Projektdokumentation

Fabian Retkowski, Pascal Maximilian Bremer

23. Juni 2016

Inhaltsverzeichnis

1 Einleitung

Mit dem Begriff der systemnahen Programmierung wird hauptsächlich die Programmierung mit Assembler verstanden. Assembler ist hierbei ein Programm, welches einen Source-Code direkt in Maschinenbefehle übersetzt. Maschinenbefehle sind Bitmuster, die von einer CPU verstanden und ausgeführt werden können. Assembler ist nur auf denjeweiligen CPUs ausführbar, daher ist in den Befehlen auch immer die physikalische Architektur der CPU wiederzuerkennen. Um das System zu verstehen, ergibt es also Sinn, sich mit Assembler zu beschäftigen – genau dieses bedeutet es nämlich, das System voll und ganz zu verstehen.

2 Grundlagen

2.1 Intel 8051

Intel 8051 ist ein 8-Bit Mikroprozessor, welcher im in den Jahren 1980 bis 1990 hergestellt wurde. Er gehörte zur Familie der MCS-51-Prozessoren. Die Taktfrequenz entsprach 6 Megahertz. Er enthält 4 Registerbänke mit je 8 Registern , ihm stehen 4 Ports zur Verfügung.

2.2 Assembler

Assembler, oft synonym mit Assemblersprache, ist eine systemnahe Programmiersprache, die für es verschiedene Rechner gibt, jeweils mit einem eigenen Befehlssatz für die CPU. Von der Maschinensprachen unterscheidet sie sich insofern, dass statt dem Binärcode die Befehle und Operanden durch mnemonische Symbole (z. B. „JPE“), Operanden als Adressen, dargestellt werden.

2.3 Entwicklungsumgebung

Entwickelt wurde mit der IDE MCU 8051, welche auf den Prozessor Intel 8051 spezialisiert ist. Die Entwicklungsumgebung unterstützt 2 Programmiersprachen. Zum einen Assembler, zum anderen C. Er erhält auch zahlreiche Simulatoren für Hardwarekomponenten wie LEDs, Displays, einen Temperatursensor, Taster und viele weitere.

3 Konzept

3.1 Idee

Das Ziel dieser Arbeit ist es einen Mikrokontroller für einen Wasserkocher zu programmieren. Der Clou hierbei ist, dass die gewünschte Temperatur vom Benutzer

festgelegt werden kann. Zudem werden wird mithilfe eines Temperatursensors die aktuelle Temperatur ausgelesen und dem Benutzer angezeigt. Der Wasserkocher kann zu jeder Zeit ausgeschaltet werden und er überprüft vor dem Starten, ob genug Wasser vorhanden ist. Eine Hupe signalisiert das Erreichen der gewünschten Temperatur.

3.2 Funktionsweise

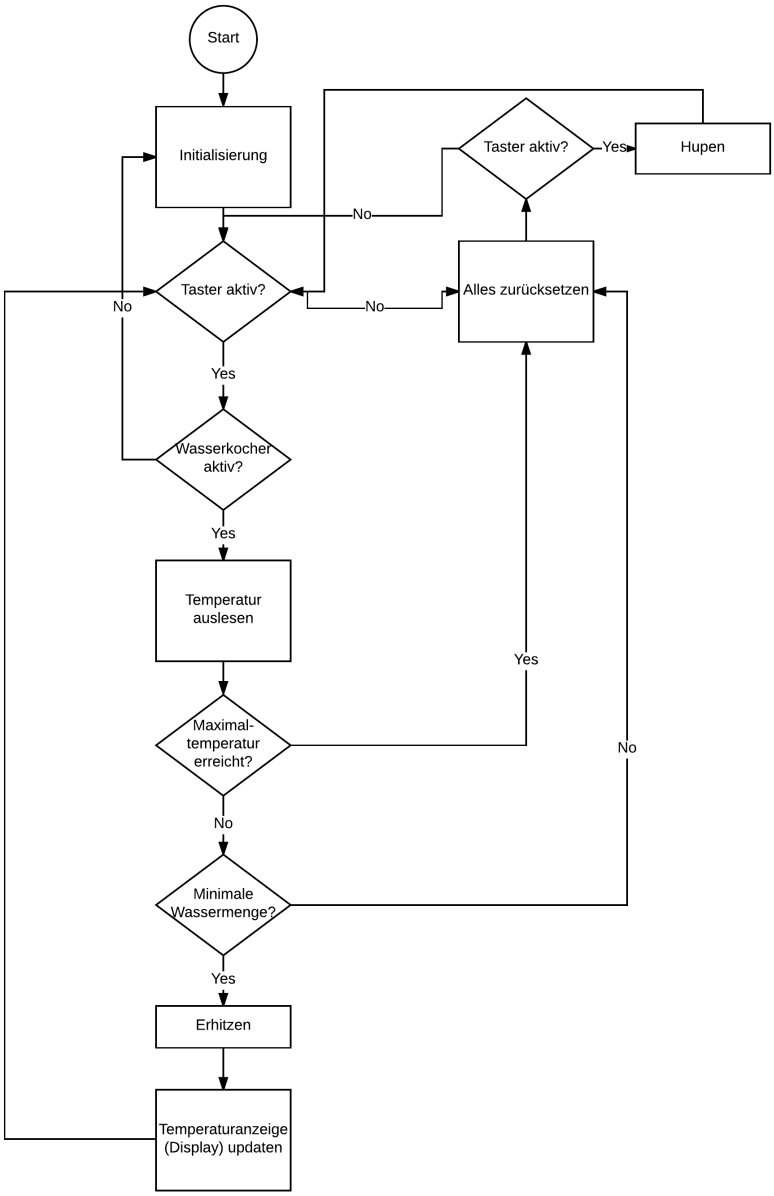

Abbildung 1: Flussdiagramm

3.3 Aufbau

Abbildung 2: Aufbau der simulierten Hardware

4 Implementierung

Aufgrund fehlender Hardware wird der Ablauf mithilfe simulierter Hardwarekomponenten simuliert. In diesem Kapitel werden nun die verwendeten Komponenten vorgestellt.

4.1 Hupe

Die Hupe signalisiert, dass der Wasserkocher selbstständig von dem aktiven in den inaktiven Status wechselte. Um eine langanhaltende Hupe zu signalisieren wird das positive Signal für die Hupe mithilfe einer Schleife länger erhalten.
Die Hupe wird nicht aktiviert, wenn der Benutzer bewusst den Wasserkocher ausschaltet.

4.2 LED-Panel

Das LED Panel gibt eine Übersicht über den internen Status des Wasserkochers. Aufgrund fehlender Hardware werden Komponenten, wie Heizstab und Hupe mit LED Lichtern simuliert. Das Panel ist wie folgt gegliedert: 1. Zeigt an, ob das Gerät aktiv ist

2. Zeigt an, ob die Hupe aktiv ist
3. Zeigt an, ob der Heizstab aktiv ist
Bei der Anzeige ist zu beachten, das die LED bei positivem Zustand aus- und bei negativem Zustand angeschaltet ist.

4.3 Display

Das Display gibt die ausgelesenen Daten des Temperatursensors aus. Aufgrund fehlender Hardware kann keine Aussage getroffen werden, ob die derzeitige Refreshrate ausreichend ist, oder ob die Anzeige für kurze Zeit deaktiviert ist.

4.4 Taster

Der Taster dient dem An- und Ausschalten des Wasserkochers. Ist er geschlossen, so wechselt der Wasserkocher in den aktiven Modus. Es ist jedoch nicht möglich ihn softwareseitig zurückzusetzen. Hier wird in der Produktion eine mechanische Komponente (evtl. Feder) benötigt.

4.5 Temperatursensor

In diesem Projekt wird der Temperatursensor DS1620 aus dem Hause "maxim integrated" simuliert. Die Besonderheit an dieser Hardwarekomponente ist die Tatsache, dass sie autonom zu dem 8051 Mikrokontroller agiert. Mithilfe eines Kommunikationskanal können Informationen zwischen den Mikrokontroller und des Temperatursensors ausgetauscht werden.

4.5.1 Allgemeine Informationen

Der Temperatursensor kann Temperaturen von $-55°C$ bis $125°C$ messen. Die Temperaturwerte werden in einem 9-Bit Datentyp gespeichert. Das LSB steht hierbei für $0.5°C$ und das MSB subtrahiert 128 von der Zahl.

Aufgrund dieser Besonderheit wird das LSB und MSB von unserem Projekt ignoriert, da Gleitkomma- und Minuszahlen die Anwendung nur unnötig komplizieren. Dies bedeutet, das uns nur lediglich 7-Bit zum speichern der Temperatur zur Verfügung stehen. Die Messspanne von 0-127(1111111) ist aber für unser Projekt vollkommen ausreichend.

Zudem ist es möglich im Sensor eine minimale und maximale Temperaturgrenze festzulegen. Wird diese überschritten, so wird der jeweilige Ausgang auf 1 gesetzt. Siehe Diagramm 3.

4.5.2 Aufbau

Der Sensor ist recht übersichtlich aufgebaut. Er besitzt, wie in Grafik 4.5.2 zu sehen ist 3 Eingänge um mit ihm zu interagieren. RST Eingang dient zum Zurücksetzen des Kommunikationskanal und wird bei einer steigenden Signalflanke aktiviert. Eine positive Signalflanke am Eingang CLK signalisiert dem Sensor, dass das Signal, welches an DQ anliegt eingelesen bzw. ausgegeben werden soll, nähere Informationen hierzu folgen in Kapitel 4.5.3.

DS1620 FUNCTIONAL BLOCK DIAGRAM Figure 1

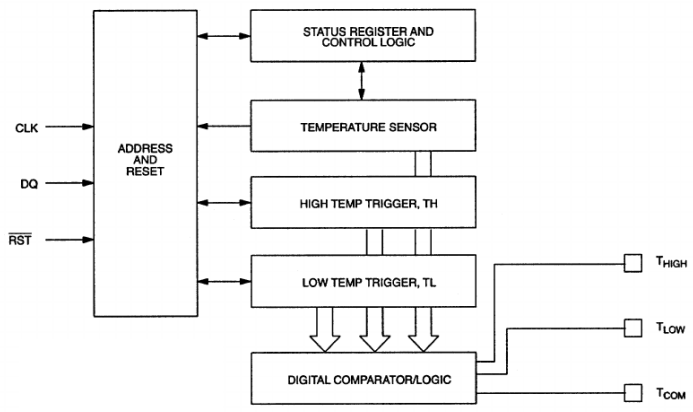

Abbildung 3: Aufbau des Sensors[1]

DETAILED PIN DESCRIPTION Table 1

PIN	SYMBOL	DESCRIPTION
1	DQ	**Data Input/Output pin** for 3-wire communication port.
2	CLK/$\overline{\text{CONV}}$	**Clock input pin** for 3-wire communication port. When the DS1620 is used in a stand-alone application with no 3–wire port, this pin can be used as a convert pin. Temperature conversion will begin on the falling edge of $\overline{\text{CONV}}$.
3	$\overline{\text{RST}}$	**Reset input pin** for 3-wire communication port.
4	GND	**Ground pin.**
5	T_{COM}	**High/Low Combination Trigger**. Goes high when temperature exceeds TH; will reset to low when temperature falls below TL.
6	T_{LOW}	**Low Temperature Trigger.** Goes high when temperature falls below TL.
7	T_{HIGH}	**High Temperature Trigger.** Goes high when temperature exceeds TH.
8	V_{DD}	**Supply Voltage.** 2.7V – 5.5V input power pin.

Table 2. DS1620 REGISTER SUMMARY

REGISTER NAME (USER ACCESS)	SIZE	MEMORY TYPE	REGISTER CONTENTS AND POWER-UP/POR STATE
Temperature (Read Only)	9 Bits	SRAM	Measured Temperature (Two's Complement) Power-Up/POR State: -60ºC (1 1000 1000)
T_H (Read/Write)	9 Bits	EEPROM	Upper Alarm Trip Point (Two's Complement) Power-Up/POR State: User-Defined. Initial State from Factory: +15°C (0 0001 1110)
T_L (Read/Write)	9 Bits	EEPROM	Lower Alarm Trip Point (Two's Complement) Power-Up/POR State: User-Defined. Initial State from Factory: +10°C (0 0001 0100)

Abbildung 4: Pin Beschreibung[1]

4.5.3 Befehle

Der Sensor wird vom Mikrokontroller über Befehle gesteuert. Hierzu muss zunächst ein Resetsignal gesendet werden, um den Sensor zu signalisieren, dass ein Kommunikationskanal geöffnet werden soll. Danach wird mithilfe einer 8-Bit Sequenz ein bestimmter Befehl aufgerufen.

Code	Befehl	Beschreibung
AAh	Read Temperature	Die nächsten 9 Bites von DQ repräsentieren die aktuell ausgelesene Temperatur
01h	Write Temperature max	Lege obere Temperaturgrenze fest
02h	Write Temperature min	Lege untere Temperaturgrenze fest

Tabelle 1: Benutze Befehle[1]

5 Fazit

Mit der Programmierung des intelligenten Wasserkochers mit vielen Funktionen (Wunschtemperatur einstellen etc.) ist bereits der Grundbaustein für einen Wasserkocher in der Praxis gelegt. Potential hat die Idee insofern, dass viele Funktionen, die dieser intelligente Wasserkocher unterstützt, bei den meisten handelsüblichen Wasserkochern nicht implementiert sind. Außerdem hat die Programmierung einen guten Einblick in die Programmiersprache Assembler gewährt und zum Verständnis dieser Sprache und dem Rechner allgemein beigetragen.

Literatur

[1] Maxim Integrated. Ds1620 specification.