicht das Bit gesetzt ist, überspringen
665         INC Start_Minute            ; sonst die Minuten inkrementieren
666         PUSH A                      ; Akku retten
667         MOV A, Start_Minute         ; Start_Minute in den Akku
668         CJNE A, #60, Meßwert_Menü3_not_overflow_Minute
669                                     ; Wenn durch die Erhöhung nicht 603 erreicht wurde,
670         MOV Start_Minute, #0        ; das Rückstellen überspringen
671 Meßwert_Menü3_not_overflow_Minute:
672         POP A
673         SJMP Meßwert_Menü3_Loop          ; Tastendruck fertig abgearbeitet, zurück
674                                     ; in die Schleife
675 Meßwert_Menü3_not_up:
676         JNB Taster3, Meßwert_Menü3_Loop
677                                     ; Wenn der Pfeil nach rechts gewählt wurde,
678                                     ; also eine Stelle weiter rechts geändert
679                                     ; werden soll...
680         JNB Start_Hour_OnChange, Meßwert_Menü3_Minute
681                                     ; Wenn das Bit gesetzt ist, überspringen
682         CLR Start_Hour_OnChange          ; sonst die Minute auswählen
683         SETB Start_Minute_OnChange
684         SJMP Meßwert_Menü3_Loop          ; Tastendruck ist abgearbeitet, zurück in den Loop
685
686 Meßwert_Menü3_Minute:
687         CLR Start_Minute_OnChange
688         SETB Start_Hour_OnChange    ; Die Stunde wieder auswählen
689         AJMP Meßwert_Menü3_Loop          ; Tastendruck ist abgearbeitet, zurück in den Loop
690
691
692 ;-------------------------------------------------------------------------------
693 ; Facharbeit-Teil   Meßwert_Menü2 - Rücksprung"routine"
694 ;
695 ; Ralf Wilke
696 ;
697
698 Meßwert_Menü2_back:
699         ANL DIVERS2, #00000001b          ; Blinken abstellen
700         LJMP Meßwert_Menü2          ; Innerhalb des Menüs Meßwert_Menü3 werden
701                                     ; JB-Befehle benötigt, die nur +-128 Byte
702                                     ; springen können. Diese Befehle springen hierher,
703                                     ; und von hier geht's dann weiter
704
705
706
707
708 ;-------------------------------------------------------------------------------
```

```
709 ; Facharbeit-Teil    Meßwert_Menü4 - Anzahl der Messungen
710 ;
711 ; Ralf Wilke
712 ;
713
714 Meßwert_Menü4:
715         MOV DPTR, #Meßwert_Menü4_DB  ; Anfangsadresse auswählen
716         CLR LCD_Zeile                ; 1. Zeile auswälen
717         LCALL Line2LCD               ; Zeile ausgeben
718         MOV DPTR, #Meßwert_Eingabe   ; OK, Back, ^ , - > Anfangsadresse auswählen
719         SETB LCD_Zeile               ; 2. Zeile auswählen
720         LCALL Line2LCD               ; Zeile ausgeben
721         SETB Anzahl_H_OnChange       ; Anzahl_High als Änderung auswählen
722
723 Meßwert_Menü4_Loop:
724         MOV LCD_ADDRESS, #8          ; LCD-Cursor auf Adresse 8 setzten
725         LCALL LCD_ADD_SET
726         MOV A, Anzahl_H              ; Anzahl_H ausgeben
727         LCALL LCD_Send_Zahl
728         MOV A, Anzahl_L              ; Anzahl_L ausgeben
729         LCALL LCD_Send_Zahl
730
731         LCALL Wait_For_Taster        ; Auf Tastendruck warten
732         JB Taster0, MM4_check        ; Wenn Taster 0 = OK gedrückt wurde, weiter
733 MM4_3:  AJMP not_Meßwert_Menü5       ; sonst: überspringen
734
735 MM4_check:    MOV A, Anzahl_H                  ; Feststellen, ob alle Werte 0 sind
736         ADD A, Anzahl_L              ; Dazu alle addieren und wenn dann
737         JZ MM4_3                     ; der Akku 0 ist, den Menüaufruf überspringen
738
739         ; Testen, ob überhaupt genug Platz im EEPROM ist
740
741         PUSH PSW
742         ORL PSW, #00011000b          ; Registerbank 3 auswählen
743         MOV R2, DIVERS3
744         ANL DIVERS3, #11111100b           ; sonst das Blinken abstellen und
745         LCALL Get_Free_MEM           ; Freier Speicher ist jetzt in R67
746
747         ; Anzahl_L und Anzahl_H enthalten jetzt die gewünschte Anzahl an Messwerten
748         ; Doch diese Byte sind dezimal geteilt; sie müssen also erst in eine 16Bit-
749         ; Zahl gewandelt werden
750
751         MOV B, #100                       ; B-Register mit 100 zum multiplizieren laden
752         MOV A, Anzahl_H              ; High_Byte in den Akku
753         MUL AB                       ; und mit 100 multiplizieren
754                                      ; Das Ergebnis der Multiplikation steht jetzt
755                                      ; in BA, also das Low-Byte im Akku und das
756                                      ; High-Byte im B-Register
757         ADD A, Anzahl_L              ; Jetzt zum Lowbyte im Akku das dezimale Low-
758                                      ; byte addieren
759         MOV R5, A                    ; das Ergebnis in R5 sichern
760         CLR A                        ; den Akku löschen
761         ADDC A, B                    ; und einen evtl. Übertrag mit dem B-Register
762                                      ; in den leeren Akku schreiben
763         MOV R4, A                    ; Diesen in R4 sichern
764
765         ; R45 enthält jetzt die binäre Anzahl der gewünschten Messungen
766         ; Nun muss zur Speicherplatzermittlung der Wert verdoppelt werden, da ein
767         ; Meßwert 2 Byte gross ist, und 10 Byte Protokoll addiert werden.
768
769         MOV A, R5                    ; LowByte aus R5 in den Akku
770         ADD A, R5                    ; und R5 addieren
771         MOV R5, A                    ; das Ergebnis wieder in R5
772         MOV A, R4                    ; und das Highbyte holen,
773         ADDC A, R4                   ; mit Übertrag mit sich selbst addieren
774         MOV R4, A                    ; und zurück
775         MOV A, R5                    ; jetzt noch die 10 Byte für das
776         ADD A, #10                   ; Datenstrukturprotokoll addieren
777         MOV R5, A
778         MOV A, R4
779         ADDC A, #0
780         MOV R4, A                    ; fertig
781 ;       LCALL snap
782
783
784
785         ; R45 enthält jetzt den für diesen Datensatz benötigten Speicher
786         ; jetzt R67 von R45 abziehen, d.h. testen, ob R67 < R45
787
```

```
788         CLR C                       ; Das Carry löschen
789         MOV A, R7                   ; Lowbyte R7 in den Akku
790         SUBB A, R5                  ; und Lowbyte R5 abziehen
791         MOV A, R6                   ; Das genaue Ergebnis ist nicht wichtig,
792                                     ; daher kann der Akku direkt wieder mit dem
793                                     ; Highbyte R6 beschrieben werden,
794         SUBB A, R4                  ; von dem dann Highbyte R4 mit Übertrag
795                                     ; abgezogen wir
796
797 ;       LCALL snap
798         JNC go_to_MM5               ; Wenn das Carry nicht gesetzt ist, war
799                                     ; R45 < R67, also noch Platz im Speicher
800                                     ; sonst die maximale Anzahl ermitteln
801
802
803         CLR C                       ; dazu die Operationen von oben rückwerts
804         MOV A, R7                   ; machen: Vom freien Speicher-Lowbyte R7
805         SUBB A, #10                      ; 10 abziehen
806         MOV R3, A                   ; und das Ergebnis in R3 sichern
807         MOV A, R6                   ; einen etvl. Übertrag auch von HighByte R6
808         SUBB A, #0                  ; abziehen
809
810 ;       LCALL SNAP
811
812         ; jetzt A:R7 durch 2 teilen. Das geschieht durch einmaliges Schieben nach
813         ; rechts
814
815         CLR C                       ; Carry löschen, damit nichts oben in den
816         RRC A                       ; Akku reinrotiert wird
817         MOV R0, A                   ; Das Highbyte in R0 sichern
818         MOV A, R3                   ; Das Lowbyte aus R3 holen
819         RRC A                       ; auch rechts rotieren, und an die 7 Stelle
820                                     ; das Carry, das die ehem. 0. Stelle des
821                                     ; High-Bytes enthält, schreiben
822         MOV R1, A                   ; dann in R1 sichern
823 ;       LCALL snap
824
825         ; R01 enthält jetzt die max. Anzahl als 16-Bit-Zahl
826
827         ; Jetzt diese Zahl wieder in eine dezimalgeteilte 2Byte-Zahl wandeln
828         ; und diese in Anzahl_L/H zurückschreiben
829
830         MOV Anzahl_H, #00                ; Anzahl_H auf 0
831         CLR C                       ; Carry löschen
832 MM4_4:        MOV A, R1                ; Das Low-Byte in den Akku
833         MOV Anzahl_L, A             ; und auch in Anzahl_L
834         SUBB A, #100                ; 100 davon abziehen
835         MOV R1, A                   ; und diesen Wert in R1 zurücksichern
836         MOV A, R0                   ; jetzt den etvl. Übertrag auch von
837         SUBB A, #0                  ; HighByte R0 abziehen
838         MOV R0, A
839
840         JC MM4_6                    ; Ist das Carry jetzt gesetzt, war R01 < 100
841                                     ; dann ist die Wandlung fertig; wiederspringen
842         INC Anzahl_H                ; sonst Anzahl_H um 1 erhöhen
843         SJMP MM4_4                  ; und nochmal testen
844
845 MM4_6:        MOV DPTR, #MM4_nicht_genug_Speicher
846         LCALL 2Lines2LCD            ; Text ausgeben
847         MOV LCD_Address, #45h
848         LCALL LCD_Add_Set
849         MOV A, Anzahl_H             ; Anzahl_L/H ausgeben
850         LCALL LCD_Send_Zahl
851         MOV A, Anzahl_L
852         LCALL LCD_Send_Zahl
853
854
855 MM4_loop2:          LCALL wait_For_Taster       ; Auf OK warten
856         JNB Taster2, MM4_loop2
857         MOV DIVERS3, R2
858         POP PSW
859         AJMP Meßwert_Menü4          ; wieder zurück
860
861
862
863
864 go_to_MM5:          POP PSW
865
866         AJMP Meßwert_Menü5          ; ein Menü weiterspringen
```

```
867 not_Meßwert_Menü5:
868         JB Taster1, Meßwert_Menü3_back; Taster 1 = Back : Zum Meßwert_Menü3 springen
869         JNB Taster2, Meßwert_Menü4_not_up
870                                     ; Wenn Taster 3 = Up nicht gedrückt wurde, die
871                                     ; Inkrementierung überspringen
872         JNB Anzahl_H_OnChange, Meßwert_Menü4_Inc_Low
873                                     ; Wenn nicht das Bit gesetzt ist, überspringen
874         INC Anzahl_H                ; sonst die Anzahl_H inkrementieren
875         MOV A, Anzahl_H             ; Anzahl_H in den Akku
876         CJNE A, #100, Meßwert_Menü4_not_overflow_High
877                                     ; Wenn durch die Erhöhung nicht 100 erreicht wurde,
878         MOV Anzahl_H, #0            ; das Rückstellen überspringen
879 Meßwert_Menü4_not_overflow_High:
880 Meßwert_Menü4_Inc_Low:
881         JNB Anzahl_L_OnChange, MM4_gotoLoop
882                                     ; Wenn nicht das Bit gesetzt ist, überspringen
883         INC Anzahl_L                ; sonst die Anzahl_L inkrementieren
884         MOV A, Anzahl_L             ; Anzahl_L in den Akku
885         CJNE A, #100, Meßwert_Menü4_not_overflow_Low
886                                     ; Wenn durch die Erhöhung nicht 603 erreicht wurde,
887         MOV Anzahl_L, #0            ; das Rückstellen überspringen
888 Meßwert_Menü4_not_overflow_Low:
889 MM4_gotoLoop:
890         AJMP Meßwert_Menü4_Loop           ; Tastendruck fertig abgearbeitet, zurück
891                                     ; in die Schleife
892 Meßwert_Menü4_not_up:
893         JNB Taster3, MM4_gotoLoop
894                                     ; Wenn der Pfeil nach rechts gewählt wurde,
895                                     ; also eine Stelle weiter rechts geändert
896                                     ; werden soll...
897         JNB Anzahl_H_OnChange, Meßwert_Menü4_Low
898                                     ; Wenn das Bit gesetzt ist, überspringen
899         CLR Anzahl_H_OnChange       ; sonst die Anzahl_Low auswählen
900         SETB Anzahl_L_OnChange
901         SJMP MM4_gotoLoop                ; Tastendruck ist abgearbeitet, zurück in den Loop
902
903 Meßwert_Menü4_Low:
904         CLR Anzahl_L_OnChange
905         SETB Anzahl_H_OnChange      ; Die Anzahl_High wieder auswählen
906         SJMP MM4_gotoLoop                ; Tastendruck ist abgearbeitet, zurück in den Loop
907
908
909
910
911
912 ;-------------------------------------------------------------------------------
913 ; Facharbeit-Teil    Meßwert_Menü3 - Rücksprung"routine"
914 ;
915 ; Ralf Wilke
916 ;
917
918 Meßwert_Menü3_back:
919         ANL DIVERS3, #11111100b           ; Blinken abstellen
920         AJMP Meßwert_Menü3          ; Innerhalb des Menüs Meßwert_Menü4 werden
921                                     ; JB-Befehle benötigt, die nur +-128 Byte
922                                     ; springen können. Diese Befehle springen hierher,
923                                     ; und von hier geht's dann weiter
924
925 Main_Menü_back:
926         CLR IE.1                    ; Interrupt Timer 0 aus
927         CLR TCON.4                  ; Timer 0 stoppen
928         SETB IE.4                   ; Interrupt ser. Schnittstelle an
929
930         AJMP Main                   ; Innerhalb des Menüs Datenverwaltung werden
931                                     ; JB-Befehle benötigt, die nur +-128 Byte
932                                     ; springen können. Diese Befehle springen hierher,
933                                     ; und von hier geht's dann weiter
934
935 ;-------------------------------------------------------------------------------
936 ; Facharbeit-Teil    Meßwert_Menü5 - Anzahl der Messungen
937 ;
938 ; Ralf Wilke
939 ;
940
941 Meßwert_Menü5:
942         CLR IE.4                    ; Interrupt der ser. Schnittstelle ausschalten
943         MOV Anzahl_H_ges, Anzahl_H  ; Anzahl speichern für die LCD-Ausgabe
944         MOV Anzahl_L_ges, Anzahl_L
945
```

```
946         MOV DPTR, #Meßwert_Menü5_DB  ; Anfangsadresse holen
947         SETB LCD_Zeile               ; 2. Zeile wählen
948         LCALL Line2LCD               ; Zeile auf LCD schicken
949
950         LCALL create_new_dataset     ; Neuen Datensatz anlegen
951         PUSH PSW
952         SETB PSW.3                   ; Registerbank 1 auswählen
953         CLR PSW.4
954         ; R345 als 24 Bit Zähler der Sekunden nutzen
955
956         LCALL create_24Bit_counter
957
958
959
960
961 ;       MOV R3, Intervall_H          ; Register initialisieren
962 ;       MOV R4, Intervall_M
963 ;       MOV R5, Intervall_S
964         MOV R7, #20
965         POP PSW                      ; Wieder die Registerbank 0 auswählen
966
967         LCALL Temp_Aufnahme
968         SETB IE.1                    ; Interrupt vom Timer anschalten
969         SETB TCON.4                       ; Timer0 starten
970 Meßwert_Menü5_Loop:
971         LCALL Wait_for_Taster        ; auf Tastendruck warten
972         JNB Taster3, Meßwert_Menü5_Loop
973         MOV A, Anzahl_H              ; Testen, ob Anzahl = 0 ist
974         ADD A, Anzahl_L
975         JZ Main_Menü_Back                 ; Wenn ja, zum Hauptmenü zurückspringen
976         PUSH DPL                     ; sonst Abbrechen-Frage stellen
977         PUSH DPH
978         MOV DPTR, #Meßwert_Stop_Sure_DB
979         LCALL 2Lines2LCD
980         POP DPH
981         POP DPL
982 Meßwert_Abb_Loop:
983         LCALL Wait_For_Taster
984         JB Taster2, Meßwert_Menü_Ende; Wenn Taster2 = "Ja, Abbrechen"
985                                      ; gedrückt wurde, zum Ende springen, sonst
986                                      ; die 2. Zeile wiederherstellen
987         JNB Taster3, Meßwert_Abb_Loop; Wenn nicht Taster 3 = Nein gedrückt wurde,
988                                      ; nochmal abfragen
989
990 Meßwert_Menü5_cont:
991         SETB LCD_Zeile               ; 2. LCD_Zeile auswählen
992         MOV DPTR, #Meßwert_Menü5_DB  ; Anfangsadresse holen
993         LCALL Line2LCD               ; Zeile auf LCD schicken
994         LCALL ErfData2LCD
995         SJMP Meßwert_Menü5_Loop           ; und zurück in die Schleife
996
997 Meßwert_Menü_Ende:
998         ; Abbrechen-Routine
999
1000        CLR IE.1                     ; Interrupt Timer 0 ausschalten
1001        SETB IE.4                    ; Interrupt ser. Schnittstelle einschalten
1002
1003        MOV DPTR, #Meßwert_Canceled_DB; DPTR auf Anfangsadresse setzten
1004        LCALL 2Lines2LCD             ; 2 Zeilen auf das LCD schreiben
1005Meßwert_Menü_Ende_Loop:
1006        LCALL Wait_for_Taster        ; Auf Tastendruck warten; nur Taster 3 = OK
1007                                     ; wird akzeptiert
1008        JNB Taster3, Meßwert_Menü_Ende_Loop
1009        AJMP Main                    ; Zurück zum Main-Menü
1010
1011
1012
1013;-------------------------------------------------------------------------------
1014; Facharbeit-Teil   Datenverwaltung_Menü - Rücksprung"routine"
1015;
1016; Ralf Wilke
1017;
1018
1019Main_Menü2_back:
1020        AJMP Main_Menü2              ; Innerhalb des Menüs Datenverwaltung werden
1021                                     ; JB-Befehle benötigt, die nur +-128 Byte
1022                                     ; springen können. Diese Befehle springen hierher,
1023                                     ; und von hier geht's dann weiter
1024
```

```
1025
1026
1027;-------------------------------------------------------------------------------
1028; Facharbeit-Teil   Datenverwaltung_Menü1 - Datenbestand teilweise löschen
1029;
1030; Ralf Wilke
1031;
1032
1033Datenverw._Menü1:
1034       MOV DPTR, #Datenverw._Menü1_DB; Anfangsadresse auswählen
1035       LCALL LCD_write_Menü         ; Menü ausgeben
1036Datenverw._Menü1_Loop:
1037       LCALL Wait_For_Taster        ; Auf Tastendruck warten
1038       JNB Taster0, Datenverw._Menü1_weiter
1039                                    ; Wenn nicht Taster 0 gedrückt wurde den
1040                                    ; LJMP-Befehl überspringen
1041       SJMP TeilDatenbestand_löschen     ; Die Teildatenbestand löschen Routine aufrufen
1042Datenverw._Menü1_weiter:
1043       JB Taster1, Main_Menü2_back  ; Zur Main-Ebene springen
1044       JB Taster2, Datenverw._Menü2 ; Taster 3 "Up" springt zum Datenverw._Menü 2
1045       JB Taster3, Datenverw._Menü2 ; Taster 4 "Down" springt zum Datenverw._Menü 1
1046       SJMP Datenverw._Menü1_Loop
1047
1048
1049
1050
1051;-------------------------------------------------------------------------------
1052; Facharbeit-Teil   Datenverwaltung_Menü2 - Datenbestand löschen
1053;
1054; Ralf Wilke
1055;
1056
1057Datenverw._Menü2:
1058       MOV DPTR, #Datenverw._Menü2_DB; Anfangsadresse auswählen
1059       LCALL LCD_write_Menü         ; Menü ausgeben
1060Datenverw._Menü2_Loop:
1061       LCALL Wait_For_Taster        ; Auf Tastendruck warten
1062       JNB Taster0, Datenverw._Menü2_weiter
1063                                    ; Wenn nicht Taster 0 gedrückt wurde den
1064                                    ; LJMP-Befehl überspringen
1065       AJMP Datenbestand_löschen_sure; Die Datenbestand_löschen-
1066                                    ; Sicherheitsabfrage aufrufen
1067Datenverw._Menü2_weiter:
1068       JB Taster1, Main_Menü2_back  ; Zur Main-Ebene springen
1069       JB Taster2, Datenverw._Menü1 ; Taster 3 "Up" springt zum Datenverw._Menü 1
1070       JB Taster3, Datenverw._Menü1 ; Taster 4 "Down" springt zum Datenverw._Menü 1
1071       SJMP Datenverw._Menü2_Loop
1072
1073
1074
1075;-------------------------------------------------------------------------------
1076; Facharbeit-Teil   "Wirklich alles löschen?"-Menü
1077;
1078; Ralf Wilke
1079;
1080
1081Datenbestand_löschen_Sure:
1082       MOV DPTR, #Sure_Del_All_DB   ; DPTR mit Anfangsadresse beschreiben
1083       LCALL 2Lines2LCD             ; 2 Zeilen ausgeben
1084Datenbestand_löschen_Loop:
1085       LCALL Wait_for_Taster        ; Auf Tastendruck warten
1086       JNB Taster2, Datenbestand_löschen_weiter
1087       AJMP Del_All_Data                 ; Zur Alles_löschen-Routine springen
1088Datenbestand_löschen_weiter:
1089       JB Taster3, Datenverw._Menü1 ; Taster 3 springt zum Datenverw._Menü1 zurück
1090       SJMP Datenbestand_löschen_Loop
1091
1092
1093
1094;-------------------------------------------------------------------------------
1095; Facharbeit-Teil   Teildatenbestand löschen
1096;
1097; Ralf Wilke
1098;
1099
1100TeilDatenbestand_löschen:
1101
1102       MOV DPTR, #OKESCUpDown_DB    ; OK,ESC,Up,Down-Zeile ausgeben
1103       SETB LCD_Zeile               ; in die 2. Ziele
```

```
1104        LCALL Line2LCD
1105
1106        MOV DSNummer, #1            ; Mit dem Auflisten bei Datensatz 1 anfangen
1107
1108TDL_get_Data:
1109        ACALL Show_DS_Data
1110        JNB SDSD_back, TDL_OK
1111        AJMP Main_Menü2
1112
1113
1114TDL_OK:      MOV DPTR, #Sure_Del_DB      ; sonst Sicherheitsfrage stellen
1115        SETB LCD_Zeile              ; Den Text und die Nummer des zu löschenden
1116        LCALL Line2LCD              ; Datensatzes ausgeben
1117        MOV DPTR, #Del_Datensatz_DB
1118        CLR LCD_Zeile
1119        LCALL Line2LCD
1120        MOV LCD_Address, #0Dh
1121        LCALL LCD_Add_Set
1122        MOV A, DSNummer
1123        ACALL LCD_Z
1124TDL_Sure_Loop:
1125        LCALL Wait_for_Taster       ; warten auf Tastendruck
1126        JB Taster2, TDL_löschung    ; Taster 2 = löschen
1127        JB Taster3, TDL_back        ; Taster 3 = nicht löschen, zurück
1128        SJMP TDL_Sure_Loop
1129TDL_back:    AJMP Teildatenbestand_löschen
1130
1131
1132TDL_löschung:
1133        ACALL find_nummer
1134        MOV A, Anzahl_DS            ; Testen, ob der zu löschende Datensatz der
1135        CJNE A, DSNummer, TDL_not_last; letzte ist
1136TDL_last_WDH:
1137        MOV I2C_W_Byte, #FFh        ; Wenn ja, die Nummer mit FFh überschreiben
1138                                    ; um ihn so für die Verwaltung scheinbar zu
1139                                    ; löschen. Der DPTR steht noch von dem
1140                                    ; find_Nummer-Aufruf an der 1. Adresse in
1141                                    ; diesem Datensatz
1142
1143        MOV R1, #10                      ; "FFh"-Zähler auf 10 setzten
1144        ACALL EE_W                  ; FFh schreiben
1145TDL_last_WDH2:
1146        INC DPTR                    ; nächste Adresse
1147        MOV A, DPL
1148        JNZ TDL_last_no_ov
1149        MOV A, DPH
1150        ANL A, #00001111b
1151        JZ TDL_back
1152
1153TDL_last_no_ov:
1154        ACALL EE_R                  ; zum Testen lesen
1155        MOV A, I2C_R_Byte                ; zum Vergleichen in den Akku
1156        CJNE A, #FFh, TDL_last_WDH  ; testen, ob FFh gelesen wurde
1157                                    ; Wenn nicht, die Schleife wiederholen und an diese
1158                                    ; Stelle ein FFh schreiben
1159
1160        DJNZ R1, TDL_last_WDH2      ; Wenn ein FFh vorlag, den "aufeinanderfolgede FFh"-
1161  ; Zähler R1 dekrementieren und das nächste Byte auf FFh-Inhalt testen.
1162  ; Ist es auch FFh, so werden insgesamt 10 Bytes getestet. Sind sie alle FFh,
1163  ; kann der nachfolgende Speicher als leer angesehen werden.
1164  ; Wird aber innerhalb dieser 10 Byte ein NICHT-FFh gefunden, so wird durch den
1165  ; Sprung auf TDL_last_WDH R1 wieder mit dem Wert 5 initialisiert.
1166
1167        SJMP TDL_back
1168
1169
1170
1171TDL_not_last:                               ; Wenn es nicht des letzte Datensatz ist...
1172
1173        PUSH PSW
1174        ORL PSW, #00011000b         ; Registerbank 3 auswählen;
1175        MOV R0, DPL                      ; R10 enthalten die zu löschende Adresse
1176        MOV R1, DPH
1177
1178        INC DSNummer                ; den nächsten Datensatz finden
1179        LCALL Find_Nummer                ; nächsten Datensatz finden
1180        MOV R3, DPH                      ; die Anfangsadresse in R32 speichern
1181        MOV R2, DPL
1182
```

```
1183TDL_next_Datensatz:
1184       ACALL EE_R                  ; Das Zählbyte lesen
1185       MOV I2C_W_Byte, I2C_R_Byte  ; zum Ausgeben vorbereiten
1186       DEC I2C_W_Byte              ; um 1 erniedrigen, um die fortlaufende Nummerierung
1187                                   ; beizubehalten
1188       MOV DPL, R0                      ; die Anfangsadresse des zu löschenden
1189       MOV DPH, R1                      ; Datensatzes holen
1190       ACALL EE_W                  ; Byte schreiben
1191       ; Jetzt ist das "fortlaufende Nummerierungsbyte" geschrieben
1192
1193
1194
1195       MOV DPH, R3                      ; R32 mit der zu lesenden Adresse in den DPTR
1196       MOV DPL, R2
1197
1198       ; R76 = R32 - R10 = Adressenversatz von alter Datensatzstelle zur neunen
1199       ; Datensatzstelle im Speicher
1200
1201       CLR C                       ; Carry löschen
1202       MOV A, R2
1203       SUBB A, R0
1204       MOV R6, A
1205       MOV A, R3
1206       SUBB A, R1
1207       MOV R7, A
1208
1209       ; R76 enthält jetzt das Ergebnis
1210
1211       INC DPTR                    ; Die 1. Adresse nach diesem Datensatz
1212       ACALL EE_R                  ; in R54 speichern
1213       MOV R5, I2C_R_Byte
1214       INC DPTR
1215       ACALL EE_R
1216       MOV R4, I2C_R_Byte
1217
1218       INC DPTR                    ; DPTR auf das Start-Tag-Byte setzten, also
1219                                   ; das erste Byte, das nicht geändert werden
1220                                   ; muss, sondern einfach nur verschoben wird
1221       MOV R3, DPH                      ; Diese Adresse in R32 sichern
1222       MOV R2, DPL
1223       MOV DPH, R1                      ; DPTR mit der Adresse des alten Nummerierungsbytes
1224       MOV DPL, R0                      ; aus R10 setzten
1225       INC DPTR
1226
1227  ; Neue 1. Adressposition nach diesem Datensatz berechnen
1228  ; neue Position = alte Position ( = R54) - Versatz durch Verschieben ( = R76)
1229  ; An die neue Stelle schreiben
1230
1231       CLR C                       ; Carry löschen
1232       MOV A, R4
1233       SUBB A, R6                  ; LowByte-Differenz berechnen
1234       PUSH A                      ; und pushen
1235       MOV A, R5
1236       SUBB A, R7                  ; HighByte-Differenz berechnen, Ergebnis
1237       MOV I2C_W_Byte, A                ; in Akku, und ausgeben
1238       ACALL EE_W
1239       INC DPTR
1240       POP I2C_W_Byte              ; LowByte-Differenz zurück poppen
1241       ACALL EE_W
1242       INC DPTR
1243
1244       MOV R1, DPH                      ; der DPTR steht jetzt an der Zieladresse
1245       MOV R0, DPL                      ; für das Start-Tag-Byte, also dem ersten
1246                                   ; Byte, das nur verschoben wird
1247
1248
1249       MOV DPH, R3
1250       MOV DPL, R2
1251TDL_next_Byte:
1252       ACALL EE_R                  ; Byte, das nur verschoben wird, lesen
1253       INC DPTR
1254       MOV R3, DPH                      ; nächste Leseadresse in R32 speichern
1255       MOV R2, DPL
1256
1257       MOV I2C_W_Byte, I2C_R_Byte  ; ins W_Byte schreiben
1258       MOV DPL, R0                      ; DPTR auf die in R10 enthaltende Schreib-
1259       MOV DPH, R1                      ; Adresse setzten
1260       ACALL EE_W                  ; das Byte an die neue Adresse schreiben
1261       INC DPTR
```

```
1262       MOV R1, DPH                      ; DPTR in R10 sichern
1263       MOV R0, DPL
1264       MOV DPH, R3                      ; R32, also die nächste Leseadresse in DPTR
1265       MOV DPL, R2
1266
1267       ; Abbruchbedingung der inneren Datensatzschleife: R32 = R45
1268       MOV B, R2                   ; Vergleiche das Lowbyte,
1269       MOV A, R4                   ; und wiederhole,
1270       CJNE A, B, TDL_next_Byte    ; wenn nicht gleich
1271       MOV B, R3                   ; Vergleiche das Highbyte,
1272       MOV A, R5                   ; und wiederhole,
1273       CJNE A, B, TDL_next_Byte    ; wenn nicht gleich
1274
1275  ; Hier ist ein Datensatz fertig
1276
1277       ACALL EE_R                  ; Das 1. Byte nach diesem Datensatz lesen
1278       MOV A, I2C_R_Byte                ; ist es nicht FFh, so folgt noch ein Datensatz
1279       CJNE A, #FFh, TDL_next_Datensatz2
1280                                   ; dann das Ganze nochmal von vorne
1281
1282       ; Sonst ist hier endlich Schluss mit dem Verschieben und Ändern
1283       ; nur noch den Rest vom neuen ( = R10) bis zum alten ( = R54) Ende mit FFh löschen
1284       MOV DPH, R1
1285       MOV DPL, R0
1286       MOV I2C_W_Byte, #FFh
1287TDL_FFh_WDH:
1288       ACALL EE_W
1289       INC DPTR
1290       MOV A, R4
1291       CJNE A, DPL, TDL_FFh_WDH    ; Testen, ob schon am Ende angekommen
1292       MOV A, R5
1293       CJNE A, DPH, TDL_FFh_WDH
1294       POP PSW
1295       AJMP Main
1296
1297TDL_next_Datensatz2:
1298       AJMP TDL_next_Datensatz
1299
1300
1301EE_W:        LJMP EEPROM_Byte_Write      ; LJMPs, um von hieraus zu den richtigen
1302EE_R:        LJMP EEPROM_Byte_Read       ; Routinen zu kommen; spart bei häufigem
1303LCD_C:       LJMP LCD_S_Char             ; Ersatz des LCALL durch ACALL auf diese
1304LCD_Z:       LJMP LCD_Send_Zahl          ; LJMPs einige Byte Programmspeicherplatz
1305
1306
1307;-------------------------------------------------------------------------------
1308; Facharbeit-Teil   Statistik_Menü - Rücksprung"routine"
1309;
1310; Ralf Wilke
1311;
1312
1313Main_Menü3_back:
1314       AJMP Main_Menü3             ; Innerhalb des Menüs Datenverwaltung werden
1315                                   ; JB-Befehle benötigt, die nur +-128 Byte
1316                                   ; springen können. Diese Befehle springen hierher,
1317                                   ; und von hier geht's dann weiter
1318
1319
1320
1321;-------------------------------------------------------------------------------
1322; Facharbeit-Teil   Statistik_Menü1 - Datensätze
1323;
1324; Ralf Wilke
1325;
1326
1327Statistik_Menü1:
1328       MOV DPTR, #Statistik_Menü1_DB ; Anfangsadresse auswählen
1329       LCALL LCD_write_Menü        ; Menü ausgeben
1330Statistik_Menü1_Loop:
1331       LCALL Wait_For_Taster       ; Auf Tastendruck warten
1332       JNB Taster0, Statistik_Menü1_weiter
1333                                   ; Wenn nicht Taster 0 gedrückt wurde überspringen
1334       MOV DSNummer, #01h          ; Mit Datensatz 1 anfangen                    ;
1335       ACALL Show_DS_Data          ; Datenausgabe aufrufen
1336       AJMP Main_Menü3
1337Statistik_Menü1_weiter:
1338       JB Taster1, Main_Menü3_back  ; Zur Main-Ebene springen
1339       JB Taster2, Statistik_Menü2  ; Taster 3 "Up" springt zum Statistik._Menü 2
1340;      JB Taster3, Statistik_Menü3  ; Taster 4 "Down" springt zum Statistik._Menü 3
```

```
1341        SJMP Statistik_Menü1_Loop
1342
1343
1344
1345;-------------------------------------------------------------------------------
1346; Facharbeit-Teil   Statistik_Menü2 - Noch frei
1347;
1348; Ralf Wilke
1349;
1350
1351Statistik_Menü2:
1352        MOV DPTR, #Statistik_Menü2_DB ; Anfangsadresse auswählen
1353        LCALL LCD_write_Menü          ; Menü ausgeben
1354Statistik_Menü2_Loop:
1355        LCALL Wait_For_Taster         ; Auf Tastendruck warten
1356        JNB Taster0, Statistik_Menü2_weiter
1357                                      ; Wenn nicht Taster 0 gedrückt wurde den
1358                                      ; LJMP-Befehl überspringen
1359        SJMP Statistik_Noch_Frei
1360Statistik_Menü2_weiter:
1361        JB Taster1, Main_Menü3_back   ; Zur Main-Ebene springen
1362;       JB Taster2, Statistik_Menü3   ; Taster 3 "Up" springt zum Statistik._Menü 3
1363        JB Taster3, Statistik_Menü1   ; Taster 4 "Down" springt zum Statistik._Menü 1
1364        SJMP Statistik_Menü2_Loop
1365
1366
1367
1368
1369
1370Statistik_Noch_Frei:
1371        LCALL Get_Free_MEM            ; Freier Speicher jetzt in R67
1372
1373        MOV R4, #00                        ; R4 (=H) auf 0
1374        CLR C                         ; Carry löschen
1375SMF_WDH:     MOV A, R7                          ; Das Low-Byte in den Akku
1376        MOV R5, A                     ; und auch in R5 (=L)
1377        SUBB A, #100                  ; 100 davon abziehen
1378        MOV R7, A                     ; und diesen Wert in R7 zurücksichern
1379        MOV A, R6                     ; jetzt den etvl. Übertrag auch von
1380        SUBB A, #0                    ; HighByte R6 abziehen
1381        MOV R6, A
1382
1383        JC SMF_Weiter                 ; Ist das Carry jetzt gesetzt, war R67 < 100
1384                                      ; dann ist die Wandlung fertig; wiederspringen
1385        INC R4                        ; sonst R4 um 1 erhöhen
1386        SJMP SMF_WDH                  ; und nochmal testen
1387
1388        ; Der freier Speicher steht jetzt dezimal getrennt in R45
1389
1390SMF_weiter:  MOV LCD_ADDRESS, #00
1391        LCALL LCD_ADD_SET
1392        MOV A, R4
1393        LCALL LCD_Send_Zahl
1394        MOV A, R5
1395        LCALL LCD_Send_Zahl
1396        MOV DPTR, #Statistik_BytesFree_DB
1397        LCALL String2LCD
1398        MOV DPTR, #Meßwert_Erf_RDY_DB
1399        SETB LCD_Zeile
1400        LCALL Line2LCD
1401
1402SMF_Loop:    LCALL Wait_For_Taster
1403        JNB Taster3, SMF_loop
1404        AJMP Main_Menü3
1405
1406
1407;-------------------------------------------------------------------------------
1408; Facharbeit-Teil   Zeige die Daten der Datensätze an
1409;
1410; Ralf Wilke
1411;
1412
1413
1414Show_DS_Data:                                 ; Liest die Beschreibungen des Datensatzes
1415                                      ; aus und stelt sie auf dem LCD dar
1416        CLR SDSD_back
1417        PUSH DSnummer
1418        ACALL find_end
1419        POP DSNummer
```

```
1420        ACALL find_Nummer                ; finde die Adresse das Datensatzes
1421
1422
1423        JB DS_gefunden, SDSD_EEPROM_nicht_leer
1424
1425
1426        MOV DPTR, #EEPROM_Leer_DB
1427                                         ; sonst Meldung ausgeben, dass Speicher leer
1428        CLR LCD_Zeile                    ; 1. Zeile
1429        LCALL Line2LCD
1430        SETB LCD_Zeile                   ; 2. Zeile
1431        MOV DPTR, #Canceled_OK_DB        ; Abbruchmeldung ausgeben
1432        LCALL Line2LCD
1433SDSD_Leer_Loop:
1434        LCALL Wait_for_Taster            ; warten, bis OK gedrückt wird
1435        JNB Taster3, SDSD_Leer_loop
1436        SETB SDSD_back
1437        SJMP SDSD_End                    ; Und Rücksprung
1438
1439
1440
1441SDSD_EEPROM_nicht_leer:                       ; Wenn das EEPROM nicht leer ist
1442        MOV LCD_Address, #00h            ; Setze den LCD-Cursor auf den Anfang
1443        LCALL LCD_Add_Set
1444        MOV A, #"N"                           ; Gebe "Nr" aus
1445        ACALL LCD_C
1446        MOV A, #"r"
1447        ACALL LCD_C
1448        MOV A, DSNummer                  ; Gebe Nummer des Datensatzes aus
1449        ACALL LCD_Z
1450        LCALL LCD_blank
1451        INC DPTR                         ; DPTR um 3 erhöhen, um am Anfang der
1452        INC DPTR                         ; Beschreibungsdaten zu beginnen
1453        INC DPTR
1454
1455        MOV R0, #2                       ; kleine Schleife, um Platz zu sparen
1456SDSD_WDH1:   ACALL EE_R                       ; Datum ausgeben
1457        MOV A, I2C_R_Byte
1458        ACALL LCD_Z
1459        MOV A, #"."                           ; Trennungspunkt ausgeben
1460        ACALL LCD_C
1461        INC DPTR
1462        DJNZ R0, SDSD_WDH1
1463
1464        ACALL EE_R                       ; Uhrzeit ausgeben
1465        MOV A, I2C_R_Byte
1466        ACALL LCD_Z
1467        MOV A, #":"                           ; Trennungsdoppelpunkt ausgeben
1468        ACALL LCD_C
1469        INC DPTR
1470        ACALL EE_R
1471        MOV A, I2C_R_Byte
1472        ACALL LCD_Z
1473
1474SDSD_Loop:   LCALL Wait_for_Taster            ; Auf Tasendruck warten
1475        JNB Taster1, SDSD_not_back       ; Wenn Taster 1 = Back nicht gedrückt wurde
1476        SETB SDSD_back
1477        RET
1478SDSD_not_back:
1479        JNB Taster2, SDSD_not_up         ; wenn Taster 2 = Up nicht gedrückt wurde,
1480        MOV A, DSNummer                  ; überspringen, sonst aktuelle Datensatz-
1481                                         ; nummer in den Akku
1482
1483        CJNE A, #FFh, SDSD_up            ; ist diese FFh, so gibt es keine weiteren
1484                                         ; Datensätze mehr, da dieses die max.
1485                                         ; Datensatznummer ist
1486        SJMP SDSD_Loop                   ; Dann zurückspringen, sonst...
1487
1488SDSD_up:           MOV A, Anzahl_DS            ; ...hier weiter. Testen, ob das Ende
1489        CJNE A, DSNummer, SDSD_up2       ; erreicht ist. Wenn nein, überspringen
1490        SJMP SDSD_Loop                   ; sonst zurück in den Loop
1491SDSD_up2:          INC DSNummer                ; RAM DSNummer erhöhen
1492        SJMP Show_DS_Data                     ; und Daten des neuen Datensatzes ausgeben
1493
1494
1495SDSD_not_up:JNB Taster3, SDSD_not_down    ; Testen, ob Taster 3 = Down gedrückt wurde
1496
1497        MOV A, DSNummer                  ; DSNummer in den Akku
1498        DEC A                            ; Diesen dekrementieren
```

```
1499       JZ SDSD_Loop                ; ist der Akku 0, war DSNummer vorher auf 1
1500                                   ; Nummer 0 ist nicht erlaubt
1501
1502SDSD_Down:   DEC DsNummer                ; Sonst DSNummer verkleinern
1503       AJMP Show_DS_Data               ; und die neuen Daten ausgeben
1504
1505SDSD_not_Down:                         ; Wenn nicht Back, Up oder Down gerückt wurde
1506       JB Taster0, SDSD_OK         ; testen, ob's wenigstens Taster 0 = OK war
1507       SJMP SDSD_Loop              ; wenn nicht, zurück
1508SDSD_OK:
1509SDSD_End:    RET
1510
1511
1512;-------------------------------------------------------------------------------
1513; Facharbeit-Teil   Alle Daten löschen
1514;
1515; Ralf Wilke
1516;
1517
1518Del_ALL_Data:
1519       CLR IE.4                    ; Ser-Schnittstelle Interrupt abschalten
1520                                   ; wir wollen uns nicht stören lassen :)
1521
1522       PUSH A                      ; Akku retten
1523       CLR LCD_RS                  ; Das ganze Display löschen
1524       MOV A, #00000001b               ; Command: Clear display
1525       LCALL LCD_WR
1526
1527       MOV DPTR, #Del_ALL_working_DB ; DPTR auf die Anfangsadresse setzten
1528       CLR LCD_Zeile               ; 1. Zeile auswählen
1529       LCALL Line2LCD              ; Die erste Zeile ausgeben
1530
1531       MOV R0, #0                  ; R0, den "Strich"zähler, auf 0 setzten
1532       MOV DPTR, #0000h            ; Am Anfang anfangen zu löschen
1533       MOV I2C_W_Byte, #FFh        ; Überall FFh hinschreiben
1534
1535Del_All_Data_next_Byte:
1536       LCALL EEPROM_Byte_write         ; Das Byte schreiben
1537       MOV A, DPL                  ; DPL in den Akku
1538       ANL A, #00111111b               ; Nur die Bits 0-5 behalten
1539       CJNE A, #00111111b, DAD_no_Graphik
1540                                   ; Wenn gerade Adresse 63 im Akku verbleibt,
1541                                   ; sind 64 Bytes geschrieben. Es wird nun dem
1542                                   ; Fortschrittsbalken ein Strich hinzugefügt
1543       MOV LCD_ADDRESS, #40h       ; Der LCD-Adressenzeiger wird auf den Anfang
1544       LCALL LCD_ADD_SET               ; der 2. Zeile gesetzt,
1545       INC R0                      ; R0 erhöht und anschliessend durch 5 geteilt,
1546       MOV B, #05                  ; denn jedes Zeichen des LCD kann 5 Striche
1547       MOV A, R0                   ; darstellen. Die Anzahl der ganzen Zeichen
1548       DIV AB                      ; (Kästchen), ist dann im Akku, die Anzahl
1549                                   ; der zusätzlichen Striche im B-Akku
1550       JZ DAD_fifth                ; Wenn gar keine vollen Kästchen benötigt
1551                                   ; werden, direkt zu den Fünfteln (also
1552                                   ; den Strichen) springen.
1553DAD_WDH:     PUSH A                        ; sonst den Akku sichern
1554       MOV A, #FFh                     ; und den ASCII-Code für ein volles Kästchen
1555       LCALL LCD_S_Char            ; auf's LCD ansgeben
1556       POP A                       ; Akku wiederherstellen
1557       DJNZ A, DAD_WDH             ; und das Ganze so oft, wie ganze Fünfer in R0
1558                                   ; waren, also bis der Akku 0 ist wiederholen
1559DAD_fifth:   MOV A, B                    ; Nun die Fünftel bearbeiten - B nach A
1560       JZ DAD_no_fifth             ; Wenn keine Fünftel da sind, direkt weiter
1561       LCALL LCD_S_Char            ; Sonst den ASCII-Code im Akku auf's LCD
1562                                   ; ausgeben. Die ASCII-Codes 01h bis 04h wurden
1563                                   ; im Init-Teil bereits als Strich-"Zeichen"
1564                                   ; entsprechend ihrem ASCII-Code definiert
1565DAD_no_fifth:
1566DAD_no_Graphik:
1567       INC DPTR                    ; nächstes Byte
1568       MOV A, DPH                  ; DPH in Akku zum Vergleich
1569       CJNE A, #00010000b, Del_All_Data_next_Byte
1570                                   ; Bis alle Bytes einmal beschrieben wurden,
1571                                   ; wiederholen
1572
1573
1574; Hier folgt ein    ++++++++++ BESCHISS ****+++++++++    des Benutzers
1575
1576; Erklärung:
1577; Der gesammte Speicherplatz beträgt 4Kbyte = 4096 Byte. Alle 64 Byte wird ein Strich
```

```
1578; hinzugefügt, bei 4096 Byte sind das dann auch 64 Striche.
1579; 64 Striche auf 7x5-DotMatrix-Zeichen verteilt sind 12 ganze Zeichen und 4 Striche
1580; Um Ende des Löschaktion ist also das letzte Kästchen nur zu 4/5 gefüllt.
1581; Rechnerisch und überhaupt ist das völlig richtig, aber der Benutzer wird sich
1582; wundern, das der letzte Strich zum vollen 13 Kästchen nicht erscheint und evtl.
1583; einen Programmfehler vermuten. Um dies zu umgehen und eine Anzeige zu schaffen,
1584; die in also ganzen vollen Zeichen mit 5 Strichen arbeitet, wird nachdem der
1585; ein eigentliche Löschvorgang bereits vorbeit ist eine gewisse Zeit, die etwa der
1586; Löschzeit für 64 Bytes entspricht, gewartet, und daraufhin das letzte Zeichen
1587; mit einem vollen Kästchen geschrieben.
1588
1589       MOV wait_time_025s, #4       ; 4 x 0,25s = 2 Sekunden warten
1590       LCALL wait_025s
1591       MOV LCD_ADDRESS, #4Ch        ; LCD-Adressenzeiger auf das 4/5-Kästchen setzten
1592       LCALL LCD_ADD_SET
1593       MOV A, #ffh                       ; Dort ein volles Kästchen hinschreiben
1594       LCALL LCD_S_Char
1595
1596; Hier ENDET ein    ++++++++** BESCHISS ****++++++++    des Benutzers
1597
1598       MOV LCD_ADDRESS, #4Eh        ; vorletztes Zeichen adressieren
1599       LCALL LCD_ADD_SET
1600       MOV A, #"O"                       ; Dort als Tasterbeschriftung OK hinschreiben
1601       LCALL LCD_S_Char
1602       MOV A, #"K"
1603       LCALL LCD_S_Char
1604
1605       POP A                        ; akku wiederherstellen
1606DAD_Wait:          LCALL Wait_For_Taster
1607       JNB Taster3, DAD_Wait        ; warten, bis OK gedrückt wurde
1608
1609       SETB IE.4                    ; Ser.-Schnittstelle-Interrupt wieder einschalten
1610       LJMP Main                    ; Zum Main_Menü zurückspringen
1611
1612
1613;-------------------------------------------------------------------------------
1614; Facharbeit-Teil   Neuen Datensatz erstellen
1615;
1616; Ralf Wilke
1617
1618; Diese Subroutine sucht beginnend bei EEPROM-Adresse 0000h die erste freie Adresse
1619; nach dem letzten Datensatz, schreib an diese Stelle die fortlaufende Nummerierung
1620; des neuen Datensatzes. Hierauf folgend werden die restlichen 9 Verwaltungsbytes
1621; geschrieben: 1. freie Adresse nach diesem neuen Datensatz, Start-Datum und Start-Zeit
1622; sowie die Intervall-Länge zwischen den Messungen.
1623; Im RAM DPL/H_Save wird die Adresse des 1. freien Bytes nach diesem Datensatz,
1624; die im RAM DPL/H_next_Add auch gespeichert wird,
1625; gespeichert.
1626;
1627
1628;
1629create_new_dataset:
1630       ACALL Find_End
1631       INC Anzahl_DS
1632       MOV I2C_W_Byte, Anzahl_DS    ; Der Stand des Zählers wird erhöht
1633       LCALL EEPROM_Byte_write           ; und an die Stelle des FFh geschreiben.
1634       INC DPTR                     ; Der DPTR wird auf die nächste Adresse erhöht
1635
1636       ; die 1. Adresse nach diesem Datensatz schreiben.
1637
1638       MOV DPH_Save, DPH                 ; DPTR für gleich, aber auch für das spätere
1639       MOV DPL_Save, DPL                 ; Messwerthinzufügen sichern
1640
1641       MOV R2, #9                   ; DPTR um 9 erhöhen, denn es folgen noch 9 Byte
1642CND_WDH:           INC DPTR                          ; Verwaltung. bis die 1. Adresse
   erreicht ist
1643       DJNZ R2, CND_WDH
1644       MOV I2C_W_Byte, DPH          ; Das High-Byte in den Write-RAM
1645       MOV R0, DPL                       ; Das Lowbyte in R0 sichern
1646
1647       MOV DPH_next_Add, DPH        ; Adresse nach dem letzten Temp-Eintrag sichern
1648       MOV DPL_next_Add, DPL
1649
1650       MOV DPH, DPH_Save                 ; Den DPTR wiederherstellen
1651       MOV DPL, DPL_Save
1652       LCALL EEPROM_Byte_write           ; High-Byte an die richtige Stelle schreiben
1653       MOV I2C_W_Byte, R0           ; Das Low-Byte vom Stack direkt in den Write-Ram
1654       INC DPTR                     ; den DPTR für die nächste Adresse erhöhen
1655       LCALL EEPROM_Byte_write           ; und schreiben
```

```
1656       INC DPTR                    ; den DPTR für die nächste Adresse erhöhen
1657       MOV R0, #Start_Day          ; Adresse von Start_Day in R0 schreiben
1658CND_WDH2:          MOV I2C_W_Byte, @R0          ; RAM-Inhalt in den Write_RAM schreiben
1659       LCALL EEPROM_Byte_write           ; ins EEPROM schreiben
1660       INC DPTR                    ; DPTR und Zeiger R0 erhöhen
1661       INC R0
1662       CJNE R0, #Intervall_S + 1, CND_WDH2
1663                                   ; Wenn nicht die RAM-Adresse des letzten Bytes
1664                                   ; erreicht ist, wiederholen
1665       RET
1666
1667;-------------------------------------------------------------------------------
1668; Facharbeit-Teil   Findet den letzten Datensatz
1669;
1670; Ralf Wilke
1671;
1672; DPTR steht auf der letzten Adresse + 1,
1673; Anzahl_DS enthält die Nummer des letzten Datensatzes
1674
1675Find_end:          MOV DSNummer , #FFh          ; Ausschliessen, dass ein Datensatz vor dem
1676                                   ; der Nummer FFh ( = absolut der letzte)
1677                                   ; gefunden wird
1678       ACALL Find_Nummer
1679       MOV Anzahl_DS, DS_Zähler
1680       RET
1681
1682
1683;-------------------------------------------------------------------------------
1684; Facharbeit-Teil   Findet den Datensatz mit der Nummer "DSNummer"
1685;
1686; Ralf Wilke
1687;
1688; DPTR steht auf der ersten Adresse,
1689; DSNummer übergibt die Nummer des Datensatzes
1690; ist DS_gefunden nachher gesetzt, so wurde der Datensatz gefunden
1691
1692
1693Find_Nummer: MOV DPTR, #0000h            ; DPTR auf den Anfang setzen
1694       MOV DS_Zähler, #00h         ; DS_Zähler als Zähler der vorhandenen
1695                                   ; Datensätze auf 0
1696       CLR DS_gefunden             ; Bit löschen
1697
1698Fe_check_if_last:                       ; überprüfen, ob noch Datensätze folgen
1699       LCALL EEPROM_Byte_read      ; Adresse auslesen
1700       XCH A, I2C_R_Byte                ; Akku und das gelesene Byte tauschen
1701       CJNE A, #FFh, Fe_check_if_Nummer
1702                                   ; Wenn das 1. Byte NICHT FFh ist, folgt ein
1703                                   ; Datensatzt. Dann überprüfen, ob die Nummer
1704                                   ; gewünscht ist
1705
1706                                   ; Wenn das 1. Byte FFh ist, folgen KEINE
1707                                   ; weiteren Datensätze mehr
1708       XCH A, I2C_R_Byte                ; Akku und das gelesene Byte zurücktauschen
1709       SJMP Found_end              ; Dann weiter
1710
1711Fe_check_if_Nummer:
1712       INC DS_Zähler               ; Den Datensatz-Zähler erhöhen
1713                                   ; Auch wenn es nicht die richtige Nummer ist, es
1714                                   ; wurde ja ein Datensatz gefunden, und somit muss
1715                                   ; auch der Zähler erhöht werden
1716                                   ; (ehem. kleiner Fehler)
1717
1718
1719       CJNE A, DSNummer, Fe_find_next; Wenn die im Byte, das gerade gelsen wurde,
1720                                   ; enthaltene Nummer nicht mit der gewünschten
1721                                   ; Nummer übereinstimmt, weitersuchen
1722Fe_found_Nummer:
1723       SETB DS_gefunden            ; Sonst Bit setzen
1724       XCH A, I2C_R_Byte                ; Den Akku wiederhestellen
1725       SJMP Found_end              ; und zum Ende springen
1726
1727
1728Fe_find_next:
1729       XCH A, I2C_R_Byte                ; Akku und das gelesene Byte zurücktauschen
1730                                   ; Wenn das 1. Byte NICHT FFh war, folgt noch
1731       INC DPTR                    ; ein Datensatz. Dann den DPTR erhöhen und
1732       LCALL EEPROM_Byte_read      ; mit den nächsten 2 Bytes die Adresse nach
1733       PUSH I2C_R_Byte             ; diesem Datensatz erhalten. Der DPTR wird
1734       INC DPTR                    ; nun auf diese neue Adresse gesetzt
```

```
1735        LCALL EEPROM_Byte_read
1736        MOV DPL, I2C_R_Byte
1737        POP DPH
1738        SJMP Fe_check_if_last       ; und die Überprüfung beginnt wieder von vorn
1739Found_end:   RET
1740
1741
1742;-------------------------------------------------------------------------------
1743; Facharbeit-Teil   Freier Platz im Speicher
1744;
1745; Ralf Wilke
1746;
1747; Rückgabe in R67
1748
1749Get_free_MEM:
1750        ACALL Find_End
1751        ; Im DPTR ist jetzt die letzte benutzte Adresse enthalten
1752        ; DPTR von 4kByte = 4096 Byte = 1000h Byte anziehen
1753        CLR A
1754        CLR C
1755        SUBB A, DPL
1756        MOV R7, A
1757        MOV A, #10h
1758        SUBB A, DPH
1759        MOV R6, A
1760        RET
1761
1762
1763
1764
1765;-------------------------------------------------------------------------------
1766; Facharbeit-Teil   Byte dezimal auf dem LCD ausgeben
1767;
1768; Ralf Wilke
1769;
1770
1771LCD_Send_Zahl:
1772        PUSH PSW
1773        PUSH B
1774        MOV B, #10                  ; Hilf-Akku B mit 10 beschreiben
1775        DIV AB                      ; Akku durch 10 teilen, Zehner sind dann im
1776                                    ; Akku, Einer im Hilfsakku B
1777        ADD A, #30h                      ; 30h zum Akku addieren, um die ASCII-Codes zu
1778                                    ; erhalten
1779        LCALL LCD_S_Char            ; Zeichen ausgeben, die Adresse wird im LCD
1780                                    ; automatisch erhöht
1781        MOV A, B                    ; Hilfsakku in den Akku schreiben
1782        POP B                       ; Hillfsakku wiederherstellen
1783        ADD A, #30h                      ; 30h zum Akku addrieren, s.o.
1784        LCALL LCD_S_Char
1785        POP PSW
1786        RET
1787
1788
1789;-------------------------------------------------------------------------------
1790; Facharbeit-Teil   Fertig-Meldung
1791;
1792; Ralf Wilke
1793;
1794
1795
1796;Ready_MSG:  MOV DPTR, #Ready_MSG_DB           ; Anfangsadresse auswählen
1797;       LCALL 2Lines2LCD            ; auf LCD ausgeben
1798;Ready_MSG_Loop:
1799;       LCALL Wait_For_Taster       ; Auf Tasterdruck warten
1800;       RET
1801
1802
1803
1804
1805;INCLUDE lcd.asm
1806;-------------------------------------------------------------------------------
1807; Facharbeit-Teil   Menü mit der jeweiligen durch den DPTR festgelegten Beschriftung
1808; und der OK-ESC-Up-Down-Zeile ausgeben
1809;
1810; Ralf Wilke
1811;
1812
1813LCD_write_Menü:
```

```
1814        CLR LCD_Zeile               ; 1. Zeile wählen
1815        LCALL Line2LCD              ; ausgeben
1816        MOV DPTR, #OkEscUpDown_DB   ; Die OK-ESC-Up-Down-Zeile auswählen
1817        SETB LCD_Zeile              ; 2. Zeile wählen
1818        LCALL Line2LCD              ; ausgeben
1819        RET
1820
1821;-------------------------------------------------------------------------------
1822; Facharbeit-Teil   2 Zeilen auf das LCD ausgeben, Übergabe der internen Speicher-
1823; Adresse über den Datenpointer
1824;
1825; Ralf Wilke
1826;
1827
1828
18292Lines2LCD:  ;PUSH A                    ; Akku retten
1830;       MOV LCD_ADDRESS, #00h
1831;       LCALL LCD_ADD_SET
1832;       CLR 2nd_row                     ; erst mal die erste Zeile
1833        CLR LCD_Zeile
1834        ACALL Line2LCD
1835        SETB LCD_Zeile
1836        ACALL Line2LCD
1837        RET
1838
1839;2Lines_Next_Line:
1840;       MOV R7, #16                     ; 16 Bytes pro Zeile holen
1841;2Lines_Next_Byte:
1842;       CLR A                       ; Akku löschen
1843;       MOVC A, @A+DPTR             ; Byte aus dem Speicher holen
1844;       LCALL LCD_S_Char            ; ausgeben
1845;       INC DPTR                    ; Datenpointer erhöhen
1846;       DJNZ R7, 2Lines_Next_Byte
1847;       JB 2nd_row, 2Lines_ready    ; wenn wir schon mal hier waren...
1848;       MOV LCD_ADDRESS, #40h       ; Anfang 2. Zeile
1849;       LCALL LCD_ADD_SET
1850;       SETB 2nd_row                ; Bit setzten, damit dieser Teil später
1851                                   ; übersprungen wird
1852;       SJMP 2Lines_Next_Line       ; 2. Zeile ausgeben
1853
1854;2Lines_ready:
1855;       POP A
1856;       RET
1857
1858
1859;-------------------------------------------------------------------------------
1860; Facharbeit-Teil   Eine Zeile auf das LCD ausgeben, Übergabe der internen Speicher-
1861; Adresse über den Datenpointer, Auswahl der Zeile über LCD_Zeile 0 = 1. 1 = 2.
1862;
1863; Ralf Wilke
1864;
1865
1866Line2LCD:    JB LCD_Zeile, Line2LCD_2Zeile
1867        MOV LCD_ADDRESS, #00h       ; Anfangsadresse 1. Zeile
1868        SJMP Line2LCD_ausgeben
1869Line2LCD_2Zeile:
1870        MOV LCD_ADDRESS, #40h       ; Anfangsadresse 2. Zeile
1871Line2LCD_ausgeben:
1872        LCALL LCD_ADD_SET               ; Adresse ans LCD ausgeben
1873String2LCD:  PUSH A                    ; Akku retten
1874        MOV R7, #16
1875Line_Next_Byte:
1876        CLR A                       ; Akku löschen
1877        MOVC A, @A+DPTR             ; Byte aus dem Speicher holen
1878        CJNE A, #A6h, L2L_out
1879        SJMP L2L_0
1880L2L_out:     LCALL LCD_S_Char          ; ausgeben
1881        INC DPTR                    ; Datenpointer erhöhen
1882        DJNZ R7, Line_Next_Byte
1883        SJMP L2L_end
1884L2L_0:  INC DPTR
1885        MOV A, #" "
1886L2L_0_next:  LCALL LCD_S_Char
1887        DJNZ R7, L2L_0_next
1888L2L_end:     POP A
1889        RET
1890
1891INCLUDE ds1620.asm
1892
```

```
1893;-------------------------------------------------------------------------------
1894; Facharbeit-Teil   Daten in das DD- oder CG-RAM schreiben    Übergabe: Akku
1895;
1896; Ralf Wilke
1897;
1898LCD_S_CHAR:
1899        SETB LCD_RS                       ; Daten ins LCD-RAM schreiben
1900;       LCALL LCD_WR
1901;             RET
1902        AJMP LCD_WR                       ; kein LCALL nötig, da direkt nach dem LCALL
1903                                    ; ein RET folgt, und so auch das RET der auf-
1904                                    ; gerufenen Routine richtig zurückspringt
1905                                    ; spart 2 Byte Stack
1906
1907;-------------------------------------------------------------------------------
1908; Facharbeit-Teil   Daten aus dem DD- oder CG-RAM lesen   Übergabe: Akku
1909;
1910; Ralf Wilke
1911;
1912LCD_R_CHAR:
1913        SETB LCD_RS                       ; Daten ins LCD-RAM schreiben
1914;       LCALL LCD_WR
1915;             RET
1916        AJMP LCD_RD                       ; kein LCALL nötig, da direkt nach dem LCALL
1917                                    ; ein RET folgt, und so auch das RET der auf-
1918                                    ; gerufenen Routine richtig zurückspringt
1919                                    ; spart 2 Byte Stack
1920
1921
1922;-------------------------------------------------------------------------------
1923; Facharbeit-Teil   LCD-Daten_Adressen setzen
1924;
1925; Ralf Wilke
1926
1927
1928LCD_ADD_SET: PUSH A                             ; Akku retten
1929        CLR LCD_RS                  ; auf Kommando setzten
1930        MOV A, LCD_ADDRESS
1931        SETB ACC.7
1932        ACALL LCD_WR
1933        POP A
1934        RET
1935
1936;-------------------------------------------------------------------------------
1937; Facharbeit-Teil   LCD-Zeichen_Adressen setzen
1938;
1939; Ralf Wilke
1940
1941
1942LCD_ADD_CG_SET:PUSH A                   ; Akku retten
1943        CLR LCD_RS                  ; auf Kommando setzten
1944        MOV A, LCD_ADDRESS
1945        ACALL LCD_WR
1946        POP A
1947        RET
1948
1949
1950;-------------------------------------------------------------------------------
1951; Facharbeit-Teil   Kommunikation mit dem LCD
1952;
1953; Ralf Wilke
1954
1955LCD_WR:            ACALL LCD_READY                  ; warten, bis LCD bereit ist
1956        CLR LCD_RW                  ; schreiben
1957        LCALL WR_Status_Ausgang           ; Status-Leitungen setzen
1958        NOP
1959        SETB ENBL_SEL0              ; Enable-Eingang auf High
1960        NOP
1961        MOV P1, A                   ; Akku auf Datenport legen
1962        NOP
1963        CLR ENBL_SEL0               ; Enable-Eingang wieder auf Low
1964        NOP
1965        RET                         ; Rücksprung
1966
1967
1968LCD_RD:            ACALL LCD_READY             ; warten, bis LCD bereit ist
1969LCD_RD2:           SETB LCD_RW                       ; lesen
1970        LCALL WR_Status_Ausgang           ; Status-Leitungen setzen
1971        MOV P1,#FFh                       ; P1 als Eingang setzten
```

```
1972        NOP
1973        SETB ENBL_SEL0              ; LCD enablen
1974        NOP
1975        MOV A, P1                   ; LCD Ausgaben in Akku
1976        CLR ENBL_SEL0               ; Rücksprung
1977        NOP
1978        RET
1979
1980LCD_READY:   PUSH OUTPUT_STATUS          ; wartet, bis das LCD bereit ist
1981        PUSH A                      ; gewünschte LCD-Betriebsart sichern
1982        CLR LCD_RS                  ; Auf read address counter and busy flag
1983        LCALL WR_Status_Ausgang           ; (ACC.7) stellen
1984LCD_READY2:  lCALL LCD_RD2
1985        JB ACC.7, LCD_READY2        ; Wenn noch beschäftig, wieder abfragen
1986        POP A
1987        POP OUTPUT_STATUS
1988        RET
1989
1990
1991
1992INCLUDE I2C.asm
1993
1994;-------------------------------------------------------------------------------
1995; Facharbeit-Teil   Auf Tastendruck warten und Taster entprellen
1996;
1997; Ralf Wilke
1998
1999Wait_For_Taster:
2000        PUSH A                      ; Akku sichern
2001        MOV R6, #255                ; R6 = 255
2002        CLR A                       ; Akku löschen
2003        MOV R5, #40                       ; R5 = 40
2004        MOV R4, #255                ; R4 = 250
2005Get_Taster:  LCALL RD_Taster              ; Taster einlesen
2006        JNB mach_Messung, keine_Messung
2007                                    ; Wenn keine Messung durch die Interruptroutine
2008                                    ; angefordert wurde, überspringen
2009        CLR mach_Messung            ; Wenn doch, das Bit löschen
2010        LCALL Temp_Aufnahme         ; und die Routine aufrufen
2011
2012keine_Messung:
2013        CJNE A, Input_Taster, Taster_changed
2014                                    ; Testen, ob ein Taster gedrückt wurde
2015        DJNZ R4, Get_Taster         ; R4 * R5 = 10000 mal Taster abfragen,
2016        MOV R4, #255                ; jede Abfragung dauert ca. 50µs, das
2017        DJNZ R5, Get_Taster         ; Blinken geschieht also alle 0,5 s
2018        MOV R5, #40
2019        LCALL LCD_Blink             ; Blinken
2020        JNB Show_Temp, Get_Taster
2021        PUSH A
2022        ACALL Take_Temp
2023        POP A
2024        MOV R6, #255
2025        SJMP Get_Taster             ; nochmal
2026
2027Taster_changed:
2028        LCALL wait_100us            ; 100µs warten
2029        DJNZ R6, Get_Taster         ; Das alles 255 Mal wiederholen
2030
2031        MOV A, Input_Taster         ; Gedrückten Status merken
2032Get_Taster2: LCALL RD_Taster              ; Taster erneut einlesen
2033        CJNE A, Input_Taster, Taster_changed2
2034                                    ; Wenn der Taster wieder losgelassen wurde, also
2035                                    ; das RAM Input_Taster vom Inhalt es Akku sich
2036                                    ; unterscheidet, wurde der Taster wieder losgelassen
2037                                    ; dann weiterspringen
2038        SJMP Get_Taster2            ; sonst nochmal
2039Taster_changed2:
2040        LCALL wait_100us            ; 100µs warten
2041        DJNZ R6, Get_Taster2        ; Das alles noch 255 Mal
2042        MOV Input_Taster, A         ; gedrückten Zustand ins RAM zurückschreiben
2043        POP A                       ; Akku wiederherstellen
2044        RET                         ; Rücksprung
2045
2046;-------------------------------------------------------------------------------
2047; Facharbeit-Teil   Taster einlesen
2048;
2049; Ralf Wilke
2050
```

```
2051
2052RD_Taster:         ; liest den Zustand der Taster 0 bis 3, sowie der Inputs 0 bis 3
2053             ; über IC6 ein.
2054             ; Ausgabe in RAM INPUT_TASTER
2055
2056
2057        SETB Ser_Int_Lock                ; Interrupts verhindern
2058        MOV P1, #11111111b          ; Datenbus auf 1 setzten, um Daten zu lesen
2059        ORL P3, #00110000b          ; Bits 4 und 5 setzten
2060                                    ; dadurch ENBL_INPUT und damit die Eingänge
2061                                    ; von IC6 auf den Datenbus legen
2062
2063        NOP
2064        NOP
2065        PUSH A
2066        MOV A, P1                   ; Die Port-Zustände an P1 stellen nun
2067                                    ; die Tasterzustände dar
2068        CPL A                       ; komplementieren, da 0 = Taster gedrückt
2069        MOV INPUT_TASTER, A         ; im RAM INPUT_TASTER speichern
2070        POP A
2071        ANL P3,#11001111b                ; ENBL_INPUT und damit IC6 deaktiveren.
2072        NOP
2073        NOP
2074        CLR Ser_Int_Lock
2075        JNB Ser_Int_arrived, RD_Taster_end
2076        SETB SCON.0
2077RD_Taster_end:
2078        RET                         ; Rücksprung
2079
2080
2081
2082
2083
2084;-------------------------------------------------------------------------------
2085; Facharbeit-Teil  Status-Ausgänge setzen
2086;
2087; Ralf Wilke
2088;
2089
2090WR_Status_Ausgang: ; Gibt über den Datenbus Status-Signale aus
2091             ; Eingabe der Werte über RAM OUTPUT_STATUS
2092
2093             ; Neuerung: Auswahl der richtigen SCL Leitung über Bit 11
2094             ; des DPTR, 0 = > SCL1      1 =  >  SCL2
2095
2096        JNB DPTR_is_I2C, WR_Status_Ausgang_no_I2C
2097                                    ; Wenn zur Zeit nicht über den DPTR ein
2098                                    ; I2C-EEPROM gewählt werden soll, überspringen
2099        PUSH A                      ; Akku sichern
2100
2101        MOV A, DPH                  ; DPH in Akku, Bit 3 DPH = Bit 11 DPTR
2102        JB ACC.3, EEPROM2                ; nach Bit 3 entscheiden, welches EEPROM
2103EEPROM1:           CLR SCL2                        ; evtl. nicht nötig, sperrt EEPROM2
2104        JB SCL_NS, EEPROM1_H        ; Wenn High gewünscht ist,...
2105        CLR SCL1                    ; EEPROM1-SCL auf Low
2106        SJMP WR_Status_Ausgang2
2107
2108EEPROM1_H:   SETB SCL1                   ; EEPROM1-SCL auf High
2109        SJMP WR_Status_Ausgang2
2110
2111EEPROM2:           CLR SCL1                    ; evtl. nicht nötig, sperrt EEPROM1
2112        JB SCL_NS, EEPROM2_H        ; Wenn High gewünscht ist,...
2113        CLR SCL2                    ; EEPROM2-SCL auf Low
2114        SJMP WR_Status_Ausgang2
2115
2116EEPROM2_H:   SETB SCL2                   ; EEPROM2-SCL auf High
2117
2118WR_Status_Ausgang2:
2119        POP A                       ; Den Akku erst mal wieder zurückholen
2120WR_Status_Ausgang_no_I2C:
2121        MOV P1, OUTPUT_STATUS       ; RAM STATUS auf Datenbus legen
2122        SETB ENBL_SEL1              ; BIT 4 setzten, dadurch ENBL_OUTPUT
2123                                    ; und damit den Datenbus
2124                                    ; auf die Ausgänge von IC5 legen
2125        LCALL wait_100us
2126        CLR ENBL_SEL1               ; BIT 4 setzen. Damit ist IC5 wieder inaktiv.
2127        LCALL wait_100us
2128        RET                         ; Rücksprung
2129
```

```
2130
2131;-------------------------------------------------------------------------------
2132; Facharbeit-Teil  Temperatur einlesen und speichern; LCD-Bearbeitung
2133;
2134; Ralf Wilke
2135;
2136
2137Temp_Aufnahme:
2138       PUSH A
2139       MOV A, Anzahl_L             ; Testen, ob schon Anzahl = 0 ist,
2140       DEC A                       ; also alle Messungen gemacht wurden
2141       MOV Anzahl_L, A
2142       CJNE A, #ffh, take_Temp          ; Wenn das Low-Byte noch nicht ffh erreicht hat
2143       MOV A, Anzahl_H             ; und das High-Byte betrachen. Wenn hier nicht
2144       CJNE A, #00, borrow_High    ; 00h ist, dieses Byte später dekrementieren
2145                                   ; sonst sind alle Messungen gemacht
2146       ;hier ist dann < Anzahl = 0000 >
2147
2148       CLR TCON.4                  ; Timer stoppen
2149       MOV Anzahl_L, #00                ; Lowbyte von ffh auf 00h zurückstellen
2150
2151
2152       MOV DPTR, #Meßwert_Erf_RDY_DB ; DPTR beschreiben
2153       SETB LCD_Zeile              ; untere Zeile auswählen
2154       LCALL Line2LCD              ; ausgeben
2155       SJMP Temp_Aufn_Ende
2156
2157
2158borrow_High: DEC A                      ; Akku mit dem High-byte dekrementieren
2159       MOV Anzahl_H, A             ; und ins RAM zurückschreiben
2160       MOV Anzahl_L, #99                ; Low-Byte auf Ausgangswert 99d stellen
2161
2162
2163Take_Temp:   ; Temperaturmessung machen und speichern
2164
2165;      MOV A, #EEh                      ; EEh = Start Conversion
2166;      LCALL proto_DS1620
2167;      CLR ds_rst
2168;      LCALL wr_status_ausgang
2169;      MOV wait_time_025s, #3
2170;      LCALL wait_025s
2171       MOV A, #AAh                      ; AAh = Read Temperature
2172       LCALL Proto_DS1620
2173       LCALL RD_DS1620
2174       MOV Temp_L, A
2175       CLR A
2176       ADDC A, #00
2177       MOV Temp_H, A
2178       ; Die Temperatur ist jetzt im RAM Temp_L und Temp_H
2179       JNB Show_Temp, Save_Temp    ; Wenn die Temperatur nicht im Main-Menü
2180                                   ; angezeigt werden soll, weiterspringen
2181       ACALL Temp2LCD
2182       RET
2183
2184
2185       ;LC-Anzeige aktualisieren
2186Save_Temp:
2187       ACALL ErfData2LCD
2188
2189
2190       ; Die Adresse, an die der Datensatz gespeichert werden muß,
2191       ; ist in DPH/L_next_Add
2192
2193       MOV DPL, DPL_next_Add       ; DPTR auf die Adresse, wo die Temp. hinge-
2194       MOV DPH, DPH_next_Add       ; schrieben werden soll, setzen
2195
2196       MOV I2C_W_Byte, Temp_H      ; High- und Low-Byte schreiben
2197       LCALL EEPROM_Byte_Write
2198       INC DPTR
2199       MOV I2C_W_Byte, Temp_L
2200       LCALL EEPROM_Byte_Write
2201                                   ; Jetzt muss das neune Ende des Datensatzes
2202                                   ; gespeichert werden, und zwar an die in
2203                                   ; DPH/L_Save gespeicherte Adresse
2204
2205       MOV DPH, DPH_Save                ; DPTR auf diese Adresse setzen
2206       MOV DPL, DPL_Save
2207
2208       MOV A, DPL_next_Add         ; Low-Byte um 2 erhöhen
```

```
2209        ADD A, #2
2210        MOV DPL_next_Add, A
2211
2212        MOV A, DPH_next_Add          ; Übertrag aufs High-Byte addieren
2213        ADDC A, #00
2214        MOV DPH_next_Add, A
2215                                     ; Der DPH/L_next_Add stehen jetzt auf der
2216                                     ; 1. freien Adresse nach dem Datensatz
2217        MOV I2C_W_Byte, A                ; Das High-Byte, das noch im Akku ist,
2218                                     ; in I2C_W_Byte schreiben
2219        LCALL EEPROM_Byte_Write
2220        INC DPTR
2221        MOV I2C_W_Byte, DPL_next_Add ; Das Low-Byte speichern
2222        LCALL EEPROM_Byte_Write
2223
2224
2225Temp_Aufn_Ende:
2226        POP A                        ; Akku mit den Tasterzuständen wiederherstellen
2227        RET
2228
2229
2230;-------------------------------------------------------------------------------
2231; Facharbeit-Teil   Daten während der Messwerterfassung in die 1. LCD-Zeile schreiben
2232;
2233; Ralf Wilke
2234;
2235
2236ErfData2LCD:
2237
2238        ACALL Temp2LCD
2239;       MOV LCD_ADDRESS, #07h        ; LCD-Adresse auf 7 setzten, da
2240;       LCALL LCD_Add_Set
2241
2242
2243        MOV A, Anzahl_H              ; Anzahl_H und Anzahl_L ausgeben, die
2244        LCALL LCD_Send_Zahl          ; enthalten, wieviele Messungen noch
2245        MOV A, Anzahl_L              ; zu machen sind
2246        LCALL LCD_Send_Zahl
2247        MOV A, #"/"                      ; Slash ausgeben als Trennung
2248        LCALL LCD_S_Char
2249        MOV A, Anzahl_H_ges          ; Anzahl_H_ges und Anzahl_L_ges ausgeben
2250        LCALL LCD_Send_Zahl          ; Diese RAMs enthalten die Gesamtanzahl
2251        MOV A, Anzahl_L_ges          ; der zu tätigenden Messungen
2252        LCALL LCD_Send_Zahl
2253        RET
2254
2255;-------------------------------------------------------------------------------
2256; Facharbeit-Teil   Temperatur auf das LCD schreiben
2257;
2258; Ralf Wilke
2259; Temp. ist in Temp_L/H
2260
2261Temp2LCD:          MOV A, Temp_H
2262        JNB ACC.0, T2L_positiv
2263        MOV A, Temp_L
2264        CPL A
2265        INC A
2266        CLR ACC.7
2267        PUSH A
2268        MOV A, #"-"
2269        SJMP T2L_Vorzeichen
2270T2L_Positiv:PUSH Temp_L
2271        MOV A, #"+"
2272T2L_Vorzeichen:
2273        MOV LCD_ADDRESS, #00h
2274        LCALL LCD_Add_Set
2275        LCALL LCD_S_Char
2276        POP A
2277        RRC A
2278
2279        LCALL LCD_Send_Zahl
2280        MOV A, #","
2281        LCALL LCD_S_Char
2282        JC T2L_halbgrad
2283        MOV A, #"0"
2284        SJMP T2L_halbgrad_out
2285T2L_halbgrad:
2286        MOV A, #"5"
2287T2L_halbgrad_out:
```

```
2288        LCALL LCD_S_Char
2289        MOV A, #DFh
2290        LCALL LCD_S_Char
2291        MOV A, #"C"
2292        LCALL LCD_S_Char
2293        RET
2294
2295
2296--------------------------------------------------------------------------------
2297; Facharbeit-Teil  LCD_Zeichen blinken
2298;
2299; Ralf Wilke
2300;
2301
2302
2303LCD_Blink_A: LJMP LCD_ADD_Set
2304LCD_Blink_S: LJMP LCD_S_Char
2305LCD_Blink_Z: LJMP LCD_Send_Zahl
2306LCD_Blink:
2307;       PUSH A
2308;       MOV A, B
2309;       INC A
2310;       JNZ LCDB10
2311;       SJMP LCDB99
2312;LCDB10:     MOV LCD_ADDRESS, A
2313;       LCALL LCD_Add_Set
2314;
2315;       JB blink, LCDB50
2316;       LCALL LCD_R_Char
2317;       MOV Blink1, A
2318;       LCALL LCD_Add_Set
2319;       MOV A, #" "
2320;       LCALL LCD_S_Char
2321;       LCALL LCD_R_Char
2322;       MOV Blink2, A
2323;       LCALL LCD_Add_Set
2324;       MOV A, #" "
2325;       LCALL LCD_S_Char
2326;       SJMP LCDB98
2327
2328;LCDB50:     MOV A, Blink1
2329;       LCALL LCD_S_Char
2330;       MOV A, Blink2
2331;       LCALL LCD_S_Char
2332
2333;LCDB98:     CPL blink
2334;LCDB99:     POP A
2335;       RET
2336
2337
2338        PUSH A
2339        JNB Intervall_H_onChange, not_Intervall_H
2340                                    ; Wenn nicht das Intervall_H ausgewählt ist,
2341                                    ; überspringen
2342        MOV LCD_ADDRESS, #7
2343        ACALL LCD_blink_A
2344        JB blink, Intervall_H_rewrite     ; wenn blink gesetzt ist, das auslöschen
2345                                    ; überspringen und die Zahlen wiederherstellen
2346;       MOV A, #" "                       ; < Leerzeichen > in den Akku
2347;       LCALL LCD_S_CHAR            ; 2 Mal ausgeben, und damit die Zahl überschreiben
2348;       LCALL LCD_S_CHAR
2349        ACALL 2blanks_lcd
2350        SJMP not_intervall_H        ; Fertig
2351
2352Intervall_H_rewrite:                      ; Wenn die Zahl wieder hingeschrieben werden
2353        MOV A, Intervall_H          ; soll, dies auch tun
2354        ACALL LCD_blink_Z
2355
2356
2357not_intervall_H:
2358        JNB Intervall_M_OnChange, not_Intervall_M
2359                                    ; Wenn nicht das Intervall_M ausgewählt ist,
2360                                    ; überspringen
2361        MOV LCD_ADDRESS, #10
2362        ACALL LCD_blink_A
2363
2364        JB blink, Intervall_M_rewrite ; wenn blink gesetzt ist, das auslöschen
2365                                    ; überspringen und die Zahlen wiederherstellen
2366;       MOV A, #" "                       ; < Leerzeichen > in den Akku
```

```
2367;       LCALL LCD_S_CHAR            ; 2 Mal ausgeben, und damit die Zahl überschreiben
2368;       LCALL LCD_S_CHAR
2369        ACALL 2blanks_lcd
2370        SJMP not_Intervall_M        ; Fertig
2371
2372Intervall_M_rewrite:                    ; Wenn die Zahl wieder hingeschrieben werden
2373        MOV A, Intervall_M          ; soll, dies auch tun
2374;       LCALL LCD_Send_Zahl
2375        ACALL LCD_Blink_Z
2376not_Intervall_M:
2377        JNB Intervall_S_OnChange, not_Intervall_S
2378                                    ; Wenn nicht das Intervall_S ausgewählt ist,
2379        MOV LCD_ADDRESS, #13        ; überspringen
2380
2381;       LCALL LCD_ADD_SET
2382        LCALL LCD_Blink_A
2383
2384        JB blink, Intervall_S_rewrite; wenn blink gesetzt ist, das auslöschen
2385                                    ; überspringen und die Zahlen wiederherstellen
2386;       MOV A, #" "                     ; < Leerzeichen > in den Akku
2387;       LCALL LCD_S_CHAR            ; 2 Mal ausgeben, und damit die Zahl überschreiben
2388;       LCALL LCD_S_CHAR
2389        ACALL 2blanks_lcd
2390        SJMP not_Intervall_S        ; Fertig
2391
2392Intervall_S_rewrite:                    ; Wenn die Zahl wieder hingeschrieben werden
2393        MOV A, Intervall_S          ; soll, dies auch tun
2394;       LCALL LCD_Send_Zahl
2395        ACALL LCD_blink_Z
2396not_Intervall_S:
2397        JNB Start_Day_OnChange, not_Start_Day
2398                                    ; Wenn nicht Start_Day ausgewählt ist,
2399                                    ; überspringen
2400        MOV LCD_ADDRESS, #10
2401;       LCALL LCD_ADD_SET
2402        ACALL LCD_blink_A
2403        JB blink, Start_Day_rewrite  ; wenn blink gesetzt ist, das auslöschen
2404                                    ; überspringen und die Zahlen wiederherstellen
2405;       MOV A, #" "                     ; < Leerzeichen > in den Akku
2406;       LCALL LCD_S_CHAR            ; 2 Mal ausgeben, und damit die Zahl überschreiben
2407;       LCALL LCD_S_CHAR
2408        ACALL 2blanks_lcd
2409        SJMP not_Start_Day          ; Fertig
2410
2411Start_Day_rewrite:                      ; Wenn die Zahl wieder hingeschrieben werden
2412        MOV A, Start_Day            ; soll, dies auch tun
2413;       LCALL LCD_Send_Zahl
2414        ACALL lcd_blink_Z
2415
2416Not_Start_Day:
2417        JNB Start_Month_OnChange, not_Start_Month
2418                                    ; Wenn nicht Start_Month ausgewählt ist,
2419                                    ; überspringen
2420        MOV LCD_ADDRESS, #13
2421;       LCALL LCD_ADD_SET
2422        ACALL LCD_Blink_A
2423        JB blink, Start_Month_rewrite ; wenn blink gesetzt ist, das auslöschen
2424                                    ; überspringen und die Zahlen wiederherstellen
2425;       MOV A, #" "                     ; < Leerzeichen > in den Akku
2426;       LCALL LCD_S_CHAR            ; 2 Mal ausgeben, und damit die Zahl überschreiben
2427;       LCALL LCD_S_CHAR
2428        ACALL 2blanks_lcd
2429        SJMP not_Start_Month        ; Fertig
2430
2431Start_Month_rewrite:                    ; Wenn die Zahl wieder hingeschrieben werden
2432        MOV A, Start_Month          ; soll, dies auch tun
2433;       LCALL LCD_Send_Zahl
2434        ACALL LCD_blink_Z
2435
2436not_Start_Month:
2437        JNB Start_Hour_OnChange, not_Start_Hour
2438                                    ; Wenn nicht Start_Hour ausgewählt ist,
2439                                    ; überspringen
2440        MOV LCD_ADDRESS, #10
2441;       LCALL LCD_ADD_SET
2442        ACALL LCD_Blink_A
2443        JB blink, Start_Hour_rewrite ; wenn blink gesetzt ist, das auslöschen
2444                                    ; überspringen und die Zahlen wiederherstellen
2445;       MOV A, #" "                     ; < Leerzeichen > in den Akku
```

```
2446;       LCALL LCD_S_CHAR            ; 2 Mal ausgeben, und damit die Zahl überschreiben
2447;       LCALL LCD_S_CHAR
2448        ACALL 2blanks_lcd
2449        SJMP not_Start_Hour         ; Fertig
2450
2451Start_Hour_rewrite:                          ; Wenn die Zahl wieder hingeschrieben werden
2452        MOV A, Start_Hour               ; soll, dies auch tun
2453;       LCALL LCD_Send_Zahl
2454        ACALL LCD_Blink_Z
2455
2456not_Start_Hour:
2457        JNB Start_Minute_OnChange, not_Start_Minute
2458                                    ; Wenn nicht Start_Minute ausgewählt ist,
2459                                    ; überspringen
2460        MOV LCD_ADDRESS, #13
2461;       LCALL LCD_ADD_SET
2462        ACALL LCD_blink_A
2463        JB blink, Start_Minute_rewrite; wenn blink gesetzt ist, das auslöschen
2464                                    ; überspringen und die Zahlen wiederherstellen
2465;       MOV A, #" "                     ; < Leerzeichen > in den Akku
2466;       LCALL LCD_S_CHAR            ; 2 Mal ausgeben, und damit die Zahl überschreiben
2467;       LCALL LCD_S_CHAR
2468        ACALL 2blanks_lcd
2469        SJMP not_Start_Minute       ; Fertig
2470
2471Start_Minute_rewrite:                   ; Wenn die Zahl wieder hingeschrieben werden
2472        MOV A, Start_Minute         ; soll, dies auch tun
2473;       LCALL LCD_Send_Zahl
2474        ACALL LCD_blink_Z
2475
2476not_Start_Minute:
2477        JNB Anzahl_H_OnChange, not_Anzahl_H
2478                                    ; Wenn nicht Anzahl_H ausgewählt ist,
2479                                    ; überspringen
2480        MOV LCD_ADDRESS, #8
2481;       LCALL LCD_ADD_SET
2482        ACALL lcd_blink_A
2483        JB blink, Anzahl_H_rewrite  ; wenn blink gesetzt ist, das auslöschen
2484                                    ; überspringen und die Zahlen wiederherstellen
2485;       MOV A, #" "                     ; < Leerzeichen > in den Akku
2486;       LCALL LCD_S_CHAR            ; 2 Mal ausgeben, und damit die Zahl überschreiben
2487;       LCALL LCD_S_CHAR
2488        ACALL 2blanks_lcd
2489        SJMP not_Anzahl_H               ; Fertig
2490
2491Anzahl_H_rewrite:                       ; Wenn die Zahl wieder hingeschrieben werden
2492        MOV A, Anzahl_H             ; soll, dies auch tun
2493;       LCALL LCD_Send_Zahl
2494        ACALL LCD_Blink_Z
2495
2496not_Anzahl_H:
2497        JNB Anzahl_L_OnChange, not_Anzahl_L
2498                                    ; Wenn nicht Anzahl_L ausgewählt ist,
2499                                    ; überspringen
2500        MOV LCD_ADDRESS, #10
2501;       LCALL LCD_ADD_SET
2502        ACALL LCD_Blink_A
2503        JB blink, Anzahl_L_rewrite  ; wenn blink gesetzt ist, das auslöschen
2504                                    ; überspringen und die Zahlen wiederherstellen
2505;       MOV A, #" "                     ; < Leerzeichen > in den Akku
2506;       LCALL LCD_S_CHAR            ; 2 Mal ausgeben, und damit die Zahl überschreiben
2507;       LCALL LCD_S_CHAR
2508        ACALL 2blanks_lcd
2509        SJMP not_Anzahl_L               ; Fertig
2510
2511
2512Anzahl_L_rewrite:                       ; Wenn die Zahl wieder hingeschrieben werden
2513        MOV A, Anzahl_L             ; soll, dies auch tun
2514;       LCALL LCD_Send_Zahl
2515        ACALL LCD_Blink_Z
2516not_Anzahl_L:
2517
2518        CPL blink                   ; Bit invertieren
2519        POP A                       ; Akku wiederherstellen
2520        RET
2521
2522
2523
2524;-------------------------------------------------------------------------------
```

```
2525; Facharbeit-Teil   Daten über die serielle Schnittstelle senden
2526;
2527; Ralf Wilke
2528;
2529; Übergabe im Akku
2530
2531Ser_Send:
2532        JB Int_is_running, Ser_Send_Int
2533;       LCALL Ser_Send_without_Int
2534;       RET
2535        LJMP Ser_Send_without_Int    ; kein LCALL nötig, da direkt nach dem LCALL
2536                                     ; ein RET folgt, und so auch das RET der auf-
2537                                     ; gerufenen Routine richtig zurückspringt
2538                                     ; spart 2 Byte Stack
2539Ser_Send_Int:
2540;       LCALL Ser_Send_while_Int
2541;       RET
2542        LJMP Ser_Send_while_Int          ; kein LCALL nötig, da direkt nach dem LCALL
2543                                     ; ein RET folgt, und so auch das RET der auf-
2544                                     ; gerufenen Routine richtig zurückspringt
2545                                     ; spart 2 Byte Stack
2546
2547
2548
2549;-------------------------------------------------------------------------------
2550; Facharbeit-Teil   Daten über die serielle Schnittstelle senden ausserhalb von Interrupts
2551;
2552; Ralf Wilke
2553;
2554; Übergabe im Akku
2555
2556
2557Ser_Send_without_Int:
2558        SETB Ser_Int_Lock                ; Diese Routine darf nicht von einem
2559                                     ; Interrupt der ser.-Schnittstelle unter-
2560                                     ; brochen werden
2561        MOV SBUF, A                      ; Akku in den Sendebuffer schreiben und
2562                                     ; dadurch aussenden
2563Ser_wait:    JNB Ser_ready, Ser_wait           ; Warten, bis Interrupt-Routine das Ende
2564                                     ; der Aussendung über SCON.1 = TI fest-
2565                                     ; gestellt, und Ser_ready gesetzt hat.
2566        CLR Ser_ready                ; Dann Ser_ready wieder löschen
2567        CLR Ser_Int_lock             ; Nun wieder Unterbrechung möglich
2568        JNB Ser_Int_arrived, Ser_Send_end
2569                                     ; Wenn zwischenzeitlich kein Empfangs-
2570                                     ; Interrupt da war, zu Ende
2571        SETB SCON.0                      ; Empfangs-Interrupt auslösen
2572Ser_Send_end:
2573        RET
2574
2575
2576;-------------------------------------------------------------------------------
2577; Facharbeit-Teil   Daten über die ser. Schnittstelle senden während eines Interrupts
2578;
2579; Ralf Wilke
2580;
2581; Übergabe im Akku
2582;
2583Ser_Send_while_Int:
2584        CLR SCON.1                   ; TI löschen, um bei wieder gesetztem
2585                                     ; Bit das Ende der Aussendung feststellen
2586                                     ; zu können
2587        MOV SBUF, A                      ; Akku in den Sendebuffer schreiben und
2588                                     ; dadurch aussenden
2589Ser_Send_while_Int_wait:
2590        JNB SCON.1, Ser_Send_while_Int_wait
2591                                     ; Warten, bis das TI-Flag durch das Ende der
2592                                     ; Warten, bis das TI, Bit gesetzt ist,
2593                                     ; das das Ende der Aussendung anzeigt
2594        CLR SCON.1                   ; TI-Flag wieder löschen, damit nicht am
2595                                     ; Ende der Interrupt-Routine
2596                                     ; der Interrupt erneut ausgelöst wird.
2597        RET
2598
2599
2600;-------------------------------------------------------------------------------
2601; Facharbeit-Teil   Interrupt-Routine zum Empfang mit der seriellen Schnittstelle
2602;
2603; Ralf Wilke
```

```
2604;
2605
2606
2607Ser_Int:           PUSH PSW
2608        PUSH A
2609        SETB Int_is_running          ; Anzeigen, dass nun ein Interrupt läuft
2610
2611        JB SCON.1, Ser_Int_from_TX   ; Wenn der Interrupt durch das Ende einer
2612                                     ; AUSSENDUNG ausgelöst wurde, zur
2613                                     ; Senderoutine
2614        SETB Ser_Int_arrived         ; Merken, dass ein Zeichen empfangen wurde
2615        CLR SCON.0                   ; RI-Flag löschen, damit der Interrupt nicht
2616                                     ; direkt wieder aufgerufen wird
2617        JB Ser_Int_lock, ser_int_end_NC
2618                                     ; Wenn vor dem Interrupt eine Routine lief,
2619                                     ; die nicht unterbrochen werden darf,
2620                                     ; die Interruptroutine verlassen
2621                                     ; Sonst:
2622        MOV A, SBUF                       ; Byte aus dem Buffer holen
2623
2624                                     ; Byte ist jetzt zur weiteren
2625                                     ; Verarbeitung im Akku
2626        CJNE A, #"e", Not_Hex_Dump   ; Wenn das gesendete Byte nicht "d" ist,
2627        LCALL SNAP                   ; überspringen, sonst Hex-Dump ausgeben
2628Not_Hex_Dump:
2629        CJNE A, #"r", Not_SNAP
2630        LCALL SNAP
2631Not_SNAP:          CJNE A, #"M", Not_Memory
2632        ACALL EEPROM_OUT
2633Not_Memory:
2634
2635
2636;       ACALL CRLF
2637;       LCALL Ser_Send_while_Int
2638;       ACALL CRLF
2639
2640
2641        CLR Ser_Int_arrived          ; Interrupt ist jetzt abgearbeitet, dies der
2642                                     ; Routine, die evtl. diesen Interrupt
2643                                     ; nachträglich aufgerufen hat, mitteilen
2644        SJMP Ser_int_end_NC
2645
2646Ser_Int_from_TX:                          ; Wenn der Interrupt durch das Ende einer
2647                                     ; Aussendung ausgelöst wurde...
2648        CLR SCON.1                   ; TI löschen, damit der Interrupt nicht
2649                                     ; sofort wieder ausgelöst wird
2650        SETB Ser_ready               ; der Sende-Routine die Mitteilung hinter-
2651                                     ; lassen, dass das Zeichen weg ist
2652Ser_Int_end_NC:
2653        CLR Int_is_running           ; Anzeigen, dass kein Interrupt mehr läuft
2654        POP A
2655        POP PSW
2656        RETI
2657
2658;-------------------------------------------------------------------------------
2659; Facharbeit-Teil   Wandelt ein Byte in die hexadizimale Schreibweise um und sendet
2660;                   und sendet es über die ser. Schnittstelle aus
2661;
2662; Ralf Wilke
2663;
2664;  Übergabe: Akku
2665
2666Hex_Send:
2667        PUSH A                       ; Akku für später sichern
2668        ANL A, #11110000b                 ; nur die ersten 4 Bits wandeln
2669        SWAP A                       ; Nibbles vertauschen
2670        PUSH A                       ; und diesen Zustand sichern
2671        CLR C                        ; Carry zur Auswertung erst mal löschen
2672        SUBB A, #10                       ; 10 vom Akku abziehen
2673        POP A                        ; Das Carry-Flag ist jetzt gesetzt, wenn der
2674                                     ; Akku einen Wert von 0-9 hatte,
2675        JNC Ah_bis_Fh                ; Wenn nicht, springen
2676        ADD A, #30h                       ; Wenn doch, 30h addieren, um die
2677        SJMP Send_Hex_Aussendung     ; ASCII-Ziffer zu erhalten
2678Ah_bis_Fh:   ADD A, #37h                       ; sonst 37h addieren, um den ASCII-Buchstaben
2679Send_Hex_Aussendung:
2680        LCALL Ser_Send               ; Abschicken...
2681        POP A                        ; Ursprünglichen Wert holen
2682        PUSH A
```

```
2683       ANL A, #00001111b                ; nur die lezten 4 Bits wandeln
2684       PUSH A                      ; und gleich wieder sichern
2685       CLR C                       ; Carry zur Auswertung erst mal löschen
2686
2687       SUBB A, #10                      ; 10 vom Akku abziehen
2688       POP A                       ; die 4 LSBs wieder zurückholen
2689       JNC Ah_bis_Fh2              ; Wenn > 9 , springen
2690       ADD A, #30h                      ; ASCII-Ziffer ermitteln
2691       SJMP Send_Hex_Aussendung2
2692Ah_bis_Fh2:  ADD A, #37h                     ; ASCII-Buchstaben ermitteln
2693Send_Hex_Aussendung2:
2694       LCALL Ser_Send              ; Abschicken..
2695;      MOV A, #"h"
2696;      LCALL Ser_Send
2697       MOV A, #" "
2698       LCALL Ser_Send
2699       POP A
2700       RET
2701
2702
2703
2704
2705
2706
2707;-------------------------------------------------------------------------------
2708; Facharbeit-Teil   Hex-Dump des EEPROM-Speichers über die ser. Schnittstelle ausgeben
2709;
2710; Ralf Wilke
2711;
2712
2713;Hex_Dump:   PUSH A
2714;      MOV DPTR, #0000h            ; Am Anfang anfangen
2715;Hex_Dump_Next_Byte:
2716;      LCALL EEPROM_Byte_read      ; Byte holen
2717;      INC DPTR                    ; DPTR auf nächstes Byte zeigen lassen
2718;      MOV A, I2C_R_Byte
2719;      LCALL Hex_Send              ; Byte verschicken
2720;      MOV A, DPL
2721;      ANL A, #00000111b
2722;      CJNE A, #00000111b, Hex_Dump_no_CR_LF
2723                                  ; Wenn 8 Byte ausgegeben wurden, CR/LF senden
2724;      ACALL CRLF
2725;Hex_dump_no_CR_LF:
2726;      MOV A, DPL                  ; Den ganzen Speicher ausgeben
2727;      CJNE A, #00100000b, Hex_Dump_next_Byte ; bzw. Teile
2728;      ACALL CRLF
2729;      POP A
2730;      RET
2731
2732
2733
2734;-------------------------------------------------------------------------------
2735; Facharbeit-Teil   Routine für den Interrupt von Timer 0
2736;
2737; Auswertung von Intervall_H bis Intervall_S, Anzahl_H und _L,
2738; Aktivierung der Messung und Speicherung des Messwertes
2739;
2740;
2741; Ralf Wilke
2742;
2743
2744Timer_Int:   MOV TH0, #3Ch               ; Timer 0 wieder nachladen
2745       MOV TL0, #B0h
2746       PUSH PSW                    ; Akku und PSW sichern
2747       PUSH A
2748       SETB PSW.3                  ; Registerbank 1 08h-0Fh auswählen
2749       CLR PSW.4
2750
2751;      DJNZ R6, Timer_Int_Reti          ; R6 dekrementieren, wenn 0, dann
2752;      MOV R6, #100                ; wieder nachladen, sonst RETI
2753;      DJNZ R7, Timer_Int_Reti          ; R7 dekrementieren, wenn 0 dann
2754;      MOV R7, #40                      ; wieder nachladen, sonst RETI
2755                                  ; Wenn die Routine hier angelangt ist, dann
2756                                  ; sind 250µs * 100 * 40 = 1.000.000µs =  1s
2757                                  ; vergangen
2758       DJNZ R7, Timer_Int_Reti          ; R7 dekrementieren
2759       MOV R7, #20                      ; R7 nachladen
2760                                  ; Wenn die Routine hier angelangt ist, dann
2761                                  ; sind 50ms * 20 = 1.000ms = 1s
```

```
2762                                    ; vergangen
2763
2764;
2765;       DEC R5                      ; R5 = Sekunden verringern
2766;       CJNE R5, #00, Timer_Int_Reti ; Testen ob Sekunden = 0
2767;       CJNE R4, #00, Timer_Int2    ; ...    ob Minuten = 0
2768;       CJNE R3, #00, Timer_Int3    ; ...    ob Stunden = 0
2769;
2770;       MOV R3, Intervall_H         ; Wenn alles 0 ist, Register nachladen
2771;       MOV R4, Intervall_M
2772;       MOV R5, Intervall_S
2773
2774;       DJNZ R5, Timer_Int_Reti
2775;       MOV R5, #59
2776;       DJNZ R4, Timer_Int_Reti
2777;       MOV R4, #59
2778;       DJNZ R3, Timer_Int_Reti
2779
2780;       MOV R3, Intervall_H         ; Wenn alles 0 ist, Register nachladen
2781;       MOV R4, Intervall_M
2782;       MOV R5, Intervall_S
2783
2784        ; R345 um 1 dekrementieren
2785        CLR C                       ; Carry löschen
2786        MOV A, #1                   ; 1 in den Akku
2787        XCH A, R5                   ; Akku und R5 tauschen
2788        SUBB A, R5                  ; 1 von R5 abziehen
2789        XCH A, R5                   ; R5 wieder zurücktauschen
2790        CLR A                       ; Akku löschen, es soll ja nur einmal 1
2791                                    ; abgezogen werden
2792        XCH A, R4                   ; Ein evtl. Borrow muss sich noch weiter
2793                                    ; wirken; R4 und Akku tauschen
2794        SUBB A, R4                  ; vom Akku R4 =  0 und Carry abziehen
2795        XCH A, R4                   ; zurücktauschen
2796        XCH A,R3                    ; R3 mit Akku tauschen
2797        SUBB A, R3                  ; vom Akku R3 = 0 und Carry abziehen
2798        XCH A, R3                   ; R3 zurücktauschen
2799
2800        ; prüfen ob R345 = 0
2801        MOV A, R5
2802        JNZ Timer_Int_Reti
2803        MOV A, R4
2804        JNZ Timer_Int_Reti
2805        MOV A, R3
2806        JNZ Timer_Int_Reti
2807
2808        ; Hier ist R345 = 0
2809        LCALL create_24Bit_counter  ; Counter nachladen
2810        SETB mach_Messung                ; Bit setzten und damit nach
2811                                    ; außen Signalisieren
2812;       SJMP Timer_Int_Reti         ; und zum Ende springen
2813
2814;Timer_Int3: DEC R3                     ; Wenn Minuten und Sekunden = 0
2815;       MOV R4, #59                      ; Diese Register nachladen und die
2816;       MOV R5, #59                      ; Stunden dekrementieren
2817;       SJMP Timer_Int_Reti         ; Zum Ende
2818
2819
2820;Timer_Int2: DEC R4                          ; Wenn Sekunde = 0
2821;       MOV R5, #59                      ; Nachladen und Minute dekremenieren
2822;       SJMP Timer_Int_Reti         ; Zum Ende
2823
2824
2825Timer_Int_Reti:                          ; Hier ist Ende
2826        POP A
2827        POP PSW
2828        RETI
2829
2830
2831;-------------------------------------------------------------------------------
2832; Facharbeit-Teil   24-Bit-Zähler, der die Intervallzeit in Sekunden enthält
2833;
2834; Ralf Wilke
2835;
2836
2837create_24Bit_Counter:
2838
2839        MOV A, Intervall_M          ; Minuten in Akku
2840        MOV B, #60                  ; Multiplikationsfaktor in B
```

```
2841        MUL AB                      ; Multiplizieren
2842        MOV R5, A                   ; Das Low-Byte in R5
2843        MOV R4, B                   ; Das High-Byte in R4
2844        MOV R3, #00h                ; R3 auf 00h
2845        MOV A, Intervall_H          ; Wenn Intervall_H ist, überspringen
2846        MOV R2, A                   ; R2 = Stundenzähler
2847        JZ C24BC_no_hour            ; sonst 0E10h = 3600d zu R345 addieren
2848C24BC_next_hour:
2849        MOV A, #10h                      ; 10h in Akku
2850        ADD A, R5                   ; mit R5 addieren
2851        MOV R5, A                   ; R5 zuückschreiben
2852        MOV A, #0Eh                      ; 0Eh in Akku
2853        ADDC A, R4                  ; Mit Carry mit R4 addieren
2854        MOV R4, A                   ; R4 zurückschreiben
2855        CLR A                       ; 00h in Akku, da 3600d = 00 0E 10h
2856        ADDC A, R3                  ; Mit Carry addieren
2857        MOV R3, A                   ; R3 zurückschreiben
2858        DJNZ R2, C24BC_next_hour    ; Stundenzähler dec. und ggf. nochmal
2859C24BC_no_hour:
2860        MOV A, Intervall_S          ; Die Sekunden noch aufaddieren
2861        ADD A, R5                   ; erstmal zu R5
2862        MOV R5, A                   ; R5 zurückschreiben
2863        CLR A                       ; Akku löschen
2864        ADDC A, R4                  ; und nur noch das Carry addieren
2865        MOV R4, A                   ; R4 zurückschreiben
2866        CLR A
2867        ADDC A, R3
2868        MOV R3, A                   ; R3 zurückschreiben
2869        RET                         ; Rücksprung
2870
2871
2872;-------------------------------------------------------------------------------
2873; Facharbeit-Teil   Kleine Hilfsroutinen
2874;
2875; Ralf Wilke
2876;
2877
2878CRLF:       PUSH A                      ; Gibt ASCII 13 und ASCII 10 aus
2879        MOV A, #10
2880        LCALL Ser_Send
2881        MOV A, #13
2882        LCALL Ser_Send
2883        POP A
2884        RET
2885
2886blank:      PUSH A                      ; Gibt Leerzeichen aus
2887        MOV A, #" "
2888        LCALL Ser_Send
2889        POP A
2890        RET
2891
28922blanks_lcd: PUSH A
2893        MOV A, #" "
2894        LCALL LCD_S_Char
2895        LCALL LCD_S_Char
2896        POP A
2897        RET
2898
2899
2900LCD_blank:   PUSH A
2901        MOV A, #" "
2902        LCALL LCD_S_Char
2903        POP A
2904        RET
2905
2906;-------------------------------------------------------------------------------
2907; Facharbeit-Teil   Snap-Shot der wichtigsten Register und des RAMs über die COM
2908; Neugestatung: RAM und EEPROM-Speicher
2909; Ralf Wilke
2910;
2911
2912EEPROM_OUT:  PUSH DPH
2913        PUSH DPL
2914        MOV DPTR, #00
2915        PUSH I2C_R_Byte
2916EEPROM_OUT_WDH:
2917        ACALL EEPROM_Byte_Read
2918        INC DPTR
2919        MOV A, I2C_R_Byte
```

```
2920        ACALL Ser_Send
2921        MOV A, DPH
2922        CJNE A, #00010000b, EEPROM_OUT_WDH
2923        POP I2C_R_Byte
2924        POP DPL
2925        POP DPH
2926        RET
2927
2928
2929;snap2:            PUSH A
2930;       MOV A, #"r"
2931;       ACALL snap
2932;       MOV A, #"e"
2933;       ACALL CRLF
2934;       ACALL snap
2935;       POP A
2936;       ACALL CRLF
2937;       ACALL CRLF
2938;       RET
2939
2940
2941SNAP:         PUSH PSW
2942        SETB PSW.4                  ; Registerbank 2 auswählen
2943        CLR PSW.3
2944        MOV R7, DPH                       ; DPTR retten
2945        MOV R6, DPL
2946        MOV R5, I2C_R_Byte          ; I2C_R_Byte retten
2947        MOV R4, A                   ; Akku retten
2948        MOV DPTR, #0000h            ; DPTR auf den Anfang stellen
2949        MOV A, #"-"                       ; 3 Mal "-" ausgeben, um eine Tabelle
2950        ACALL Ser_Send              ; zu erzeugen
2951        ACALL Ser_Send
2952        ACALL Ser_Send
2953
2954        ; Jetzt wird erstmal die Spaltenbeschriftung ausgegeben
2955
2956        MOV A, #FFH                       ; Akku auf FFH setzten
2957SNAP01:            MOV R0, #8                  ; Zählregister R0 auf 8
2958SNAP06:            INC A                       ; Akku erhöhen
2959        ACALL Hex_Send              ; Akku im Hexadezimalsystem ausgeben
2960        DJNZ R0, SNAP06             ; Das ganze 8 Mal
2961        CJNE A, #7, SNAP09          ; Wenn der Akku dann auf 7 steht, also
2962                                    ; von 0 aus 8 Byte ausgegeben wurden
2963        ACALL blank                       ; ein Leerzeichen einfügen
2964                                    ; sonst weiterspringen
2965        SJMP SNAP01                       ; und nochmal
2966
2967
2968SNAP09:            ACALL CRLF                  ; Neue Zeile
2969        CLR A                       ; Akku und R0 auf 0 (insgesamt nur 2 Byte
2970        MOV R0, A                   ; Programmspeicherplatz !)
2971        ACALL Hex_Send              ; Die 16er-Stelle als Zeilenbeschriftung ausgeben
2972
2973        ; Hier folgt die Unterscheidung RAM < - > EEPRROM
2974SNAP10:            CJNE R4, #"e", SNAP11       ; R4 enthält das vom Terminal empfangene
   Zeichen
2975                                    ; ist dieses NICHT "e" = EERPOM, zur
2976                                    ; RAM-Routine springen
2977        LCALL EEPROM_Byte_read      ; und ein Byte aus dem EEPROM lesen
2978        MOV A, I2C_R_Byte                 ; Diese Byte in den Akku...
2979        INC DPTR                    ; Den DPTR noch inkrementiert
2980        SJMP SNAP12                       ; und fertig.
2981
2982SNAP11:      ; Hier wird das RAM ausgelesen
2983        MOV A, @R0                  ; und die von R0 gezeigte Adresse lesen
2984
2985SNAP12:      ; Hier beginnt wieder der gemeinsame Teil
2986
2987        ACALL Hex_Send              ; Byte als Hex ausgeben
2988                                    ; Es folgen einige Unterscheidungen
2989
2990                                    ; Fertig ?
2991        CJNE R0, #FFh, SNAP14       ; Wenn wir NICHT schon FF Byte gelesen haben,
2992                                    ; zur nächsten Abfrage springen, sonst
2993        ACALL CRLF                  ; Neue Zeile...
2994        SJMP SNAP99                       ; Und Schluss
2995
2996
2997SNAP14:                                   ; Neue Zeile ?
```

```
2998        MOV A, R0                   ; Den Zähler R0 in den Akku
2999        ANL A, #00001111b                ; Nur das Low-Nibble maskieren
3000        CJNE A, #15, SNAP15         ; Ist es NICHT 15, weiterspringen
3001        ACALL CRLF                  ; sonst neue Zeile
3002        INC R0                      ; R0 erhöhen
3003        MOV A, R0                   ; und in den Akku kopieren
3004        ACALL Hex_Send              ; um die neue Zeilenbeschriftung auszugeben
3005        SJMP SNAP10                      ; nächstes Byte holen
3006
3007SNAP15:                                 ; Leerzeichen nach 8 Byte ?
3008        MOV A, R0                   ; R0 in den Akku
3009        ANL A, #00000111b                ; Die unteren 3 Bits maskieren
3010        CJNE A, #7, SNAP20          ; sind diese NICHT alle gesetzt, weiterspringen
3011        ACALL blank                      ; sonst Leerzeichen einfügen
3012        SJMP SNAP20
3013
3014SNAP20:            INC R0                    ; R0 erhöhen
3015        SJMP SNAP10                 ; und von vorn das nächste Byte holen
3016
3017SNAP99:
3018        MOV A, R4                   ; Alle Register wieder zurück
3019        MOV I2C_R_Byte, R5
3020        MOV DPL, R6
3021        MOV DPH, R7
3022        POP PSW                     ; Wieder die ursprüngliche Registerbank
3023        RET
3024
3025;-------------------------------------------------------------------------------
3026; Facharbeit-Teil   Begrüßungsmeldung über das LC-Display ausgeben
3027;
3028; Ralf Wilke
3029;
3030
3031Pwr_on_Msg:  PUSH A                      ; Akku sichern
3032        MOV LCD_ADDRESS, #00
3033        LCALL LCD_ADD_SET
3034        MOV DPTR, #Welcome_DB       ; Adresse des ersten Datenfeldes in den DPTR
3035;       MOV R7, #64
3036        MOV R7, #00
3037POM_Next_Byte:
3038        CLR A
3039        MOVC A, @A+DPTR             ; Byte aus dem Speicher holen
3040        LCALL LCD_S_Char            ; auf LCD ausgeben
3041        INC DPTR                    ; Datenpointer erhöhen
3042;       MOV R7, DPL                      ; Das Datenfeld beginnt bei 0D00h, deshalb
3043                                    ; ist das Low-Byte hier für die Bestimmung
3044        INC R7                      ; ausreichend
3045        CJNE R7, #64, Check_EOL          ; Wenn NICHT das 64te Byte gerade gelesen
3046                                    ; wurde, überprüfen, ob die Zeile zu Ende ist
3047        SJMP POM_end_data_transfer  ; SONST ist der Datentransfer fertig
3048Check_EOL:   CJNE R7, #32, POM_Next_Byte  ; Wenn das EndOfLine (EOL) noch nicht erreicht
3049                                    ; ist, nächstes Byte ausgeben, sonst:
3050        MOV LCD_ADDRESS, #40h       ; Cursor auf Anfang der 2. Zeile setzen
3051        LCALL LCD_ADD_SET                ; dieses auch wirklich tun
3052        SJMP POM_Next_Byte          ; die 2. Zeile ausgeben
3053POM_end_data_transfer:
3054
3055
3056; Dieser Init-Teil ist hier untergebracht, da in dieser Stelle vom dem Scrollen des
3057; Textes eine 2 sekündige Pause abgewartet wird. Laut Datenblatt des DS 1620 dauert
3058; eine Temperaturwandlung 400 bis 1000 ms. Diese Wartezeit, die nach dem Senden des
3059; Start-Protokolls gewartet werden muss, ist jetzt geschickt in den 2 Sekunden Warte-
3060; zeit der PowerOnMeldung enthalten; eine Verzögerung der Erscheinens der Main-Menüs,
3061; wie es auftreten würde, ständen diese Zeilen am Anfang im eingentlichen Init-Teil,
3062; wird somit vermieden.
3063
3064
3065DS1620_Init: MOV A, #EEh                        ; EEh = Start Conversion
3066        LCALL proto_DS1620
3067        CLR ds_rst
3068        LCALL wr_status_ausgang
3069
3070; Ab hier geht die normale POM-Routine weiter
3071        MOV wait_time_025s, #8
3072        ACALL wait_025s             ; wait_time_025s * 0,25s warten
3073        MOV R7, #16                      ; 16 Mal das Display scrollen
3074        CLR LCD_RS                  ; Auf LCD-Kommando umschalten
3075        MOV A,#00011000b            ; Display und Cursor nach rechts bewegen
3076POM_next_shift:
```

```
3077        LCALL LCD_WR                ; ausgeben
3078        MOV wait_time, #150         ; 150ms warten
3079        ACALL wait_ms
3080        DJNZ R7, POM_next_shift           ; das Ganze 16 Mal
3081        MOV wait_time_025s, #12           ; 12*0,25 s warten
3082        ACALL wait_025s
3083        MOV A,#00000001b            ; Clear display
3084        LCALL LCD_WR
3085        MOV A,#00000010b            ; Return Home
3086        LCALL LCD_WR
3087        MOV A, #"O"
3088        LCALL Ser_send
3089        MOV A, #"N"
3090        LCALL Ser_Send
3091        ACALL CRLF
3092        POP A
3093        RET
3094
3095;-------------------------------------------------------------------------------
3096
3097
3098
3099wait_025s:   MOV wait_time, #250         ; Pro Schleife 250ms = 0,25s warten
3100        ACALL wait_ms
3101        DJNZ wait_time_025s, wait_025s
3102        RET
3103
3104wait_ms:
3105_wait_ms1:   MOV wait_time1, #249
3106_wait_ms2:   NOP
3107        NOP
3108        DJNZ wait_time1, _wait_ms2
3109        DJNZ wait_time, _wait_ms1
3110        RET
3111
3112
3113
3114wait_100us:  MOV wait_time1, #49
3115_wait_100us1:
3116        DJNZ wait_time1, _wait_100us1
3117        RET
3118
3119;-------------------------------------------------------------------------------
3120
3121The_End:     SJMP The_End
3122;-------------------------------------------------------------------------------
3123
3124
3125
3126
3127
3128
3129
3130(0DEEh):
3131Welcome_DB:
3132        DB: "Mikrocontroller "      ; Reihenfolge nachher: 1
3133        DB: "   Ralf Wilke   "      ;                      3
3134        DB: "  Thermometer   "      ;                      2
3135        DB: "Version 23.12.01"      ;                      4
3136
3137Main_DB:     DB: "Men", F5h, " Save    Aus"
3138
3139Sure_End_DB:
3140        DB: "  Wirklich aus- "
3141        DB: "schalten? Yes No"
3142
3143Power_Off_DB:
3144        DB: "Sie k", EFh, "nnen jetzt"
3145        DB: "   abschalten", A6h
3146
3147OkEscUpDown_DB:
3148        DB: "OK  ESC  Up Down"
3149
3150Main_Menü1_DB:
3151        DB: "Me", E2h, "werterfassung"
3152Main_Menü2_DB:
3153        DB: "Datenverwaltung "
3154Main_Menü3_DB:
3155        DB: "   Statistik", A6h
```

```
3156;Main_Menü4_DB:
3157;       DB: " Einstellungen", A6h
3158
3159Sure_Del_All_DB:
3160        DB: " Wirklich alles "
3161Sure_Del_DB:
3162        DB: "l", EFh, "schen? Ja Nein"
3163
3164Del_ALL_working_DB:
3165        DB: " L", EFh, "schung aktiv "
3166Del_Datensatz_DB:
3167        DB: "Datensatz Nr.", A6h
3168
3169
3170Meßwert_Eingabe:
3171        DB: "OK Zur", F5h, "ck ", 00h, "   ", 7Eh, " "
3172
3173Meßwert_Menü1_DB:
3174        DB: "Abstand", A6h
3175Meßwert_Menü2_DB:
3176        DB: "Startdatum", A6h
3177Meßwert_Menü3_DB:
3178        DB: "Startzeit", A6h
3179Meßwert_Menü4_DB:
3180        DB: "Anzahl", A6h
3181MM4_nicht_genug_Speicher:
3182        DB: "Leider zu gro", E2h, A6h
3183        DB: "Max.      OK", A6h
3184Meßwert_Menü5_DB:
3185        DB: "Erfassung   Stop"
3186Meßwert_Erf_RDY_DB:
3187        DB: "** Fertig **  OK"
3188Meßwert_Stop_Sure_DB:
3189        DB: "Temp.-Erfassung "
3190        DB: "stoppen? Ja Nein"
3191Meßwert_Canceled_DB:
3192        DB: "Temp.-Erfassung "
3193Canceled_OK_DB:
3194        DB: "abgebrochen  OK "
3195
3196Datenverw._Menü1_DB:
3197        DB: "DatensatzL",  EFh, "schen"
3198Datenverw._Menü2_DB:
3199        DB: " Alles l", EFh, "schen", A6h
3200
3201EEPROM_Leer_DB:
3202        DB: "Speicher leer,", A6h
3203
3204Statistik_Menü1_DB:
3205        DB: "   Datens", E1h, "tze", A6h
3206Statistik_Menü2_DB:
3207        DB: "   Noch  frei", A6h
3208Statistik_BytesFree_DB:
3209        DB: " Bytes frei ", A6h
3210
3211
```

```
3212; Include Datei  I²C-Bus
3213
3214
3215;-------------------------------------------------------------------------------
3216; Facharbeit-Teil  I²C-Bus - START condition
3217;
3218; Ralf Wilke
3219;
3220
3221
3222
3223I2C_Start:   SETB SCL_NS                      ; SCL_NS auf 1
3224       LCALL WR_Status_Ausgang           ; Zeit CHDX = 600ns
3225       LCALL wait_100us
3226       CLR SDA                     ; die eigentliche Startbedingung
3227       LCALL wait_100us
3228       CLR SCL_NS
3229       LCALL WR_Status_Ausgang           ; Zeit CHCL =  600ns
3230       RET                         ; + DLCL = 600ns
3231
3232
3233;-------------------------------------------------------------------------------
3234; Facharbeit-Teil  I²C-Bus - STOP condition
3235;
3236; Ralf Wilke
3237;
3238
3239
3240I2C_Stop:          SETB SCL_NS
3241       LCALL WR_Status_Ausgang           ; CHDH = 600ns
3242       LCALL wait_100us
3243       SETB SDA                    ; die eigentliche Stopbedingung
3244       LCALL wait_100us
3245       NOP                         ; DHDL = 1,3µs
3246       CLR SCL_NS                  ; EEPROM deaktivieren, so dass eine Start-
3247                                   ; bedingung nicht erkannt wird. So lassen
3248                                   ; sich die beiden I2C-Busse an einer SDA-
3249                                   ; Leitung betreiben
3250       LCALL WR_Status_Ausgang
3251       RET
3252
3253
3254;-------------------------------------------------------------------------------
3255; Facharbeit-Teil  I²C-Bus - Device_Selection_Byte ermitteln
3256;
3257; Ralf Wilke
3258;
3259
3260I2C_Dev_Sel:
3261       PUSH A                      ; Akku retten
3262       MOV DEV_SEL, #10100000b           ; feste EEPROM-Adresse
3263       MOV A, DPH
3264       ANL A, #00000111b                 ; es sind nur die Bits 0-2 nötig
3265       RL A                        ; einmal nach links schieben, um für
3266                                   ; das R/'W-Bit Platz zu machen
3267       ORL DEV_SEL, A              ; Die 3 Bits dem RAM DEV_SEL hinzufügen
3268       POP A
3269       RET
3270
3271
3272;-------------------------------------------------------------------------------
3273; Facharbeit-Teil  I²C-Bus - Byte des Akku ausgeben
3274;
3275; Ralf Wilke
3276;
3277
3278I2C_Byte_write:
3279       PUSH B
3280       MOV B, #8                   ; es sollen 8 Bits ausgegeben werden
3281Next_BitW:   RLC A                       ; Akku links rotieren, MSB ist im Carry
3282       LCALL wait_100us
3283       MOV SDA, C                  ; auf SDA-Leitung ausgeben
3284       LCALL wait_100us            ; Zeit CLCH = 1,3µs ist wegen 2 Zyklen
3285                                   ; Ausführungszeit gegeben
3286       SETB SCL_NS                       ; DXCX = 100ns sind durch Sub-Routine sicher
3287       LCALL WR_Status_Ausgang           ; gegeben
3288       CLR SCL_NS
3289       LCALL WR_Status_Ausgang
```

```
3290        DJNZ B, Next_BitW                ; das Ganze noch 7 Mal
3291        SETB SDA                    ; Es auf 1 setzen, um als Eingang für
3292                                    ; das Acknowledge Bit zu dienen
3293        SETB SCL_NS
3294        LCALL WR_Status_Ausgang
3295        MOV C, SDA                  ; Acknowledge Bit einlesen
3296        CLR SCL_NS                  ; Takt beenden
3297        LCALL WR_Status_Ausgang
3298        CLR SDA
3299;       JC Fehler_I2C_Write         ; Wenn kein Acknowledge kam, Fehler
3300        POP B
3301        RET
3302
3303
3304
3305;Fehler_I2C_Write:
3306;       RET
3307
3308
3309;-------------------------------------------------------------------------------
3310; Facharbeit-Teil  I²C-Bus - Byte in den Akku einlesen
3311;
3312; Ralf Wilke
3313;
3314
3315
3316I2C_Byte_read:
3317        CLR A                       ; Akku löschen
3318        SETB SDA                    ; SDA für Input auf 1 setzten
3319        PUSH B
3320        MOV B, #8
3321Next_BitR:   SETB SCL_NS                       ; Takt erzeugen
3322        LCALL WR_Status_Ausgang
3323        MOV C, SDA                  ; Bit ins Carry einlesen
3324        RLC A                       ; Und im Akku "verstauen"
3325        CLR SCL_NS                  ; Takt erzeugen
3326        LCALL WR_Status_Ausgang
3327        DJNZ B, Next_bitR           ; das Ganze noch 7 Mal
3328        MOV C, I2C_NoAck
3329        MOV SDA, C                  ; Acknowledge setzten, wenn gewünscht
3330        SETB SCL_NS                      ; Ack. heraustakten
3331        LCALL WR_Status_Ausgang
3332        CLR SCL_NS
3333        LCALL WR_Status_Ausgang
3334        POP B
3335        RET
3336
3337;-------------------------------------------------------------------------------
3338; Facharbeit-Teil  1 Byte aus EEPROM lesen
3339; Adresse im DPTR, Byte in I2C_R_Byte
3340;
3341; Ralf Wilke
3342;
3343
3344EEPROM_Byte_read:
3345        PUSH PSW
3346        ANL PSW, #11100111b         ; Registerbank 0 auswählen
3347        SETB DPTR_is_I2C            ; Festlegen, dass die Adresse im DPTR für
3348                                    ; die EEPROM-Auswahl herangezogen werden soll
3349        LCALL I2C_Dev_Sel                ; Das Device-Selection-Byte ermitteln
3350        ANL DEV_SEL, #11111110b          ; Das R'W-Bit auf schreiben setzten
3351        PUSH A                      ; Akku retten
3352;       PUSH PSW                    ; Carry, usw. retten
3353        LCALL I2C_Start             ; Start-Condition ausgeben
3354        MOV A, DEV_SEL
3355        LCALL I2C_Byte_Write        ; Byte ausgeben
3356        MOV A, DPL                  ; Das Lowbyte in den Akku
3357        LCALL I2C_Byte_Write        ; Byte ausgeben
3358        LCALL I2C_Stop              ; Stop-Condition, evtl. nicht nötig
3359        LCALL I2C_Start             ; reSTART-Condition s.o. !!
3360        ORL DEV_SEL, #00000001b          ; Das R'W-Bit auf lesen setzten
3361        MOV A, DEV_SEL              ; Device-Selection-Byte zum Ausgeben i.d. Akku
3362        LCALL I2C_Byte_Write        ; Byte ausgeben
3363        SETB I2C_NoAck              ; NoAck festlegen
3364        LCALL I2C_Byte_Read         ; Byte einlesen
3365        MOV I2C_R_Byte, A                ; Byte, dass gelesen wurde, in den Akku
3366        LCALL I2C_Stop              ; Stop-Condition ausgeben
3367;       POP PSW
3368        POP A
```

```
3369        CLR DPTR_is_I2C             ; DPTR nicht mehr als Adressierung
3370                                    ; der EEPROMs benutzen
3371        POP PSW
3372        RET
3373
3374;-------------------------------------------------------------------------------
3375; Facharbeit-Teil  1 Byte im EEPROM speichern
3376; Adresse im DPTR, Byte in I2C_W_Byte
3377;
3378; Ralf Wilke
3379;
3380
3381
3382EEPROM_Byte_write:
3383        PUSH PSW
3384        ANL PSW, #11100111b         ; Registerbank 0 auswählen
3385        SETB DPTR_is_I2C            ; Festlegen, dass die Adresse im DPTR für
3386                                    ; die EEPROM-Auswahl herangezogen werden soll
3387        LCALL I2C_Dev_Sel                ; Das Device-Selection-Byte ermitteln
3388        ANL DEV_SEL, #11111110b          ; Das R'W auf schreiben setzten
3389        PUSH A                      ; Akku retten
3390;       PUSH PSW                    ; Carry, usw. retten
3391        LCALL I2C_Start             ; Start-Condition ausgeben
3392        MOV A, DEV_SEL              ; Device-Selection-Byte zum Ausgeben i.d. Akku
3393        LCALL I2C_Byte_Write        ; Byte ausgeben
3394        MOV A, DPL                  ; Das Lowbyte in den Akku
3395        LCALL I2C_Byte_Write        ; Byte ausgeben
3396        MOV A, I2C_W_Byte                ; Byte, dass gesendet werden soll in den Akku
3397        LCALL I2C_Byte_Write        ; Byte ausgeben
3398        LCALL I2C_Stop              ; Stop-Condition ausgeben
3399        MOV wait_time, #10
3400        LCALL wait_ms
3401;       POP PSW
3402        POP A
3403        CLR DPTR_is_I2C             ; DPTR nicht mehr als Adressierung
3404                                    ; der EEPROMs benutzen
3405        POP PSW
3406        RET
3407
```

```
3408;Include Datei  DS1620-Bus
3409;-------------------------------------------------------------------------------
3410; Facharbeit-Teil   Protokol an den Temperatur-Sensor DS1620 senden
3411;
3412; Ralf Wilke
3413;
3414; Übergabe im Akku
3415
3416
3417Proto_DS1620:
3418       SETB DS_RST                      ; RST'-Leitung auf 1 setzen
3419       LCALL WR_Status_Ausgang          ; ausgeben
3420;      LCALL Wait_100us            ; warten. Im Datenblatt: TCC
3421       CLR DS_CLK                  ; Die Clock-Leitung auf 0 setzen
3422       LCALL WR_Status_Ausgang          ; ausgeben
3423;      LCALL wait_100us
3424       MOV R0, #7
3425Proto_DS_WDH:
3426       RRC A                       ; Den Akku nach rechts durch das Carry rotieren
3427                                   ; dadruch steht jetzt im Carry das Bit, das
3428                                   ; ausgegeben werden soll
3429       MOV DS_DQ, C                ; Carry in BIT DS_DQ schreiben und damit ausgeben
3430       LCALL Wait_100us            ; Im DB: TDC
3431       SETB DS_CLK                      ; Die Clock-Leitung auf 1 setzten
3432       LCALL WR_Status_Ausgang
3433;      LCALL wait_100us
3434       CLR DS_CLK                  ; Clock-Leitung auf 0 setzten
3435       LCALL WR_Status_Ausgang
3436;      LCALL wait_100us
3437
3438       DJNZ R0, Proto_DS_WDH       ; das Ganze noch 7 mal, dann sind Bits 0 bis 6
3439                                   ; draussen
3440
3441       RRC A                       ; Akku das letzte Mal rotieren
3442       MOV DS_DQ, C                ; Bit 7 ausgeben
3443;      LCALL Wait_100us
3444       SETB DS_CLK                      ; Die ClockLeitung auf 1 setzten. Das Bit 7
3445       LCALL WR_Status_Ausgang          ; ist damit übernommen. Hier setzt dann
3446                                   ; entweder eine Lese- oder Schreibroutine
3447                                   ; an und überträgt die eingentlichen Daten
3448       RET
3449
3450
3451;-------------------------------------------------------------------------------
3452; Facharbeit-Teil   Daten (8-Byte) an den Temperatur-Sensor DS1620 senden
3453;
3454; Ralf Wilke
3455;
3456; Übergabe im Akku
3457
3458
3459WR_DS1620:
3460       CLR DS_CLK                  ; Clock-Leitung auf 0, der Takt des 7.
3461       LCALL WR_Status_Ausgang          ; Protokoll-Bytes ist damit abgeschlossen
3462       MOV R0, #7
3463WR_DS1620_WDH:
3464       RRC A                       ; Akku rechtsrum durch das Carry rotieren
3465       MOV DS_DQ, C                ; ausgeben
3466       SETB DS_CLK                      ; Clock-Leitung auf 1
3467       LCALL WR_Status_Ausgang
3468       CLR DS_CLK                  ; Clock-Leitung auf 0
3469       LCALL WR_Status_Ausgang
3470       DJNZ R0, WR_DS1620_WDH      ; noch 6 Mal, bis Bit 0 bis 6 draussen sind
3471
3472       RRC A                       ; Das ursprüngliche Carry wieder in Carry holen
3473       MOV DS_DQ, C                ; und ausgeben
3474       SETB DS_CLK                      ; die Clock-Leitung zum letzten Mal High
3475       LCALL WR_Status_Ausgang
3476       CLR DS_RST                  ; die Reset-Leitung auf Low und damit ist
3477       LCALL WR_Status_Ausgang          ; der Transfer beendet
3478       MOV wait_time, #100         ; 100 ms warten, da das Chipinterne EEPROM
3479       LCALL wait_ms               ; so lange braucht, die Daten zu speichern
3480
3481       RET
3482
3483;-------------------------------------------------------------------------------
3484; Facharbeit-Teil   Daten (9-Byte) aus dem Temperatur-Sensor DS1620 lesen
3485;
```

```
; Ralf Wilke
;
; Übergabe im Akku und im Carry  C, ACC.7, ACC.6,..., ACC.0


RD_DS1620:   CLR A
       SETB DS_DQ                  ; DQ auf 1 setzten, um diesen Pin jetzt als
                                   ; Eingang nutzen zu können
       CLR DS_CLK                  ; Clock auf 0, der Takt des 7. Protokoll-
       LCALL WR_Status_Ausgang           ; Bytes ist damit abgeschlossen
       MOV R0, #8
RD_DS1620_WDH:
       MOV C, DS_DQ                ; Bit in das Carry einlesen
       RRC A                       ; Akku mit Carry rechtsrum rotieren, das Carry
                                   ; wird damit in den Akku geholt und ist nach
                                   ; 8 Rotationen an der richtigen Stelle
       SETB DS_CLK                       ; Clock auf 1 setzten
       LCALL WR_Status_Ausgang
       CLR DS_CLK                  ; CLock auf 0 setzten
       LCALL WR_Status_Ausgang
       DJNZ R0, RD_DS1620_WDH      ; noch 7 Mal, bis Bit 0 bis 7 drin sind

       MOV C, DS_DQ
       SETB DS_CLK
       LCALL WR_Status_Ausgang
       CLR DS_RST
       LCALL WR_Status_Ausgang
       RET
```

Visual Basic for Applications-Quelltext

```
1  Attribute VB_Name = "Modul1"
2  ' Dieses Makro funktioniert nur in Verbindung mit dem Mikrocontroller-Thermometer von
3  ' Ralf Wilke Facharbeit Werner-Jaeger-Gymnasium Nettetal Stufe 12 Informatik 2001
4  ' Es benutzt die DLL "RSAIP.DLL" von H.J. Berndt und B.Kainka von der CD zum Buch
5  ' Messen, Steuern und Regeln mit Word und Excel aus dem Franzis' Verlag 1997
6
7
8  Declare Sub OPENCOM Lib "RSAPI.DLL" (ByVal Parameter$)
9  Declare Sub CLOSECOM Lib "RSAPI.DLL" ()
10 Declare Sub TIMEOUT Lib "RSAPI.DLL" (ByVal ms%)
11 Declare Sub SENDBYTE Lib "RSAPI.DLL" (ByVal Zeichen%)
12 Declare Function READBYTE Lib "RSAPI.DLL" () As Integer
13
14 Public ByteZähler As Integer
15 Public Spalte2 As Integer
16 Public Rohdatenblatt As String
17 Public Temperaturdatenblatt As String
18
19 Sub WriteRowCaptions()
20 ' Diese Sub schreibt die Beschriftung für das Rohdatenblatt
21
22   Cells(1, 1).Value = "Datensatz Nummer"
23   Cells(2, 1).Value = "Adresse nach diesem Datensatz H"
24   Cells(3, 1).Value = "Adresse nach diesem Datensatz L"
25   Cells(4, 1).Value = "Start der Messung - Tag"
26   Cells(5, 1).Value = "Start der Messung - Monat"
27   Cells(4, 1).Value = "Start der Messung - Stunde"
28   Cells(5, 1).Value = "Start der Messung - Minute"
29   Cells(6, 1).Value = "Messintervall - Stunde"
30   Cells(7, 1).Value = "Messintervall - Minute"
31   Cells(8, 1).Value = "Messintervall - Sekunde"
32 End Sub
33 Sub DiagrammeLöschen()
34 ' Diese Sub löscht alle von diesem Makro erstellten Diagramme
35 Dim i As Integer
36   ThisWorkbook.Sheets(Temperaturdatenblatt$).Activate
37   i = 1
38   While (Cells(1, i).Value <> "")
39   'Solange noch Daten, aus denen das Diagramm besteht vorhanden sind
40     Sheets("Datensatz " + CStr(((i - 1) / 2) + 1)).Select
41   'Das betreffende Diagramm löschen
42     ActiveWindow.SelectedSheets.Delete
43     i = i + 2
44     ThisWorkbook.Sheets(Temperaturdatenblatt$).Activate
```

```
45   Wend
46   ' Die Tabellen löschen
47   Sheets(Rohdatenblatt$).Select
48   Selection.ClearContents
49   Sheets(Temperaturdatenblatt$).Select
50   Selection.ClearContents
51   Range("A1").Select
52 End Sub
53 Function NumberToRange(Number As Integer) As String
54 'Diese Function wandelt einen Integerwert zwischen 1 und 26 in den entsprechenden
55 'Buchstaben um. Sie ist als WorkAround gedacht, da des bei der Bezeichnung
56 'des Datenbereichs für ein Diagramm Probleme gab.
57   If (Number > 0) And (Number < 27) Then
58     NumberToRange = Chr(Number + Asc("A") - 1)
59   End If
60 End Function
61 Function SerRead() As Integer
62 'Diese Function inkrementiert die Public-Variable ByteZähler und ließt ein
63 'Byte über die ser. Schnittstelle ein.
64   ByteZähler = ByteZähler + 1
65   SerRead = READBYTE
66 End Function
67 Sub ReadoutData()
68 Attribute ReadoutData.VB_Description = "Makro am 18.08.2001 von Ralf aufgezeichnet"
69 Attribute ReadoutData.VB_ProcData.VB_Invoke_Func = " \n14"
70 'Diese Sub ließt den gesammten Speicher des MC-Thermometers ein und gibt ihn
71 'Datensatzweise in die Tabelle Roh-Daten aus.
72 Dim Zeichen As Integer
73 Dim Add_after_DS As Integer
74 Dim DatensatzZähler As Integer
75 Dim i As Integer
76
77     DatensatzZähler = 0
78     ByteZähler = 0
79     OPENCOM ("COM1,4800,N,8,1")
80     TIMEOUT 500
81     SENDBYTE (Asc("M"))
82     Zeichen = SerRead
83     If Zeichen = 255 Then ' Wenn das erste Byte FFh ist, ist der Speicher leer
84       MsgBox "Der Speicher leer, keine Daten zu Auswertung vorhanden", vbOKOnly,
   "Temperaturauswertung"
85     Else
86       While (Zeichen <> 255)
87         DatensatzZähler = Zeichen
88         Cells(1, DatensatzZähler + 1).Value = Zeichen
89         Zeichen = SerRead
90         Cells(2, DatensatzZähler + 1).Value = Zeichen
91         Add_after_DS = Zeichen * 256
92     'HighByte der nächsten Adresse nach diesem Datensatz mit 256 multiplizieren
93         Zeichen = SerRead
94         Cells(3, DatensatzZähler + 1).Value = Zeichen
95         Add_after_DS = Add_after_DS + Zeichen ' LowByte addieren
96         i = 4 ' Mit Zeile 4 beginnen
97         While (ByteZähler <> Add_after_DS)
98           Cells(i, DatensatzZähler + 1).Value = SerRead
99           If i = 10 Then ' Wenn Zeile 10 erreicht
100            i = i + 2
101        'eine Freizeile einfügen zur Trennung zwischen Verwaltungsdaten und Temperaturen
102          Else
103            i = i + 1
104          End If
105        Wend
106        Zeichen = SerRead
107      Wend
108    End If
109    CLOSECOM
110 End Sub
111
112 Function RawToTemp(Temp_H, Temp_L As Integer)
113 ' Diese Function wandelt die Roh-Temperaturdaten in Grad Celsius um
114 Dim Temp As Integer
115   If Temp_H <> 0 Then
116     Temp = 256 - Temp_L
117     RawToTemp = Temp / -2
118   Else
119     RawToTemp = Temp_L / 2
120   End If
121 End Function
122
```

```
123 Sub Main()
124 Dim Zeitpunkt As Date
125
126   Rohdatenblatt$ = "Roh-Daten"
127   Temperaturdatenblatt$ = "Temperaturen"
128   DiagrammeLöschen
129
130   ThisWorkbook.Sheets(Temperaturdatenblatt$).Activate
131   Sheets("Temperaturen").Select
132   Selection.ClearContents
133   Sheets("Roh-Daten").Select
134   Selection.ClearContents
135   Range("A1").Select
136   WriteRowCaptions
137   ' evtl. vorhandene Daten und Diagramme löschen
138   ReadoutData ' Daten einlesen
139   ' Jetzt Rohdaten umwandeln
140   Spalte = 2
141   Spalte2 = 1
142   Zeile = 12
143   While Cells(Zeile, Spalte).Value <> ""
144     Zeitpunkt = DateValue("1.1.1900 00:00") 'StartDatum
145     Zeitpunkt = DateAdd("d", Cells(4, Spalte).Value - 1, Zeitpunkt) ' Zeit aufaddieren
146     Zeitpunkt = DateAdd("m", Cells(5, Spalte).Value - 1, Zeitpunkt)
147     Zeitpunkt = DateAdd("h", Cells(6, Spalte).Value, Zeitpunkt)
148     Zeitpunkt = DateAdd("n", Cells(7, Spalte).Value, Zeitpunkt)
149     Intervall_H = Cells(8, Spalte).Value
150     Intervall_M = Cells(9, Spalte).Value
151     Intervall_S = Cells(10, Spalte).Value
152
153     'Anzeige-Format festlegen
154     ThisWorkbook.Sheets(Temperaturdatenblatt$).Activate
155     Columns(Spalte2).Select
156     Selection.NumberFormat = "dd/mm/ hh:mm:ss"
157     Columns(Spalte2 + 1).Select
158     Selection.NumberFormat = "0.0"
159     Range("A1").Select
160     ThisWorkbook.Sheets(Rohdatenblatt$).Activate
161
162     While Cells(Zeile, Spalte).Value <> ""
163       Temp_H% = Cells(Zeile, Spalte).Value
164       Zeile = Zeile + 1
165       Temp_L% = Cells(Zeile, Spalte).Value
166       Zeile = Zeile + 1
167       Temp = RawToTemp(Temp_H%, Temp_L%)
168       ThisWorkbook.Sheets(Temperaturdatenblatt$).Activate
169       'Zeit der Messung berechnen und ausgeben
170       Cells(Int(Zeile / 2) - 6, Spalte2).Value = Zeitpunkt
171
172       'Intervall aufaddieren
173       Zeitpunkt = DateAdd("h", Intervall_H, Zeitpunkt)
174       Zeitpunkt = DateAdd("n", Intervall_M, Zeitpunkt)
175       Zeitpunkt = DateAdd("s", Intervall_S, Zeitpunkt)
176       'Temperatur ausgeben
177       Cells(Int(Zeile / 2) - 6, Spalte2 + 1).Value = Temp
178       Columns(Spalte2).Select
179       'Selection.EntireColumn.AutoFit
180       Selection.ColumnWidth = 17
181       Columns(Spalte2 + 1).Select
182       Selection.ColumnWidth = 8
183       Range("A1").Select
184
185       ThisWorkbook.Sheets(Rohdatenblatt$).Activate
186     Wend ' Nächste Temperatur
187
188     'Diagramm erstellen
189     Lastzeile% = Int(Zeile / 2) - 6
190     Bereich$ = NumberToRange(Spalte2) + "1:" + NumberToRange(Spalte2 + 1) + CStr(Lastzeile)
191     ThisWorkbook.Sheets(Temperaturdatenblatt).Activate
192     'Variablen für später festlegen
193     Ursprung = Cells(1, Spalte2)
194     MaxX = Cells(Lastzeile, Spalte2)
195     ThisWorkbook.Sheets(Rohdatenblatt).Activate
196
197     Charts.Add
198     ActiveChart.ChartType = xlXYScatterLines
199     ActiveChart.SetSourceData Source:=Sheets("Temperaturen").Range(Bereich$), _
200         PlotBy:=xlColumns
201     ActiveChart.Location Where:=xlLocationAsNewSheet, Name:="Datensatz " + CStr((Spalte2 +1) /
```

```
   2)
202     With ActiveChart
203         .HasTitle = True
204         .ChartTitle.Characters.Text = "Temperatur"
205         .Axes(xlCategory, xlPrimary).HasTitle = True
206         .Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text = "Zeit"
207         .Axes(xlValue, xlPrimary).HasTitle = True
208         .Axes(xlValue, xlPrimary).AxisTitle.Characters.Text = "Temperatur in °C"
209     End With
210     With ActiveChart.Axes(xlCategory)
211         .HasMajorGridlines = True
212         .HasMinorGridlines = False
213     End With
214     With ActiveChart.Axes(xlValue)
215         .HasMajorGridlines = True
216         .HasMinorGridlines = False
217     End With
218     ActiveChart.Axes(xlCategory).Select
219     With ActiveChart.Axes(xlCategory)
220     'Grenzen festlegen
221         .MinimumScale = Ursprung
222         .MaximumScale = MaxX
223         '.MinorUnit = (MaxX-Ursprung
224         '.MayorUnit = (MaxX - Ursprung) / 5
225         .Crosses = xlCustom
226         .CrossesAt = Ursprung
227         .ReversePlotOrder = False
228         .ScaleType = xlLinear
229         .DisplayUnit = xlNone
230     End With
231     ActiveChart.HasLegend = False
232     ActiveChart.ApplyDataLabels Type:=xlDataLabelsShowNone, LegendKey:=False
233     ThisWorkbook.Sheets(Rohdatenblatt$).Activate
234     Spalte = Spalte + 1
235     Spalte2 = Spalte2 + 2
236     Zeile = 12
237   Wend 'Nächsten Datensatz
238
239 End Sub
240
```

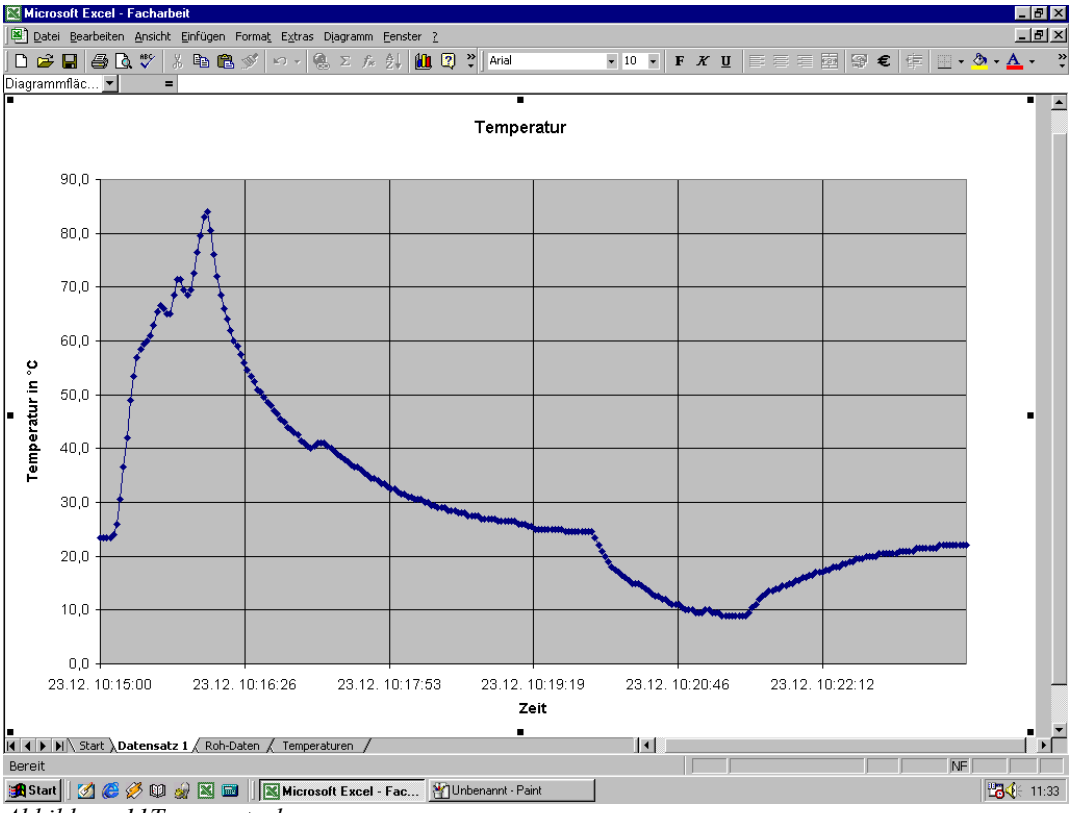

Abbildung 11Temperaturkurve

ASCII-Tabelle

Hex	ASCII	Hex	ASCII	Hex	ASCII	Hex	ASCII
0	NUL	20	SP	40	@	60	
1	SOH	21	!	41	A	61	a
2	STX	22	„	42	B	62	b
3	ETX	23	#	43	C	63	c
4	EOT	24	$	44	D	64	d
5	ENQ	25	%	45	E	65	e
6	ACK	26	&	46	F	66	f
7	BEL	27	'	47	G	67	g
8	BS	28	(	48	H	68	h
9	HT(TAB)	29	)	49	I	69	i
A	LF	2A	*	4A	J	6A	j
B	VT	2B	+	4B	K	6B	k
C	FF	2C	,	4C	L	6C	l
D	CR	2D	-	4D	M	6D	m
E	SO	2E	.	4E	N	6E	n
F	SI	2F	/	4F	O	6F	o
10	DLE	30	0	50	P	70	p
11	DC1 (X-ON)	31	1	51	Q	71	q
12	DC2 (TAPE)	32	2	52	R	72	r
13	DC3(X-OFF)	33	3	53	S	73	s
14	DC4 (/TAPE)	34	4	54	T	74	t
15	NAK	35	5	55	U	75	u
16	SYN	36	6	56	V	76	v
17	ETB	37	7	57	W	77	w
18	CAN	38	8	58	X	78	x
19	EM	39	9	59	Y	79	y
1A	SUB	3A	:	5A	Z	7A	z
1B	ESC	3B	;	5B	[	7B	{
1C	FS	3C	<	5C	\	7C	|
1D	GS	3D	=	5D	]	7D	}
1E	RS	3E	>	5E	^	7E	~
1F	US	3F	?	5F	_	7F	DEL(RUB OUT)

Literaturverzeichnis

1: Dietsche/Ohsmann, Mikrocontroller Handbuch, 1993 elektorAachen
2: KS0070B 16COM / 80SEG DRIVER & CONTROLLER FOR DOT MATRIX LCD
3: AT89 Series Hardware Description, ATMEL, www.atmel.com
4: H.J.Berndt/B.Kainka, Messen, Steuern und Regeln mit Word und Excel, 1997 Franzis'

Abbildungsverzeichnis