Synthetisierung einer Aufzugsteuerung für einen 8051 Micro Controller

Projektbericht

für die Prüfung zum

Bachelor of Science

des Studienganges Angewandte Informatik
an der

Dualen Hochschule Baden-Württemberg Karlsruhe

von

Christian Verdion, Moritz Gabriel, Oliver Mahlke, Jan Kalweit

31.01.2015

Erklärung

gemäß § 5 (2) der „Studien- und Prüfungsordnung DHBW Technik“ vom 18. Mai 2009.

Ich habe die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Ort, Datum Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

In nahezu allen Dingen unseres Alltags steckt mittlerweile eine ganze Menge Technik. Kleine Sensoren sorgen dafür, dass ein bestimmter Zustand der Umwelt erkannt wird. Motoren sorgen dafür, dass sich Dinge bewegen. Lichter geben Zustände oder Informationen für den Menschen lesbar aus. In Mitten dieser Komponente findet man immer einen Micro Controller.

Kleine Computer, die wenig Strom verbrauchen und effizient Berechnungen durchführen. Ohne diese Chips wären viele technische Geräte wie wir sie kennen gar nicht möglich. Aufgrund der geringen Größe, und ihrer nahezu universellen Einsetzbarkeit wäre an keine Alternative zu denken, die so leise und komfortable ihren Dienst verrichtet.

Ebenso wie die Größe dieser Bausteine sinkt, steigt auch unser Anspruch an sie. Immer mehr Funktionen müssen eingebaut werden. Mit mehr Funktionen und komplexeren Tätigkeiten muss die Geschwindigkeit steigen um zum Beispiel Echtzeitanforderungen einzuhalten, wenn es um eingebettete Systeme in Autos oder anderen kritische Aufgabenbereiche geht.

1.1. Aufgabenstellung

Im Rahmen dieser Arbeit soll eine Aufzugsteuerung synthetisiert und Implementiert werden, die auf einen 8051 Micro Controller läuft.

2. Grundlagen

2.1. Assembler

Bei Assembler handelt es sich um eine hardwarenahe Programmiersprache. Das heißt, das geschriebene Programm ist für ein bestimmten Prozessor oder Chip-Architektur ausgelegt. Es kann dadurch möglich sein, dass ein und dasselbe Programm auf verschiedenen Prozessoren oder Chips nicht lauffähig ist.

In Assembler wird größtenteils mit einfachen Funktionen wie dem Addieren oder Subtrahieren zweier Werte programmiert. Auch Bit-weise Operationen wie „Und“, „Oder“ und „Shift“ sind möglich. Auf weitere Methoden und deren Funktion wird später eingegangen.

Der entstandene Assembler-Code wird danach kompiliert, also übersetzt, um direkt auf dem Chip ausgeführt zu werden. Das sogenannte Kompilat, also das Übersetzte Programm, ist dann in Maschinensprache geschrieben und für einen Menschen nicht mehr zu lesen. Lediglich die Micro Controller, oder die Micro Controller einer bestimmten Architektur, für die das Programm übersetzt wurde können mit diesem Code etwas anfangen.

2.2. 8051 Micro Controller

Der 8051 Micro Controller gehört zu der von Intel Micro Controller Familie MCS-51 und wurde im Laufe der 80er-Jahre eingeführt. Dabei handelt es sich um einen 8-Bit Micro Controller, was bedeutet, dass dieser mit 8-Bit langen Wörtern arbeitet, welche während eines Taktes verarbeiten kann. (1)
Heutzutage kommen immer noch viele Prozessoren mit 8-Bit Technologie zum Einsatz, so zum Beispiel bei einer Vielzahl von USB-Peripherie. Diese werden aber nicht mehr alle von Intel hergestellt. Neben Derivaten von Firmen wie Texas Instruments und Motorola gibt es natürlich auch andere Micro Controller, die auf 8-Bit Technologie setzen und gerade für eingebettete Systeme sehr beliebt sind.

Ein 8051 Micro Controller enthält eine CPU, also eine Recheneinheit, die über einen Bus mit anderen Bauteilen verbunden ist. Für dieses Projekt relevant sind zum einen die 4 Ein-/ Ausgabe Ports, die über jeweils 8 Pins oder Bit verfügen und die Möglichkeit externe Interrupts auszulösen.

Des Weiteren verfügt er noch über spezielle Anschlüsse, um einen externen Takt für die Timer anzuschließen und eine Datenleitung, um auf einen externen zusätzlichen RAM zuzugreifen. Für diesen Anwendungsbereich liefert der interne Timer aber ausreichend Genauigkeit. (2)

3. Konzept

Bevor man mit der Implementierung beginnen kann müssen zunächst einige Konzepte erstellt werden. Es muss eine Architektur aufgestellt werden, die angibt an welchen Anschlüssen welche Bauteile stecken und von welchen Voraussetzungen auszugehen sind.
Danach muss ein grober Ablauf festgelegt werden, dem der Programmverlauf folgen soll.

3.1. Architektur

Als erstes wurden die Portbelegungen der einzelnen Sensoren, Knöpfe, Motoren und Displays festgelegt. Es empfiehlt sich diese Belegung im Code als Header anzugeben. Bei der Planung hat man sich für folgende Belegung entschieden:

Port	Pin		
0	0 1 2 3 4 5 6 7	Knöpfe Außen Knöpfe Innen	Erdgeschoss 1. Stock 2. Stock Erdgeschoss 1. Stock 2. Stock
1	0 1 2 3 4 5 6 7	Motor für die Türen Motor für den Aufzug Licht für Stockwerkknopf gedrückt Gong wenn Tür offen	Auf Zu Hoch Runter Erdgeschoss 1. Stock 2. Stock

2	0 1 2 3 4 5 6 7	Sensor für die Tür Sensor für Stockwerk	Auf Zu Druck Lichtschranke Erdgeschoss 1. Stock 2. Stock
3	0 - 7	Display (7-Segment Anzeige)	
Externer Interrupt	0		
Register Register Register Register	0 1 2 7	Zwischenspeicher für Erdgeschoss gedrückt Zwischenspeicher für 1. Stock gedrückt Zwischenspeicher für 2. Stock gedrückt Richtung des Fahrstuhls	

Wie in der Architektur für Eingebettete und Sicherheitskritische Systeme wurde eine Steuerung mit Low-Active-Pegeln eingesetzt. Das bedeutet, dass Sensoren eine „0“ oder „kein Signal“ senden, sollten sie ausgelöst werden. So wird beispielsweise bei dem Drücken vom Knopf für die Stockwerkwahl im Aufzug für den 1. Stock der Pin P0.4 der Stromfluss unterbrochen, wodurch eine „0“ anliegt. Somit ist ohne viel Aufwand zu erkennen, ob ein Schalter oder eine Leitung defekt ist.
Selbiges Konzept wurde für die Ansteuerung des Displays verwendet. Bei der Motorsteuerung hat es sich als sinnvoller herausgestellt ihn mit „1“ also dem Anlegen einer Spannung zu steuern, damit er nicht unerwünscht beginnt zu laufen, sollte einmal ein Defekt bei einem Kabel vorliegen.

3.2. Programmablaufplan

Nachdem das Konzept entwickelt wurde konnte der Ablauf des Programms erstellt werden. Dabei setzt man im Allgemeinen auf einen Programmablaufplan, hierbei dienen die Knoten als Status im Programm. Durch verschiedene Eingangspegel, welche an den Pfeilen angegeben sind kann zwischen den verschiedenen Status. Es

empfiehlt sich hierbei verschiedene Szenarien zu modellieren, sodass ein Diagramm nicht unnötig groß und unübersichtlich wird.

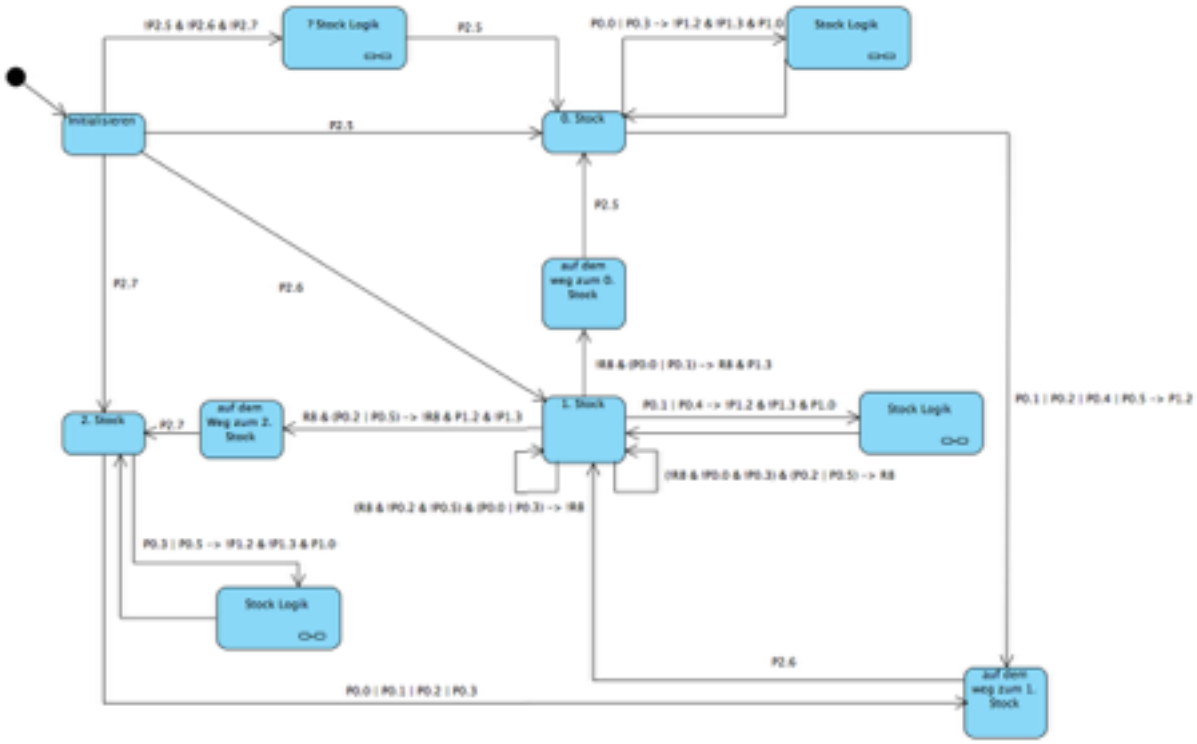

Abbildung 1: Programmablaufplan für die Programm Hauptschleife

In der Programmhauptschleife wird immer wieder überprüft, ob ein Stockwerk ausgewählt wurde. Wurde ein Stockwerk ausgewählt, so wird zu diesem Stockwerk gefahren und die Tür wird geöffnet. Unter Verwendung eines Hilfsregisters wird sichergestellt, dass zum einen nicht immer nur zwischen 2 Stockwerken gefahren wird, selbst wenn auch der 3. Stock gewählt wird und zum anderen, dass die Richtung des Fahrstuhls wenn es sich anbietet beibehalten wird.

Wurde ein Stockwerk ausgewählt und der Fahrstuhl befindet sich in diesem, so wird eine Routine ausgeführt, die eigens für den Ablauf dieses Falls entwickelt wurde.

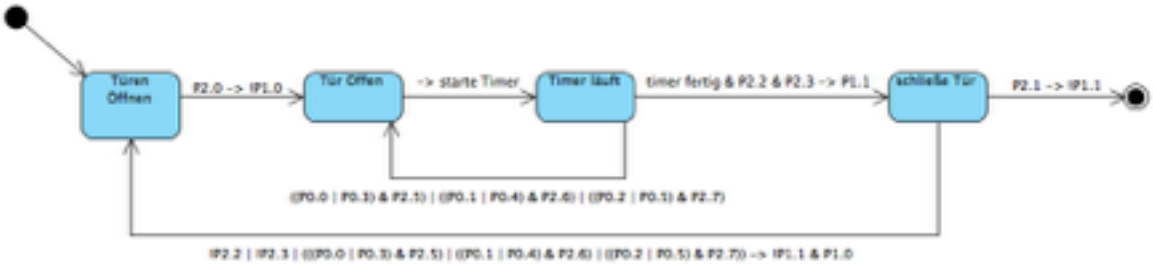

Abbildung 2: Programmablaufplan für die Logik innerhalb eines Stockwerks

In dieser Routine wird eben dieses Verhalten ausgeführt. Es werden die Türen geöffnet und nach Ablauf eines Zeitintervalls wieder geschlossen, sofern keiner der Sensoren betätigt oder Knöpfe gedrückt wurde.

Sollte die Aufzugsteuerung gestartet werden und der Fahrstuhl keinem Stockwerk zugeordnet werden können, so wird die folgende Routine durchgeführt, bei der der Fahrstuhl in einen Initialzustand geht, aber erst nachdem ein Knopf gedrückt wurde.

Abbildung 3: Programmablaufplan für die Erkennung des Initialzustands

4. Implementierung

Da zu diesem Zeitpunkt die gesamte Architektur steht, und sowohl die Algorithmen als auch die Programmablaufe feststehen kann bei der Implementierung der Assemblercode einfach anhand des Programmablaufplans programmiert werden.

Hierbei kommen einfache Befehle wie „MOV A,B“ bei dem der Inhalt aus „B“ in „A“ geschrieben oder „SETB C“ bei dem das Bit „C“ auf 1 gesetzt wird vor.
Auch logische Operationen wie „ANL A,B“ oder „ORL A,B“ wurden eingesetzt, um logische „Und“ und „Oder“ Operationen durchzuführen.
Um das Programm mit weniger Schleifen und dadurch performanter zu implementieren wurden diverse Interrupts eingesetzt, die im folgenden Teil erklärt werden.

4.1. Interrupt

Die Ansteuerung des Notruf-Knopfes erfolgt über einen Externen Interrupt. Dadurch wird der Programmcounter auf den Stack gelegt und eine eigenständige Routine gestartet. Ist diese abgearbeitet wird der Programmcounter wieder aus dem Stack geholt, wodurch das Programm dort fortgesetzt wird, wo es unterbrochen wurde.

Die Möglichkeit Interrupts überhaupt auslösen zu lassen wie folgt aktiviert werden:

Zuerst muss das Special-Function-Register gesetzt werden. Das Register heißt „IE“ und besteht aus folgenden Bits:

Bit	Interrupt
7	EA – Interrupts Global Aktivieren / Deaktivieren
4	ES – Serielle Interrupts Aktivieren / Deaktivieren
3	ET1 – Timer Interrupt 1 Aktivieren / Deaktivieren
2	EX1 – Externer Interrupt 1 Aktivieren / Deaktivieren
1	ET0 – Timer Interrupt 0 Aktivieren / Deaktivieren
0	EX0 – Externer Interrupt 0 Aktivieren / Deaktivieren

Abbildung 4: Interrupt Special-Function-Register zum Steuern der Interrupts (3)

Wird nun ein Interrupt gefeuert, so springt der Programmcounter nach dem speichern der aktuellen Position an eine für jeden Interrupt festgelegte Adresse diese Adressen lauten:

Einsprungadresse	Interrupt
003h	Externer Interrupt 0
00Bh	Timer Interrupt 0
013h	Externer Interrupt 1
01Bh	Timer Interrupt 1
023h	Serieller Interrupt

Abbildung 5: Einsprungadresse der Interrupts (3)

4.1.1. Timer Interrupt

Zur Realisierung der Fahrstuhlsteuerung ist es zweckmäßig, sowohl die Tasten zum Rufen des Fahrstuhls und zur Stockwerkwahl, als auch die Sensoren der Tür, also die Lichtschranke und der Berührungssensor an der Tür, über externe Interrupts an dem Micro Controller anzuschließen. Dies setzt allerdings voraus, dass genügend Eingänge für externe Interrupts zur Verfügung stehen. Da der für diese Lösung eingesetzte 8051 lediglich über zwei externe Interrupts verfügt und die Zahl der Sensoren und Tasten deutlich größer ist, muss mit so genanntem Polling gearbeitet werden. Das heißt, es wird immer wieder geprüft, welche Werte anliegen.

Folgende Probleme können hierbei auftreten:

- Systemauslastung:

Das System ist immer wieder damit beschäftigt die anliegenden Eingänge zu Überprüfen, selbst wenn dieser Vorgang nicht nötig wäre, da die Eingänge sich nicht verändert haben. Es kann sogar vorkommen, dass, wenn keine anderen Befehle ausgeführt werden müssen und der Polling-Aufruf explizit im Code steht, eine CPU Auslastung von 100% herrscht.

- Ereignisdauer zu kurz:

Wird der Aufruf explizit im Code ausgeführt, so kann es sein, dass durch einen externen Interrupt oder einem zu großen Abschnitt zwischen 2 Aufrufen der Polling Routine das Eingangssignal, wie etwa das Drücken der Taste zum Rufen des Aufzugs, zu kurz anliegt und somit der Tastendruck durch die Logik nicht berücksichtigt wird.

Diesen Problemen kann man allerdings dadurch entgegenwirken, indem die Polling Routine nach Ablauf einer bestimmten Zeit von einem Timer Interrupt ausgeführt wird. Hierbei empfiehlt sich die Verwendung von Intervallen zwischen 5 ms und 20 ms. Ist das Intervall zu groß, so kann es sein, dass ein kurzer Tastendruck nicht erkannt wird. Ist das Intervall zu klein, so kann es sein, dass die übrigen Befehle nicht schnell genug ausgeführt werden.

Durch den Aufruf durch einen Timer Interrupt ist es möglich, innerhalb eines anderen Interrupts den Timer Interrupt auszuführen, wodurch ungeplante, längere Pausen vermieden werden können. Außerdem bleibt die Prozessorauslastung relativ niedrig, sofern das Programm derzeit keinen Befehl ausführen muss.

Es empfiehlt sich die Tasten von den Türsensoren zu trennen. Da der Micro Controller über 2 Timer Interrupts verfügt, werden diese in 2 verschiedenen Timer Interrupt Service Routinen überprüft, die durch 2 verschiedene Timer Interrupts ausgelöst werden.

Der 8051 Micro Controller verfügt über 2 Timer: Timer0 und Timer1.

Die Timer zählen rückwärts und sind in 2 Hälften unterteilt. So gibt für den Timer0 zum einen TL0, was für Timer0_Low steht und von einer 8-Bit Zahl dargestellt wird und zum anderen TH0, welcher für Timer0_High steht. Dieser stellt auch eine 8-Bit Zahl dar. Beide Timer verfügen über 2 Modi. Im Modus 1 werden beide 8-Bit Zahlen zusammen genommen und stellen somit einen 16-bit Zähler dar, bei dem TH0 immer reduziert wird, sobald TL0 überläuft. Hierbei muss allerdings der Timer bei jedem Überlauf manuell neu initialisiert werden.

Alternativ lässt sich Modus 2 nutzen, bei dem lediglich eine 8-Bit Zahl im TL0 runtergezählt wird. Dieser Modus bringt den Vorteil, dass bei einem Überlauf neben der Interrupt Service Routine TL0 automatisch mit dem Wert aus TH0 initialisiert wird.

Beide Modi stehen sowohl für Timer0 als auch Timer1 zur Verfügung. Die Namenskonventionen sind für Timer1 analog.

Der 8051 wird in der Regel mit 12 MHz betrieben und zum Ausführen eines Befehls werden mindestens 12 Maschinenzyklen benötigt. Das bedeutet, dass nach etwa 1 Micro Sekunde ein Befehl abgearbeitet sein sollte. In einer Millisekunde werden also 1000 Befehle abgearbeitet.

Möchte man nun 5 Millisekunden abstände, so muss der Timer von 5000 auf 0 Zählen. Dafür reichen allerdings die 8-Bit oder 256 Werte nicht aus.

Aus diesem Grund ist es nötig die Werte auf beide Timer-Hälften aufzuteilen, und den Timer im Modus 1 zu betreiben, sodass (TL0+1)*(TH0+1) = 5000.

Im Falle des Timer0 für die Türsensoren ergibt sich also folgende Rechnung:
(249+1)*(9+1) = 5000

Aus diesem Grund hat man sich dafür entschlossen für die Türsensoren Timer0 in Modus2, TL0 mit 249 und TH0 mit 9 zu initialisieren.

Der Timer für die Tasten wird Timer1 sein, der auch in Modus 1 initialisiert wird, denn auch in diesem Fall genügen die 256 Werte des TL1 nicht, sodass nicht auf das Auto Reload des Modus1 Zurückgegriffen werden kann. Also müssen in beiden Fällen die Timer im Laufe der Service Routine neu initialisiert werden.

Die WerteTL1 und TH1 werden mit 249 und 19 initialisiert, wodurch 10000 Berechnungen durchgeführt werden. Zum Erfassen der Tasten genügt eine Überprüfung im 10 ms Takt.

Die Wahl der Timer 0 und 1 ist auch dadurch begründet, dass die Interrupts in einer bestimmten Reihenfolge ausgeführt werden, sollten mehrere gleichzeitig auftreten. Denn es ist wichtiger, dass die Tür wieder auf geht, sollte der Türsensor berührt oder die Lichtschranke durchbrochen werden, als dass der Knopfdruck erkannt wird.

Externer Interrupt 0
Timer Interrupt 0
Externer Interrupt 1
Timer Interrupt 1
Serieller Interrupt

Abbildung 6: Interrupt Priorität; Höchste Priorität oben (4)

5. Zusammenfassung

Dieses Projekt zeigt, dass mit wenig Aufwand und geringer Anzahl Code die Steuerung eines Aufzugs mit einigen Funktionen möglich ist. Diese Steuerung ist auf Hardware mit geringen Ressourcen wie Speicher und Rechenleistung möglich. Das Ergebnis ist natürlich mit weniger Codezeilen möglich, wobei das natürlich auf Kosten der Wartbarkeit gehen würde. In diesem Fall wurde sich für eine ausreichend große Implementierung entschieden und mit genügend Kommentaren versehen, um sich schnell im Code zu Recht zu finden. Die Kommentare werden im Übrigen vom Kompiler direkt beim Übersetzen hausgenommen, wodurch kein weiterer Speicherverbrauch dadurch zu befürchten ist.

Mit einem MicroController mit mehr Externen Interrupts und mehr Ein-/ Ausgabeports wäre eine deutlich umfangreichere Lösung möglich gewesen.

6. Literaturverzeichnis

1. **http://www.mikroe.com/chapters/view/65/chapter-2-8051-microcontroller-architecture/.** [Online] [Stand von: 09. 01 2015.]

2. **http://de.wikipedia.org/wiki/Intel_MCS-51.** [Online] [Stand von: 09. 01 2015.]

3. **http://www.8052.com/tutint.phtml.** [Online] [Stand von: 09. 01 2015.]

4. **http://mohitjoshi999.files.wordpress.com/2009/08/080409_0748_interruptpr1.png.** [Online] [Stand von: 08. 01 2015.]

5. **http://www.mikrocontroller.net/articles/8051_Timer_0/1.** [Online] [Stand von: 09. 01 2015.]