

Temperatursteuerung mittels  
CAN-Bus-Netzwerk  
Dokumentation

Stefan Bunge, Ramin Sayed-Moussavi, Kristian Müller  
Prozessdatenverarbeitung-Labor / Sensorik-Labor  
**TFH-Berlin**

Wintersemester 2004/2005

# Inhaltsverzeichnis

<b>1 Allgemein</b>	<b>3</b>
<b>2 Arbeitsumgebung</b>	<b>5</b>
2.1 Hardware . . . . .	5
2.1.1 Allgemeines . . . . .	5
2.1.2 Versuchsaufbau . . . . .	5
2.1.3 Der verwendete Microcontroller . . . . .	6
2.1.4 Beschreiben des Programmspeichers . . . . .	7
2.2 Software . . . . .	8
2.2.1 Allgemein . . . . .	8
2.2.2 Software-Entwicklungsumgebung . . . . .	8
2.2.3 Compiler . . . . .	9
2.2.4 Programmieren des PIC18F458 . . . . .	9
2.2.5 CANKing - analysieren am CAN-Busses . . . . .	10
<b>3 Entwicklung</b>	<b>11</b>
3.1 Allgemein . . . . .	11
3.2 Temperaturmessung . . . . .	12
3.2.1 Allgemein . . . . .	12
3.2.2 Messverstärker . . . . .	12
3.2.3 Analog- / Digital-Wandlung . . . . .	12
3.2.4 Entwicklungsaufbau . . . . .	13

<i>INHALTSVERZEICHNIS</i>	2
3.2.5 A/D-Wandler Routinen . . . . .	13
3.3 LCD Interface . . . . .	14
3.3.1 Allgemein . . . . .	14
3.3.2 Vorgehensweise . . . . .	14
3.3.3 Funktionsbeschreibungen . . . . .	15
3.3.4 Hardware . . . . .	16
3.4 Datenübertragung . . . . .	17
3.4.1 Allgemein . . . . .	17
3.4.2 Überlegungen zur Kabellänge . . . . .	17
3.4.3 Umsetzung . . . . .	18
3.4.4 Software . . . . .	18
3.4.5 CAN-Bus Konfigurationseinstellungen . . . . .	19
<b>4 Zusammenfassung</b>	<b>21</b>
4.1 Status . . . . .	21
4.2 Abschließend . . . . .	21

# Kapitel 1

## Allgemein

Dieses Projekt hat sich zum Ziel gestellt eine zentral gesteuerte Temperaturregelung für ein Haus mit mehreren Räumen zu realisieren. Dazu wird in jedem Raum eine Messstation installiert. Diese liefert dann über ein CAN-Bus-Netzwerk die Daten an den Server, der wiederum die Signale zum Regeln der Heizungsthermostate zurückschickt.

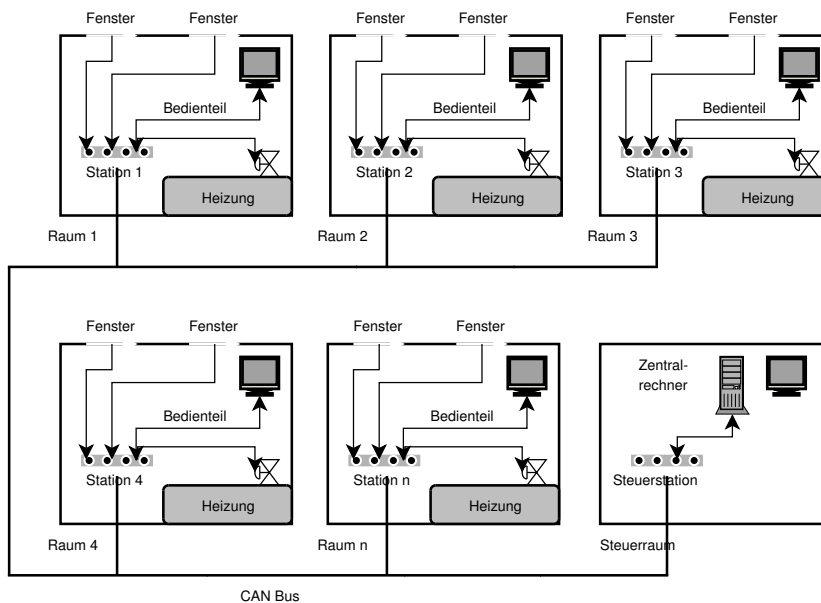


Abbildung 1.1: Ein Beispielaufbau mit 5 Räumen und dem Kontrollraum

In dieser Dokumentation wird auf unsere Arbeitsweise, die Realisierungsschritte und die verschiedenen Teilaspekte des Projektes eingegangen.

Das Projekt wurde innerhalb von 2 Semestern in den Lehrveranstaltungen Prozessdatenverarbeitung-Labor und Sensorik-Labor an der TFH-Berlin in Zusammenarbeit mit dem Fachbereich IV - Studiengang Technisches Gebäudemanagement durchgeführt. Wobei die Projektidee selbst von Prof. Dr. Mathias Fraaß und Herrn Dipl.-Ing. (FH) Albert Premer (TFH-Berlin, Technisches Gebäudemanagement) entwickelt wurde.

# Kapitel 2

## Arbeitsumgebung

### 2.1 Hardware

#### 2.1.1 Allgemeines

Um ein funktionierendes Testsystem aufzubauen, beschränkten wir uns auf ein Entwicklungssystem, deren Schaltplan später in ein serienreifes Layout umgewandelt werden kann. Die Bauelemente hierfür bekamen wir aus einem Evaluationsboard, vom ortsansässigen Elektronikhandel, aus dem Bauelementepool des PST-Labors sowie aus Musterbestellungen beim Hersteller.

#### 2.1.2 Versuchsaufbau

Der Aufbau wurde auf zwei Lochrasterplatinen durchgeführt. Damit können zwei Zimmer-Controller simuliert werden.

Zusätzlich wurde vom PST-Labor ein Microchip Entwicklungsboard "PIC-DEM CAN LIN II" zur Verfügung gestellt. Auf diesem Board befinden sich 2 PIC18F458, die vorläufig auch für den Versuchsaufbau auf den Lochrasterplatinen genutzt werden können. Im Laufe des Projektes bekamen wir von der Firma Microchip sowohl zwei PIC18F458 als auch zwei MPC2551<sup>1</sup> als Muster

---

<sup>1</sup>Die verwendeten CAN-Bus Pegelwandler

zur Verfügung gestellt. Zur Entwicklung wurde „CAN-NODE 1“<sup>2</sup> verwendet, der einen Potentiometer, neun LEDs und zwei Taster sowie einen CAN-Bus Port als Testperipherie bietet. Das Entwicklungsboard (im Folgenden auch Demoboard genannt) „PICDEM CAN LIN 2“ kann über eine RS232 Schnittstelle mit einem PC verbunden werden. Hierbei werden die Daten als 7 Bit zuzüglich einem Start- und einem Stopp-Bit mit 38400 Baud/s übertragen. Diese Schnittstelle wird hauptsächlich von der Software „CANKing“ benutzt.

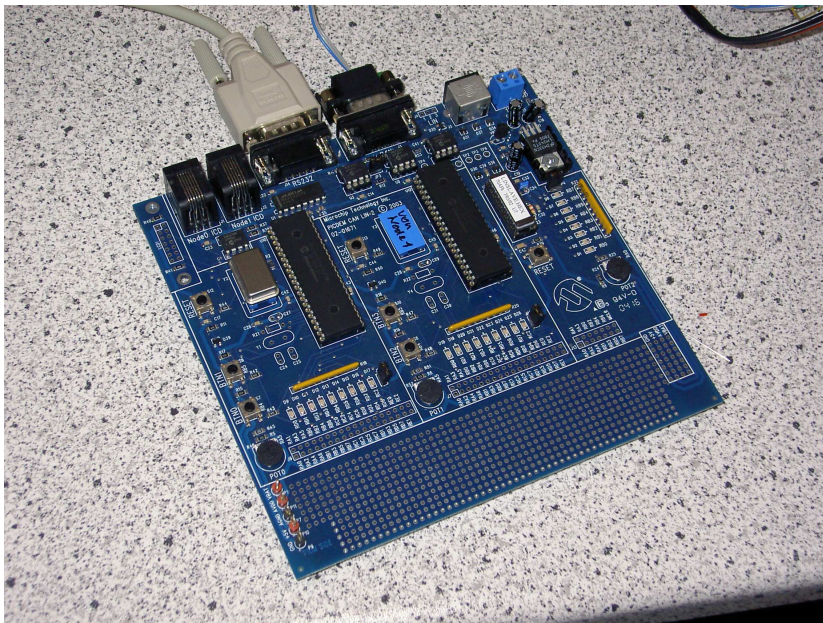


Abbildung 2.1: Das DemoBoard „PICDEM CAN LIN 2“

### 2.1.3 Der verwendete Microcontroller

Wir entschieden uns auf Grund des Preises und auf Grund der Produktfeatures einen Microchip PIC18F458 zu verwenden.

Die relativ neue PIC18 Serie verwendet eine für die Programmiersprache C optimierte Architektur, die auf der schon in älteren PIC Microcontrollern verwendeten RISC Architektur basiert. Die Firma Microchip selbst definiert:

---

<sup>2</sup>CAN-NODE 1 ist eines von 3 separaten Modulen auf dem „PICDEM CAN LIN II“ Entwicklungsboard

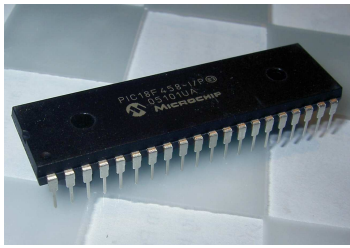


Abbildung 2.2: PIC18F458

„... combining RISC features with a modified Harvard dual bus architecture ...“.<sup>3</sup>

Wobei der PIC18 Core mit 10 MIPS<sup>4</sup> arbeiten könne. Der verwendete Controller kann mit bis zu 40 MHz getaktet werden. Wir nutzten folgende der speziellen Features, die der PIC18F458 zu bieten hat:

- Aktiver Bus-Node für CAN 2.0B (im Controller integriertes CAN-Modul)
- 10 Bit A/D Wandler
- 33 IO-Pins
- wiederbeschreibbarer Flash-Programmspeicher

#### 2.1.4 Beschreiben des Programmspeichers

Da wir keinen Microchip ICD (In Circuit Debugger) zur Verfügung hatten, wurde ein Nullkraftsockel zwischen dem PIC18F458 und dem Sockel des PIC-DEM 2 und später auf dem Testaufbau installiert. Damit ist es möglich den PIC mit Hilfe eines externen GALEP III<sup>5</sup> innerhalb kurzer Zeit zu programmieren und wieder einzusetzen. Damit der Mikrocontroller beim Herausnehmen nicht in Betrieb ist, wurde eine kleine Platine, die nur einen Schiebeschalter enthält, zwischen die Spannungsversorgung und einem, dem Board vorgeschalteten, Strommesser installiert.

<sup>3</sup>aus „8 Bit PIC Microcontrollers First Half 2005“ - 69630B.pdf

<sup>4</sup>MIPS - Abkürzung für Million Instructions per Second (englisch) - Millionen Instruktionen pro Sekunde (Wikipedia.de 22.06.2005)

<sup>5</sup>Die MPLAB IDE unterstützt das direkte Programmieren aus der IDE nur mit den externen Programmieradaptern PICSTART Plus; MPLAB PM 3; PRO MATE III; PICkit 1



Abbildung 2.3: Galep III

## 2.2 Software

### 2.2.1 Allgemein

Für die Softwareentwicklung verwendeten wir einen Rechner im PST-Labor, auf dem Windows XP installiert ist. Es handelte sich also um einen Intel-kompatiblen 32 Bit Rechner. Teilweise wurde kostenlose Software verwendet und teilweise kostenpflichtige Software. Auf die hier aufgelisteten Programme und Bibliotheken, wird im folgenden Abschnitt näher eingegangen.

Titel	Hersteller	Plattform	Beschreibung	Preis
MPLAB IDE	Microchip	Windows	Entwicklungsumgebung	kostenlos
C18	Microchip	Windows	Compiler	ca. 400 Euro
can18xx8	Microchip	PIC18xx8	Bibliothek (CAN-Modul)	kostenlos
CAN-King	Kvaser AB	Windows	CAN-Bus Analysator	kostenlos
EAGLE	CadSoft	GNU/Linux x86	Layout des Schaltplans	ab ca. 50 Euro
galep32	Conitec	Windows	Beschreiben des PIC18F458	kostenlos
Lyx/L <sup>A</sup> T <sub>E</sub> X	-	GNU/Linux PPC	Dokumentation	open source

Tabelle 2.1: verwendete Software und Programm-Bibliotheken

### 2.2.2 Software-Entwicklungsumgebung

Zum Erstellen der Software, die auf dem Controller laufen soll, wird die MPLAB IDE<sup>6</sup> verwendet. Die MPLAB IDE ist kostenlos von der Firma Microchip zu beziehen und läuft ausschließlich unter Microsoft Windows (32 Bit Version). Die IDE enthält einen Assembler für alle derzeit existierenden Mikrocontroller

<sup>6</sup>Integrated Development Environment (integrierte Entwicklungsumgebung)

der PIC-Serie. Zusätzlich zum syntaxhighlighting-fähigem Editor ist eine Projektverwaltung integriert, die sowohl das Anbinden an bestimmte Compiler als auch einen Debugger sowie Interfaces für bestimmte Programmiergeräte bietet.

### 2.2.3 Compiler

Um das Programm in C schreiben zu können, ist ein C-Compiler für den PIC18 Maschinencode nötig. Zu diesem Zweck wurde uns vom PST-Labor der Compiler MPLAB C18 von Microchip zur Verfügung gestellt. Der kostenpflichtige Compiler enthält neben vielen Standardfunktionen, wie verschiedenen Delay- und Timerfunktionen auch vorgefertigte Methoden zur Benutzung von Hitachi kompatiblen LCD-Displays, des A/D-Wandlers<sup>7</sup>, der seriellen Schnittstellen und eines externen CAN-Controllers. Leider sind diese Funktionen teilweise unvollständig oder fehlerhaft. So dass einige Schnittstellenfunktionen für bestimmte Controller-Module und externe Peripherie von uns selbst implementiert werden mussten. Außerdem ist in der Bibliothek kein Interface für die Benutzung des CAN-Modules im PIC18F458 enthalten. Hierfür wurde aus den Beispiel-Programm-Quellen eine zusätzliche CAN-Bibliothek verwendet.

Der MPALB C18 Compiler ist in die MPLAB IDE eingebettet und erzeugt beim erfolgreichen Aufruf der Makefunktion unter anderem eine Intel-hex Datei mit dem PIC18 Bytecode.

### 2.2.4 Programmieren des PIC18F458

Der PIC18F458 kann nach Erzeugen des Maschinencodes vom GALEP III aus programmiert werden.

Dazu wird die Intel-hex Datei in das Programm "galep32" geladen, sowie das zu programmierende Bauteil<sup>8</sup> ausgewählt. Vor jedem Programmier-

---

<sup>7</sup>in den von Microchip mitgelieferten Quellcodebeispielen ist erwähnt, dass diese Bibliotheksfunktionen fehlerhaft sind. Siehe Anlage "hw\_pic.c" Zeile 80 (Bemerkung: nachvollzogen, - KM 13.03.05)

<sup>8</sup>in unserem Fall ist das Bauteil unter MCU -> Microchip -> PIC18F458 -> DIL zu finden

vorgang sollte zunächst der Flash-Programmspeicher des PIC18F458 gelöscht werden, da der GALEP III nur die veränderten Speicherbereiche beschreibt. „galep32“ erkennt selbstständig, ob sich die Intel-hex Datei verändert hat und lädt diese nach erfolgreichem Kompilieren in seinen Datenpuffer. Außerdem kann ein automatischer Programmablauf mit Lösch-, Programmier- und Prüfungsvorgang definiert werden.

Im „galep32“-Programm sollte zusätzlich die Konfigurationseinstellung des Microcontrollers spezifiziert werden. Dazu gehörte in unserem Fall das Abschalten der Watchdog-Funktionen (um im Testaufbau sicher sein zu können, dass dieser nicht den Controller zum Neustarten anregt). Außerdem wird die Low-Voltage-Programming-Schnittstelle von uns momentan nicht unterstützt. Die Einstellungen des Frequenzeinganges sind auf einen 25 MHz Oszillator abzustimmen. Hier ist laut Datenblatt HS (High Speed) zu wählen.

### 2.2.5 CANKing - analysieren am CAN-Busses

Die Firma Microchip hat mit dem Demoboard PICDEM CAN LIN 2 auch die Software CANKing geliefert. CANKing ist eine von Kvaser AB entwickelte universell einsetzbare Software zum analysieren von Paketen am CAN-Bus. Um die am Demoboard verfügbare Peripherie vom PC aus anzusprechen, ist von Microchip ein vordefiniertes Template mitgeliefert worden. Mit diesem automatisch mit-installierten Template ist es möglich alle Pakete, die von den CAN-Nodes des PICDEM CAN LIN 2 auf den Bus gesendet werden selbst zu erzeugen und dabei das Packtformat zu analysieren. Ausserdem kann in der Packetliste nahezu alles was auf dem CAN-Bus gesendet wird, mitverfolgt werden. Angeschlossen wird der PC mit der CANKing Software über die serielle Schnittstelle<sup>9</sup>. Über diese Schnittstelle ist es mit den Templatefunktionen sogar möglich den Inhalt der CAN-Empfangs- und -Sende-Puffer in den Controllern einzusehen.

---

<sup>9</sup>Die Software CANKing bietet auch die Möglichkeit den parallelen Port als Interface zu benutzen.

# Kapitel 3

## Entwicklung

### 3.1 Allgemein

Im Folgenden wird auf die konkrete Lösung der einzelnen Teilprojekte eingegangen. Das umfasst die Temperaturmessung mit Signalwandlung, die Fenster-Sensoren, das Benutzer-Interface, sowie die Busanbindung.

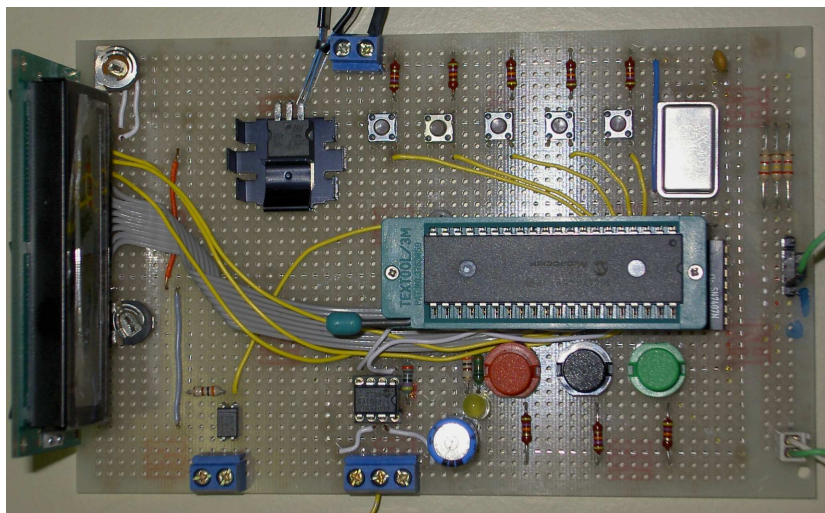


Abbildung 3.1: Testaufbau auf einer Euro Lochrasterplatte

## 3.2 Temperaturmessung

### 3.2.1 Allgemein

Um die Temperatur im Raum einzulesen muss die Wandlung eines analog vorliegenden Signals in einen digitalen Datenstrom erfolgen. Zum Einlesen des Wertes wird der Messsensor „W 3d - 81-210“<sup>1</sup> verwendet. Das analoge Signal wird dann mit dem A/D-Wandler des Controllers aufgenommen und danach ausgewertet.

### 3.2.2 Messverstärker

Da das Signal des Messfühlers nur als Widerstanswert vorliegt<sup>2</sup>, muss der gemessene Wert in eine Spannung umgewandelt werden. Hierzu wird ein Spannungsteiler mit nachgeschaltetem Operationsverstärker verwendet. Da der Sensor über eine längere Anschlussleitung verfügen kann, wird er mit einer Kapazität von  $> 10 \text{ nF}$  gegen induktiv erzeugte Spannungsspitzen geschützt<sup>3</sup>. Der „W3d - 81-210“ ist im Grunde genommen ein Widerstand, der seinen Wert entsprechend der Temperatur annähernd linear ändert. Da in dieser Anwendung nur der Temperaturbereich von  $0 \text{ }^\circ\text{C}$  bis  $50 \text{ }^\circ\text{C}$  interessiert, kann man den Wert innerhalb dieser Grenzen zu einer Spannung von  $0 \text{ V}$  bis  $5 \text{ V}$  wandeln.

### 3.2.3 Analog- / Digital-Wandlung

Das Messsignal ist an Pin AN0 angelegt. Der PIC18F458 hat die Möglichkeit diesen Wert mittels eines 10-Bit A/D-Wandlers zu konvertieren. Dafür benutzt er das Sägezahnverfahren (Successive Approximation Register - Verfahren). Zum Einlesen der aktuellen Temperatur wird der A/D-Wandler einmal initialisiert. Da die Taktfrequenz (Fosz) des Controllers  $25 \text{ MHz}$  beträgt, muss ein Vorteiler für den A/D-Wandler gesetzt werden. Mit einem Teilerwert von 32

---

<sup>1</sup>Anhang: Datenblatt „Philips KTY81-2 series“ und Datenblatt „Siemens KT 210“

<sup>2</sup>Eine von Albert Premer aufgenommene Messreihe zu diesem Sensor befindet sich im Anhang „W 3d - 81-210 - Messreihe“

<sup>3</sup>siehe Anlage: Siemens Datenblatt „KT210“ Seite 1 Fußnote 1

ergibt sich ein Messintervall von  $\frac{12 \cdot 32}{25 \text{ MHz}} = 15,36 \mu\text{s}$  was einer maximal möglichen Messfrequenz von 66 kHz entspricht.

### 3.2.4 Entwicklungsaufbau

Zur Durchführung der Versuche am A/D-Wandler wurde das Potentiometer des NODE 1 des "PICDEM LIN CAN 2" benutzt. Zu beachten war hier, dass der PIC18F458 auf dem Board mit 25 MHz getaktet ist<sup>4</sup> und das Potentiometer an Pin AN4 angeschlossen ist. Daher wird auch hier der Vorteiler 32 gewählt.

### 3.2.5 A/D-Wandler Routinen

Um den A/D-Wandler zu initialisieren kann, wie bereits erwähnt, nicht die entsprechende Bibliotheksfunktion des C18-Compilers verwendet werden. Deshalb wurden zwei kleinere Funktionen erstellt, die die grundlegenden Funktionen des A/D-Wandlers nutzbar machen.

#### 3.2.5.1 initADC()

Diese Funktion initialisiert den A/D-Wandler. Hierzu wird die Frequenz eingestellt sowie der Modus und der Eingabe-Pin ausgewählt.

#### 3.2.5.2 unsigned short readADC()

Mit dieser Funktion wird ein Wandlungszyklus angestoßen. Nach erfolgreicher A/D-Wandlung wird der Wert in einem 16-Bit-Integer zurückgeliefert. Da der Wandler nur 10- der 16 Bit ausnutzt sind die 6 höchstwertigsten Bits immer 0. Die gewählte Darstellung des Rückgabewertes ist Big Endian.

---

<sup>4</sup>In den ersten Testaufbauten unterschieden sich die Frequenz des PICDEM CAN LIN 2 und unseres Testboardes.

## 3.3 LCD Interface

### 3.3.1 Allgemein

Dem Benutzer sollen auf einem 2-Zeilen-LCD-Display Statusinformationen, wie die aktuelle Temperatur, die Solltemperatur und Bedienungshilfen, angezeigt werden. Hierfür wurde im Versuchsaufbau ein Hitachi-kompatibles Display benutzt.

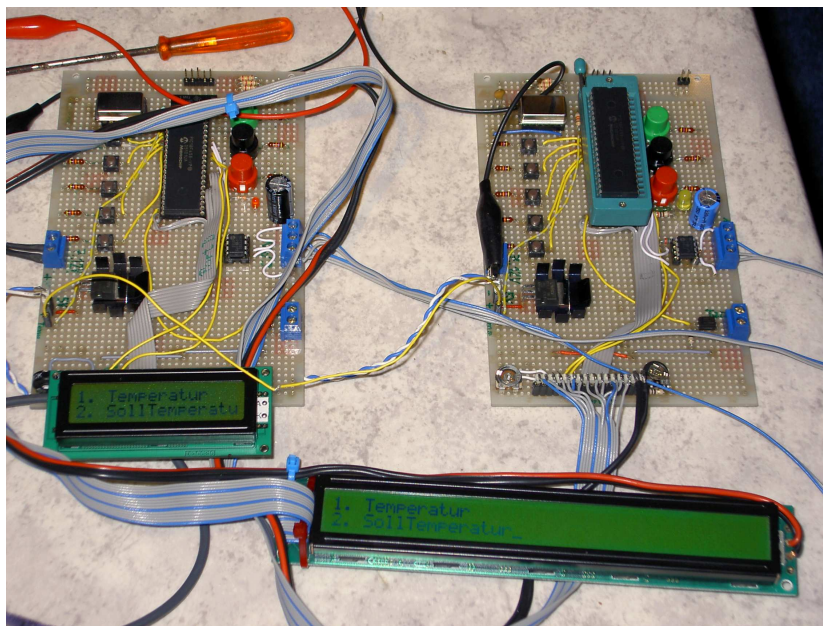


Abbildung 3.2: beide Testboards mit ihren LC-Displays

### 3.3.2 Vorgehensweise

Da scheinbar die Bibliotheksfunktionen für die Displayansteuerung nicht mit dem verwendeten Display kompatibel sind (die Bibliotheksfunktion „OpenXLCD()“<sup>5</sup> erzeugte im Versuch eine Endlosschleife) musste auch hier eine eigene Lösung entwickelt werden.

<sup>5</sup>Anleitung: MPLAB C18 C Compiler Libraries (DS51297C) Seite 67

### 3.3.3 Funktionsbeschreibungen

Im Folgenden sind die wichtigsten Funktionen zur Ansteuerung des Displays aufgeführt und näher erläutert.

#### 3.3.3.1 `initLCD()`

Im Testaufbau ist `initLCD()` für das direkte Aufrufen von 3 Initialisierungskommandos verantwortlich. Wobei das Display auf zwei Zeilen mit 5x7 Pixel großen Zeichen aus einem 6-Bit großen Zeichensatz initialisiert wird. Im eingestellten Modus wird bei fortlaufender Ausgabe automatisch nach dem 40ten Zeichen in die zweite Zeile umgebrochen. Da wir im Verlauf der Entwicklung, zu Testzwecken, auch auf ein 2x40 Zeichen großes Display wechseln konnten, war diese Ausgabeart optimal.

#### 3.3.3.2 `clearLCD() / homeLCD()`

Diese beiden Funktionen setzen den Cursor auf das erste Zeichen der ersten Spalte. Wie der Name schon sagt, löscht `clearLCD()` dabei auch den Inhalt des Displayspeichers.

#### 3.3.3.3 `putLCDData(unsigned char iCmd, unsigned char iRS)`

Die Funktion hat die Aufgabe Daten an den Displaycontroller zu senden. Hierbei bestimmt `iCmd` das Datenbyte und das niederwertigste Bit von `iRS`<sup>6</sup> ob es sich um ein Datum oder um ein Kommando handelt.

#### 3.3.3.4 `writeLCDChar(unsigned char iChar)`

Die Funktion gibt das Zeichen „`iChar`“ an der aktuellen Cursorposition aus und bewegt den Cursor anschließend ein Zeichen weiter.

---

<sup>6</sup>`iRS` bestimmt hierbei direkt den Wert des „RS“-Eingangs des Displaycontrollers.

### 3.3.3.5 writeLCDString(unsigned char \*iChar)

Diese Funktion gibt einen Null-terminierten String an der Cursorposition aus. Sie benutzt dazu writeLCDChar().

### 3.3.3.6 writeLCDcharbin(unsigned char iChar)

Diese Funktion gibt ein Byte in binärer Form auf dem Display aus, was zum debuggen - z.B. zum Anzeigen von Registerinhalten - sehr hilfreich sein kann.

### 3.3.3.7 moveXY(x, y) / moveLDCursorLeft() / moveLDCursorRight()

Mit diesen Funktionen kann der Cursor auf dem Display platziert werden.

## 3.3.4 Hardware

Das LC-Display ist an „PortB“ und an „PortD“ des PIC18F458 angeschlossen. Wobei an Pin0 und Pin1 von Port B die Kommandoleitungen „Enable“ und „RS“ angeschlossen sind und an den 8 Leitungen von Port D die Datenleitungen anliegen. Das ebenfalls benötigte Signal „R/W“ wurde von uns direkt mit Masse verbunden, da wir nur Schreibzugriffe auf das Display ausführen und diese Zugriffsart durch ein, an den „R/W“-Eingang angelegtes, Low-Signal ausgewählt wird.

Um den Kontrast einstellen zu können sind die entsprechenden Pins des Displays gemäß Datenblatt<sup>7</sup> an ein Trimpotentiometer auf dem Entwicklungsbord angeschlossen. Die LED-Hintergrundbeleuchtung wurde über einen 0 - 100 Ohm Trimpotentiometer direkt an Vss angeschlossen, da auf dem Displaycontroller bereits ein 100 Ohm Widerstand installiert ist, der den Strom der LED begrenzt.

---

<sup>7</sup>Kontrast siehe Anlage: Datenblatt „L1682“ Seite 3 „Standart STN & 12022“

## 3.4 Datenübertragung

### 3.4.1 Allgemein

Die Datenübertragung zwischen dem Kontrollnode und den Messnodes erfolgt über den CAN-Bus (Controller Area Network). Hierzu werden Pegelwandler zwischen den Signalpins des PIC-Controllers und der Kabelstrecke benötigt. Wie auch auf dem Demoboard „PICDEM CAN-LIN 2“ wird in unserem Testaufbau der MPC2551 benutzt.

### 3.4.2 Überlegungen zur Kabellänge

Wie im CAN-Standard <sup>8</sup> definiert ist, nimmt bei länger werdenden Kabellängen die maximal mögliche Datenrate ab. Die Buskabel sollten ein verdrehtes Kupferkabel mit zusätzlicher Ummantelung besitzen. Der Wellenwiderstand sollte  $120\Omega$  betragen. Bei einem Kabelquerschnitt von  $0,8\text{ mm}^2$  kann mit einem solchen Kabel eine maximale Geschwindigkeit von ca. 50 kBit/s bei einer maximalen Buslänge von 1 km erzielt werden. Weitere Angaben eines einzelnen Anbieters für CAN-Bus-Kabel können der Anlage „Unitronic BUS CAN UL/CSA“ entnommen werden. Die von uns benutzten 128 kBit/s erlauben eine Buslänge von ca. 500m. Sollte in der späteren Anwendung mehr als 1 km Kabellänge nötig sein muss der unten genannte CAN\_BAUDRATE\_PRESCALE auf 0x0F gestellt werden, woraus eine Busgeschwindigkeit von 31,25 kBit/s resultieren würde. Für das Physikalische Interface zum Bus wird von uns aus Kostengründen eine 3-Polige Industriebuchse gewählt. Wobei Interfacekabel zum CAN-Bus meist 9 Polige D-Sub besitzen. Um die Testboards mit dem Demoboard verbinden zu können wurde von uns ein D-Sub Interfacestecker erstellt. Da der verwendete Pegelwandler zwischen den CAN-H und CAN-L Leitungen eine Maximale Spannung von 5 V erzeugt muss vor dem produktiven Einsatz noch geprüft werden, ob die maximale Buslänge wirklich mit diesen Signalpegeln zu erreichen ist.

---

<sup>8</sup>CAN ist in der ISO 11898 international genormt.

### 3.4.3 Umsetzung

Die ersten Tests auf dem PICDEM CAN-LIN 2 Entwicklungsborad erfolgten noch ohne Datenkabel. Auf dem PICDEM - Tesboard sind die beiden Testnodes (nach dem MPC2551) direkt miteinander verbunden<sup>9</sup>. Zu den weiteren Tests wird der MPC2551 auf die Lochplatine umgesockelt und das CAN-Kabel an den entsprechenden Klemmen befestigt. Nach dem Erhalt der Muster-Chips konnten sowohl das PICDEM CAN LIN 2 als auch unsere Testboard mit den nötigen MCP2551 ausgestattet werden und so auch miteinander verbunden werden. Hier war besonders auf gleiche Busgeschwindigkeit zu achten, weswegen wir uns an diesem Punkt der Entwicklung entschieden haben auch die beiden Testboards mit einem 25 MHz Oszillator auszustatten.

### 3.4.4 Software

Der PIC18F458 enthält ein als aktives CAN-Bus-Knoten verwendbares CAN-Modul. Allerdings ist zur Nutzung dieses Modules das Ansprechen der entsprechenden Register nötig. Die aktuelle, von uns genutzte Version der MPLAB C18 C-Bibliothek, enthält zwar umfangreiche CAN-Interface-Funktionen, diese sind aber für den externen CAN-Controller MPC2510 (Firma Microchip) gedacht. Das heißt, dass wir keine dieser mitgelieferten Bibliotheks-Funktionen für unser Projekt nutzen konnten.

Zum PICDEM CAN LIN 2 wurde allerdings der Beispielcode der CAN-Nodes mitgeliefert. So konnten wir in vielen Versuchen nach und nach die CAN-Funktionen der PICDEM CAN LIN 2 - Nodes für uns nutzbar machen. Leider ist es nicht gelungen mit diesen Funktionen das Interface zum CAN-Bus zu aktivieren. Während die Funktionen Softwareseitig zu funktionieren schienen ist am Bus kein Signal erzeugt worden. Also suchten wir in den Dokumentationen zur PIC18Fxx8 - Serie nach genauen Beschreibungen des Modul-Interfaces. Wie sich herausstellte mussten einerseits die Ausgänge zum CAN-

---

<sup>9</sup>vergleiche Anlage „PICDEM CAN-LIN 2 User's Guide - Board Schematic Node2 MC, LIN and CAN Transceivers“ (DS51334A) Seite 50

Pegelwandler (CAN-RX & CAN-TX) entsprechend ihrer Funktion als Eingang bzw. Ausgang konfiguriert werden andererseits müssen komplexe Einstellungen für die verschiedenen Modi des Busses getroffen werden. Zusätzliche Schwierigkeiten bei Erforschen der (für Leihen der CAN-spezifischen Akürzungen) relativ schwer zu lesenden Programmquellen begründeten sich im komplexen CAN-King-Protokoll, das zwar als logische Schicht über dem eigentlichen CAN-Bus-Interface liegt, aber in den Quellen zwischen die Zugriffsfunktionen eingeflochten ist.

Nach vielen erfolglosen Test, fanden wir in den zum PIC DEM CAN LIN 2 mitgelieferten Daten, eine Bibliothek für das CAN-Modul der PIC18Fxx8 Baureihe. Diese Bibliothek besteht aus einer einzelnen .c und einer .h -Datei und befindet sich im Verzeichnis „AppNotes/an738/“ auf der mitgelieferten CD. Die zugehörige Dokumentation ist in der Datei 00738b.pdf im selben Verzeichnis zu finden. Interessanter Weise ist diese, relativ übersichtliche und gut dokumentierte Bibliothek, nicht vom CAN-Node des Demoboards verwendet worden.

Während der Umstellung unseres Codes auf die neuen Interface Funktionen ist uns auch aufgefallen, dass die Port-Pins von uns falsch initialisiert wurden. Die eigentlich für den Pin RB2 und RB3 gedachten Ein-/Ausgabe Modi wurden versehentlich auf Pin RB 1 und RB2 angewendet. Ursache für diesen Fehler war, dass beim Erstellen der Binärmaske für das Ein-/Ausgabe-Konfigurations-Register von Port B „TRISB“ nicht mit Null bei Bit 0 zu Zählen begonnen wurde sondern mit „1“. Nach der Korrektur dieses Fehlers konnten erstmals Bussignale Signale hinter dem CAN-Pegelwandler gemessen werden.

### 3.4.5 CAN-Bus Konfigurationseinstellungen

In der Funktion CANInitialize() kann der Modus des Betriebes des CAN-Modules gewählt werden. Da wir in unserer Entwicklung kompatibel mit dem Bus des Demoboards PICDEM CAN LIN 2 sein wollten setzen wir hier den gleichen Modus wie auch auf den CAN-Nodes des Demoboards.

Die genutzten Einstellungen sind in folgender Tabelle ersichtlich.

Einstellung	Abkürzung	Wert	resultierende Einstellung
CAN_SYNC_JUMP_WIDTH	SJW	0x02	$3 * T_{Quarz}$
CAN_SAMPLE_BUS_LINE	SAM	0x01	dreimaliges Samplen des Busses
CAN_PHASE_SEGMENT_1	PHSEG1	0x07	$8 * T_{Quarz}$
CAN_PHASE_SEGMENT_2	PHSEG2	0x07	$8 * T_{Quarz}$
CAN_PROPAGATION_TIME_SELECT	PRSEG	0x07	$8 * T_{Quarz}$
CAN_BAUDRATE_PRESCALE	BRP	0x02	mit anderen E. $128 \frac{kBit}{s}$

Tabelle 3.1: Einstellung des CAN-Bus-Modes

Der erhoffte Vorteil einer Anpassung des Busses der Testboards mit dem Demoboard ist, dass über das Demoboard, das mit dem PC verbunden werden kann, die CAN-Bus-Pakete am PC mitverfolgt werden können. Dadurch wäre es auch möglich, eine eigene Anwendung zu entwickeln, die die an den PC (über den RS232-Bus) gesendeten Daten auswertet. Diese Anwendung könnte dann über das selbe Interface auch Daten vom PC an den CAN-Bus senden. Denkbar wäre auch die Anbindung über einen USB-Seriell-Adapter an eine beliebige PC-Plattform.

# Kapitel 4

## Zusammenfassung

### 4.1 Status

Am 25.07.2005 sind beide Testboards in Betrieb und die Temperaturmessung erfolgt periodisch. Es existiert eine intuitive Benutzerführung mit Hilfe eines Menüs auf dem LC-Display. Der Schaltplan des funktionierenden Boards ist in Eagle erstellt und kann zu einer produktionsreifen Platine geroutet werden. Das Interface zum CAN-Bus ist Softwaretechnisch umgesetzt so, dass Befehle am Testboard empfangen werden können. Ausserdem können Informationen über die Raumtemperatur und den Status der Fenstersensoren abgerufen werden.

### 4.2 Abschließend

Das Projekt war sehr umfangreich, und durch viele unvorhergesehene Probleme sowie zeitliche Engpässe bei den Entwicklern, musste es um ein Semester verlängert werden. Dennoch war es seinen Aufwand wert und es konnten viel Erfahrung gesammelt werden.

Eine derartig im Haus verteilte und doch zentral gesteuerte Temperaturregelung hat viele Vorteile, die es in weiteren Projekten heraus zu arbeiten gilt.

# Abbildungsverzeichnis

1.1	Ein Beispielaufbau mit 5 Räumen und dem Kontrollraum . . . .	3
2.1	Das DemoBoard „PICDEM CAN LIN 2“ . . . . .	6
2.2	PIC18F458 . . . . .	7
2.3	Galep III . . . . .	8
3.1	Testaufbau auf einer Euro Lochrasterplatine . . . . .	11
3.2	beide Testboards mit ihren LC-Displays . . . . .	14

# Tabellenverzeichnis

2.1	verwendete Software und Programm-Bibliotheken . . . . .	8
3.1	Einstellung des CAN-Bus-Modes . . . . .	20