Putzi

Halbautonomes Fensterscheibenreinigungsgerät

Gruppe: Putzi

Marianne Krabi
Justin Freywald
Thomas Naprawski

Im Rahmen von:

Technische Informatik II
Prof. Dr. Miroslaw Malek
Sommersemester 2009

*„[...] our Education: […] **understand** confusious [...]“*

Inhaltsverzeichnis

1 Problemdarstellung

1.1 Einleitung

Höher, größer, gefährlicher – das ist das Motto unserer heutigen Gesellschaft. Auf der Jagd nach dem höchsten Gebäude der Welt konkurrieren Architekten, Immobilienmakler und Hotelbesitzer aus den verschiedensten Ländern. Damit auch jeder in das oder aus dem Gebäude schauen kann, müssen Glasfronten her. Diese Glasscheiben sorgen natürlich auch für gewünschte Reflexionen, die Imposanz und Größe noch verstärken. Je größer, desto besser – gerne auch in Erdbeben gefährdeten Gebieten wie Dubai oder China. Oder aber in heißen Wüstengebieten, in denen Sandstürme und sengende Hitze einen Aufenthalt im Freien häufig unmöglich machen. Jetzt haben wir also riesige Wolkenkratzer mit großen Verglasungen auf denen Sand, Pollen, Regen oder fettige Kinderhände ihre Spuren hinterlassen. Wer soll diese nun reinigen, damit sie ihren Glanz nicht verlieren?

Sicher könnte man teure Reinigungsfirmen mit vielen Mitarbeitern engagieren. Aber sicher werden diese einen Aufschlag für das Risiko in schwindelerregender Höhe zu arbeiten verlangen. Und sicher ist es ein großer Aufwand, die Befestigungen für die Arbeiter in jeder Etage anzubringen und sicherzustellen, dass diese auch bei Wind und Regen noch genügend Schutz bieten. Das muss auch leichter gehen, wie wäre es also, stattdessen, mit Hilfe einer Maschine schnell, sicher und mit weniger Aufwand die Fensterfronten zu putzen, ohne dass sich dabei ein Mensch den Gefahren oder Witterungen aussetzen muss?

Das macht unser Fensterputzgerät Putzi möglich. Nach Anbringung an der Scheibe reinigt es selbstständig die Scheiben, egal in welcher Höhe es sich befindet. Einfach zu bedienen und in Größe und Design an die Witterungsbedingungen angepasst, ist es eine große Hilfe bei der Reinigung großer Fensterflächen. Die genaue Arbeitsweise des Gerätes wird in Abschnitt 1.4 erläutert.

1.2 Problemanalyse

Natürlich gibt es bei dem Gerät bzw. der Arbeitsweise einige Schwierigkeiten, die man berücksichtigen muss.

Da wäre zum Einen die Fixierung des Gerätes an der Scheibe. Das Gerät muss dabei sein eigenes Gewicht plus Zusatzlasten des Putzmittels und des Wassers sicher an der Scheibe halten. Auch darf es durch Wind oder Erschütterungen nicht an Halt verlieren. Einher damit geht die Fortbewegung. Es muss eine gut koordinierte Bewegung sein, die das Gerät im stabilen Gleichgewicht hält und die Fixierung nicht beeinträchtigt. Dieses Problem wollen wir durch hydraulische Saugnäpfe realisieren, die das Gerät in vertikaler sowie horizontaler bewegen können und gleichzeitig einen optimalen Halt auch auf nassem Untergrund

sichern. Außerdem muss trotz der Bewegung das Putzen gewährleistet werden und analog dazu darf das Reinigen nicht das Fortbewegen stören.

Interessant für den Kunden wird sein, ob es verschiedene Putzmöglichkeiten gibt, bzw. passende vordefinierte Putzprogramme existieren.

Eine weitere zu lösende Aufgabe wird die Arbeitsweise auf der Scheibe selbst sein. Das Gerät wird die Fläche von oben nach unten in mäandernder Form abarbeiten. Die Frage ist, wie die Maschine die vertikale Putzdistanz bis zum Ende des Fensters ermittelt. Das könnte zum einen durch einen Mitarbeiter geschehen, dafür sind jedoch genaue bauliche Kenntnisse von Nöten, die dem Beaufsichtiger meist nicht bekannt sind. Sicherer ist es daher, dass das Gerät sich diese Distanz automatisch berechnet. Das kann zum Einen durch die zurückgelegte Distanz beim Hinauffahren der Scheibe geschehen. Besser wäre es jedoch, Sensoren zu benutzen, die die Oberflächenbeschaffenheit abtasten und dadurch bemerken wann die Scheibe zu Ende ist. Diese Variante kann auch zum Abtasten der seitlichen Scheibenenden genutzt werden.

Des weiteren muss man beachten, dass das Gerät mit einem Akkumulator betrieben wird. Daraus resultieren bestimmte Ressourcensicherungsmaßnahmen. Damit Putzi nicht mitten auf der Scheibe stehen bleibt, wenn der Akku leer ist, muss es selbst feststellen können, wann die Leistung noch reicht, um wieder sicher an die Unterseite der Scheibe zu kommen. Wenn dann der Akku gewechselt wurde, dann muss Putzi natürlich auch wissen, wo es die Arbeit an der Scheibe fortsetzen soll. Das ist auch bei niedrigem Wasser oder Putzmittelbestand der Fall.

Systemanforderungen

Putzi kommt mit geringen Systemanforderungen aus, der Akku wird inklusive Ladegerät mitgeliefert und das muss lediglich an eine Steckdose angeschlossen werden. Des weiteren benötigt das Gerät zum Funktionieren noch Wasser und Putzmittel.

1.3 Funktionsweise

Wie schon oben erwähnt, ist Putzi ein halbautonomes Scheibenreinigungsgerät. Bestandteile des Gerätes sind der Korpus, in dem sich die Fortbewegung sowie die Reinigung abspielen, ein Behälter für das Putzmittel bzw. Wasser, ein Auffangbecken mit integrierter Filteranlage zur Wasserreinigung und ein Schlauch zur zum hinaufpumpen des gereinigten Wassers in den Wasserbehälter.

In dem Korpus selbst befinden sich acht Schienen, an denen sich je zwei Saugnäpfe befinden. Vier Schienen zeigen in vertikale und ebenfalls vier Schienen in horizontale Richtung. Die Bewegung wird durch die passenden Schienen realisiert. In vertikaler Richtung werden die Saugnäpfe an den vier Schienen nacheinander bewegt, währen die horizontalen passiv bleiben und in horizontaler Richtung

bleiben die vertikalen Saugnäpfe passiv. Die genaue Abarbeitung der Fortbewegung durch die Bewegung der Saugnäpfe wird in Abschnitt 6.4 erläutert. An der Decke des Korpus sitzen Sprühköpfe für die Reinigung. Die Außenkante des Korpus wird durch Gummischienen gebildet, die für das Abziehen der Scheibe nach der Reinigung zuständig sind. Als Stromquelle dient ein Akkumulator und Solarzellen.

Nun kommen wir zu der eigentlichen Arbeitsweise von Putzi. Vor Beginn der Reinigung wird Wasser und Putzmittel in die dafür vorgesehenen Behälter gegeben. Der Nutzer kann mittels der Programme selbst entscheiden, zu welchen Anteilen er die Komponenten mischen möchte. Ein aufgeladener Akku wird eingesetzt und ein Putzprogramm wird gewählt. Dann wird Putzi an die Scheibe gesetzt und saugt sich dort fest. Dort beginnt der selbstständige Reinigungsprozess.

Putzi fährt hinauf bis ans Scheibenende unter Benutzung der vertikalen Saugnäpfe. Die zurückgelegte Distanz wird dabei gemessen, um bei evtl. Batterieschwäche wieder zurück zum Boden zu gelangen. Der Putzprozess läuft dabei noch nicht. Dann wird von den vertikalen auf die horizontalen Schienen umgestellt, sodass Putzi nun die quer über die Scheibe fährt. Dabei startet der Reinigungsprozess. Die Sprühstärke und -dauer, sowie die Geschwindigkeit des Gerätes wird durch die Putzprogramme gesteuert und in Abschnitt 4.9 näher erklärt. Eins ist jedoch bei allen Programmen gleich, denn die Reinigung wird durch das Sprühen erreicht und das Abziehen des Wassers von der Scheibe geschieht automatisch durch die Gummiränder, wenn das Gerät sich fortbewegt.

Wenn Putzi nun durch Sensoren feststellt, dass es die rechte Seite der Scheibe erreicht hat, schalten sich wieder die vertikalen Schienen zu und die horizontalen ab, sodass Putzi auf die nächste zu putzende Bahn fährt. Dort angekommen, setzt das Gerät diesen Prozess mäandernd fort, bis es an der Unterseite der Scheibe angekommen ist.

Wenn es nun aus diversen Gründen, wie zum Beispiel bei Energieknappheit oder per manuellen Rückruf, zum Abbruch des Putzens kommt, dann merkt sich Putzi eben diese vertikale Distanz um nach Wiederaufnahme der Arbeit auf diese Höhe zurückzukehren und wieder an der linken Seite zu beginnen.

Eine weitere Funktion ist die automatische Wasseraufbereitung. Durch die Sprühköpfe wird ständig Flüssigkeit in das Gerät gesprüht, das sich an der Unterseite sammelt. Um dann in das Auffangbecken zu gelangen, durchläuft das Wasser verschiedene Filterstationen, bis es gereinigt dort ankommt. In dem Sammelbecken befindet sich nun ein Sensor, der auf einer bestimmten Höhe angebracht ist. Wenn nun Wasser an den Sensor kommt, wird das hinaufpumpen in den Putzwasserbehälter automatisch für eine vorgegebene Zeit in Gang gesetzt.

1.4 Einschränkungen

Bei der Bearbeitung dieses Projektes gingen wir ausschließlich nach informationstechnischen Kriterien vor. Dies impliziert insbesondere, dass außer nebensächlichen Betrachtungen jegliche physikalischen außer Acht gelassen worden sind, obwohl eben auch solche Betrachtungen bei diesem Projekt eine große Rolle spielen. Da wäre das Gesamtgewicht anzuführen, dass möglichst gering sein muss. Die Belastbarkeit und Tragfähigkeit von Scheiben müsste im Allgemeinen und abhängig von ihrer Größe erörtert werden. Das äußere Form müsste unter aerodynamischen Gesichtspunkten neu entworfen werden, somit bei hohen Windgeschwindigkeiten in großen Höhen das Gerät sich einwandfrei bewegen könnte.Weiterhin ist zu ermitteln, wie hoch der Energieverbrauch des Gerätes bei gewöhnlicher Auslastung ist und welche Akkumulatorart dementsprechend zu verbauen wäre.

In der Praxis müssten sich die Putzprogramme erst beweisen. Im Allgemeinen wurde keine Komponente dieses Roboters getestet. Ohnehin müsste sich die Auswahl der technischen Elemente einer finanziellen Prüfung unterziehen.

2 Architektur

2.1 Datengewinnung

Der Wasserstand in Eingebauten Containers wird von zwei Füllstandsensoren für flüssige Medien überwacht. Sie signalisieren jeweils die Zustände: „Ausreichend“ – (1) und „nicht ausreichend“ – (2). Für den Spülmittel Container ist ein Sensor der gleichen Art zuständig.

Das Fensterscheibenreinigungsgerät benutzt folgende Typen von Sensoren. Die ersten sind Konduktive Sensoren mit Relaisausgang (Liqui-Switch), die für die Füllstandserfassung oder zur Leckage-Detektion elektrisch leitfähiger Flüssigkeiten verantwortlich sind. So werden wir immer wissen können, wie hoch der Füllstand ist.

Die Energieleistung der Batterie überprüft ein weiterer Sensor, falls die kritisch Stromleistung wird. Um den Strominhalt zu kontrollieren, nutzen wir einen fibre-optical voltage Sensor, der die Daten misst und die Daten Zu dem LED Display, das in dem Gerät eingebaut ist, übergibt.

Um die Positionen der Schienenlänge zu kontrollieren, brauchen wir 4 Sensoren, die spezielle Zustände kontrollieren:

Schiene vertikal links oben verschoben (Bitzahl durch 1 definiert), noch nicht verschoben (Bitmuster durch 0 definiert);

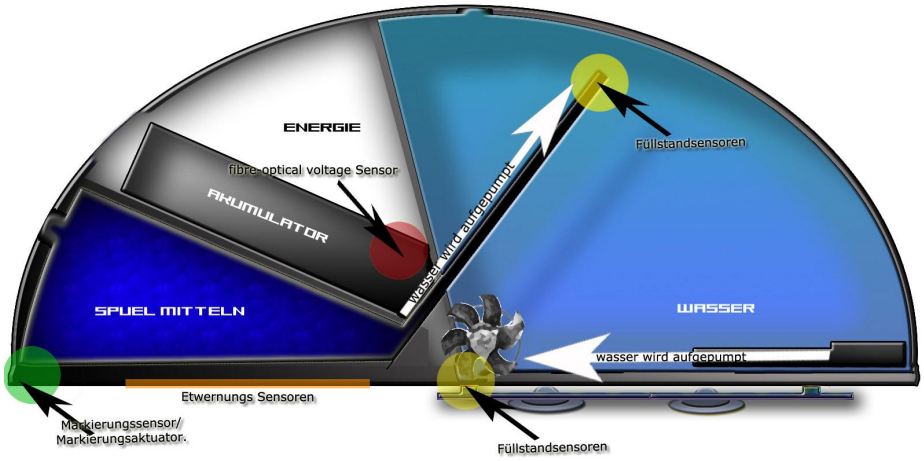
Schiene vertikal rechts oben verschoben (1), noch nicht verschoben (0);

Schiene vertikal links unten verschoben (1), noch nicht verschoben (0);

Schiene vertikal rechts unten verschoben (1), noch nicht verschoben (0);

Weiterhin nutzen wir Aktuatoren, die prüfen ob unsere Saugknöpfe ausreichend angesaugt sind (Bitmuster durch 1 definiert) oder noch noch weiterhin angesaugt werden sollen (Bitmuster 0). Das Gerät soll des weiteren „wissen“, wo es sich befindet. Deshalb nutzt es einen Markierungssensor und einen Markierungsaktuator.

Das Gerät mit den angezeigten Sensoren/Aktuatoren:



2.2 Dateneingabe

Es gibt zwei Möglichkeiten der Inbetriebnahme des Gerätes – manuell per Kippschalter oder per Fernbedienung.

In der manuellen Spezifikation haben wir einen Schalter der 2 Zustände besitzt – „ein“ oder „aus“. Er besitzt ein Flip-Flop, das vom Prozessor zurückgesetzt werden kann und vom Benutzer durch Drücken erneut gesetzt wird.

2.3 Datentypen und -formate

Die folgende Tabelle gibt die Datentypen- und Formate an, mit denen mit den Sensoren und Aktuatoren kommuniziert wird. Die Kanäle und die Richtung sind ebenfalls angegeben. Alle Zahlen sind Vorzeichenlos, da keine negativen Zahlen übermittelt werden brauchen, daher existiert auch kein Vorzeichenbit.

Kanal	Daten für/von...	Richtung	Datenformat

1	Füllhöhe Wasserstand mittels Ultraschall	In	2 bit
2	Füllhöhe Putzmittelkammer mittels Ultraschall	In	2 bit
3	Stromanzeige	In	2 bit
4	Motor zum Putzmittelmischen	Out	4 bit
5	Kompressor zum Wasserpumpen	Out	3 bit
6	Kompressor zum Ansaugen der Saugnäpfe	Out	1 bit
7+8	Oberflächenbestimmung seitlich (links,rechts)	In	1 bit
9+10	Obere und untere Oberflächenbestimmung	In	1 bit
11	unbelegt		
12	Position der Schiene vertikal 1 (links oben)	In	2 bit
13	Position der Schiene vertikal 2 (rechts oben)	In	2 bit
14	Position der Schiene vertikal 3 (links unten)	In	2 bit
15	Position der Schiene vertikal 4 (rechts unten)	In	2 bit
16	Position der Schiene horizontal 1 (l.o.)	In	2 bit
17	Position der Schiene horizontal 2 (r.o.)	In	2 bit
18	Position der Schiene horizontal 3 (l.u.)	In	2 bit
19	Position der Schiene horizontal 4 (r.u.)	In	2 bit
20	LED Anzeige Gerät	In/Out	4 bit
21	LED Anzeige Fernbedienung	In/Out	4 bit
22	Schmutzwasser Füllstandmessung	In	1 bit
23	Empfangen (Fernbedienung)	In	5bit
24	Senden (Fernbedienung)	Out	5bit
25-32	Motor Schienen 1-8 (25 = l.o. 1, 26 = r.o. 1, 27 = l.u. 1, 28 = r.u. 1, 29 = l.o. 2, 30 = r.o. 2, 31 = l.u. 2, 32 = r.u. 2)	Out	2 bit
33-40	Motor der 8 Saugnäpfe (horizontal) (33 = l.o. 1, 34 = r.o. 1, 35 = l.u. 1, 36 = r.u. 1, 37 = l.o. 2, 38 = r.o. 2, 39 = l.u. 2, 40 = r.u. 2)	Out	2 bit
41-48	Motor der 8 Saugnäpfe (vertikal) (41 = l.o. 1, 42 = r.o. 1, 43 = l.u. 1, 44 = r.u. 1, 45 = l.o. 2, 46 = r.o. 2, 47 = l.u. 2, 48 = r.u. 2)	Out	2 bit
49	Markierungsaktuator	Out	1 bit
50	Markierungssensor	In	1 bit
51-66	Ventilklappen der Saugnäpfe vertikal (5= l.o. 1, 52= r.o. 1, 53= l.u. 1, 54= r.u. 1, 55= l.o. 2, 56= r.o. 2, 57= l.u. 2, 58= r.u. 2) und horizontal (59= l.o. 1, 60= r.o. 1, 61= l.u. 1, 62= r.u. 1, 63= l.o. 2, 64= r.o. 2, 65=l.u. 2, 66=r.u. 2)	Out	1 bit
67-98	Position der Saugnäpfe an der Schiene vertikal (67-70= l.o, 71-74= r.o., 75-78= l.u., 79-82= r.u.) und horizontal (83-86= l.o., 87-90= r.o., 91-94= l.u., 95-98= r.u.)	In	1 bit

99-106	Motoren der einzelnen Saugnäpfe horizontal (99= l.o. 1, 100= l.o. 2, 101= r.o. 1, 102= r.o. 2, 103= l.u. 1, 104= l.u. 1, 105= r.u. 1, 106= r.u. 2)	Out	2 Bit
107-114	Motoren der einzelnen Saugnäpfe vertikal (107= l.o. 1, 108= l.o. 2, 109= r.o. 1, 110= r.o. 2, 111= l.u. 1, 112= l.u. 1, 113= r.u. 1, 114= r.u. 2)	Out	2 Bit

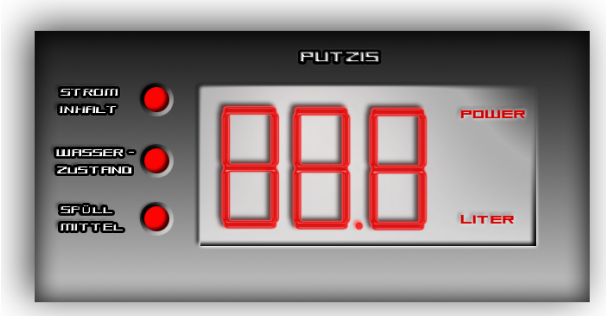
2.4 Verarbeitung- und Speicherung

2.5 Datenausgabe

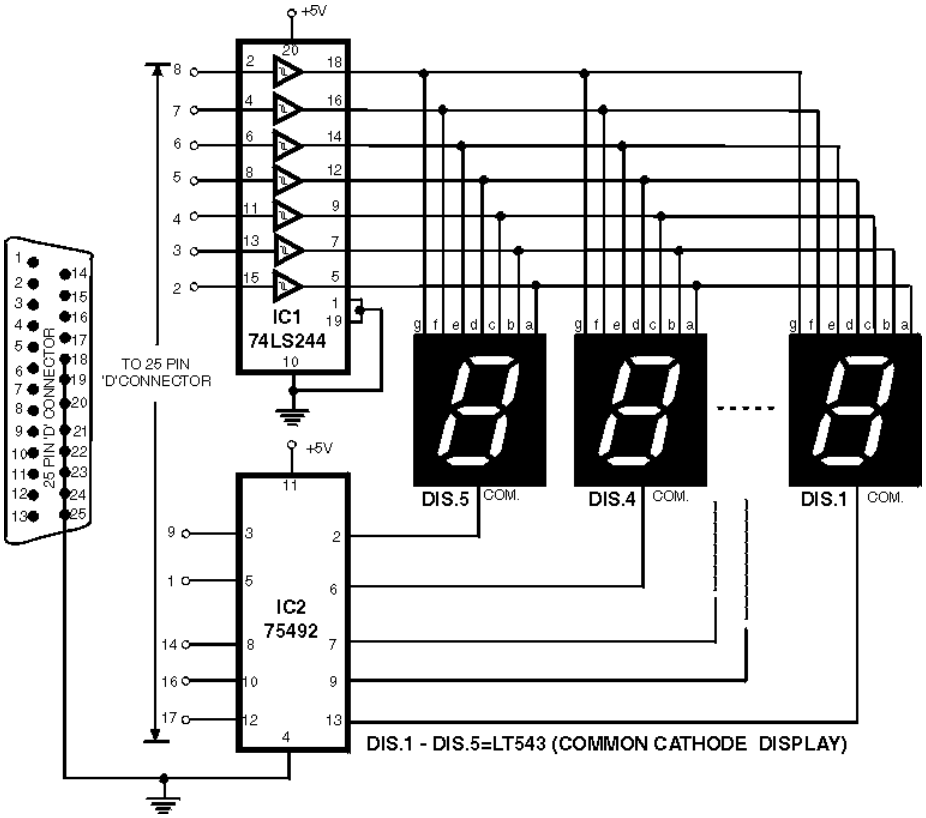
Unser Fensterscheibenreinigungsgerät kann vier verschiedene Daten ausgeben. Dazu zählen vorher festgelegte Funktionen, wie Programme, eine SD-Karte, die Fernbedienung und ein Display auf dem Gerät selbst. Zu diesem Display gehören die Anzeigen von Spülmittel- und Wasserfüllstatus, und die Batteriestatusanzeige. Diese können auf Wunsch nach Knopfdruck in den Einheiten % und Liter angegeben werden. Diese Anzeige nutzt ein einfaches Sevensegment-Display, da dies die Energie sparendste und ökonomischste Variante ist.

Die Fernbedienung bekommt ihre Daten aus einer Funksende-/empfangseinheit, die in Putzi eingebaut ist. Die Ausgabe der Daten findet auf einem kleinen 1,5 Zoll LED Display statt, das an eine fünf-Volt-Batterie angeschlossen ist.

Komponenten : 3 Knöpfe, die die Daten ausgeben; Analog Display



Das Display funktioniert folgendermaßen:



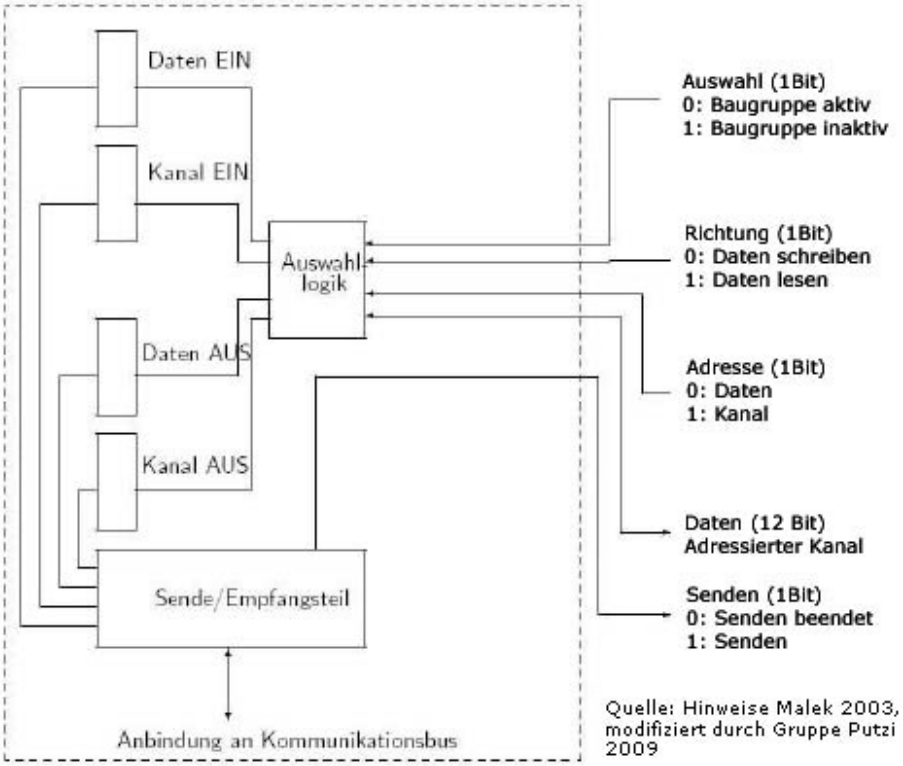
2.6 Roboter

Der Roboter besitzt eine Geräte-Identifikationsnummer, die im Prä-Systemcode festgelegt ist und sich von anderen Geräten unterscheiden muss. Mit der Fernbedienung, eine Erweiterung, kann gezielt ein Roboter angesprochen und befähigt werden.

3 Mikroprozessor-Umgebung

3.1 Schnittstellen

Schnittstellen befinden sich zwischen den Sensoren, Aktuatoren und den Speicherstellen und dem Kommunikationsbus. Es gibt hier eine gemeinsame Schnittstelle, mit der über die Angabe der verschiedenen Kanäle mit den Controllern kommuniziert werden kann.



Das Senden der Daten geschieht, indem das Richtungsbit auf 0 – Schreiben(-Senden) gesetzt wird, das Adressbit auswählt ob ein Kanal oder Daten geschrieben werden sollen. Dann wird die Kanalnummer in das Kanalregister geschrieben gefolgt von einem 5-Bit Datenwort in das Datenregister. Entsprechend diesen Angaben werden dann die vorgegebenen Daten an den gewünschten Kanal gesendet. Wenn diese Daten gesendet wurden, folgt das Signal 0 – Senden beendet, der Vorgang wird abgeschlossen.

Das Lesen der Daten geschieht ähnlich. Das Richtungsbit wird auf 1 – lesen gesetzt, das Adressbit wird auf Kanal gesetzt. Dann wird das Kanalregister gelesen. Wenn die folgende Kanalnummer nicht 00000 anzeigt, befindet sich ein Kanal im Register und der Datenregister kann gelesen werden.

Ein Sende(Schreib-)vorgang dauert im besten Fall fünf Mikrosekunden und im schlechtesten Fall 10 Millisekunden.

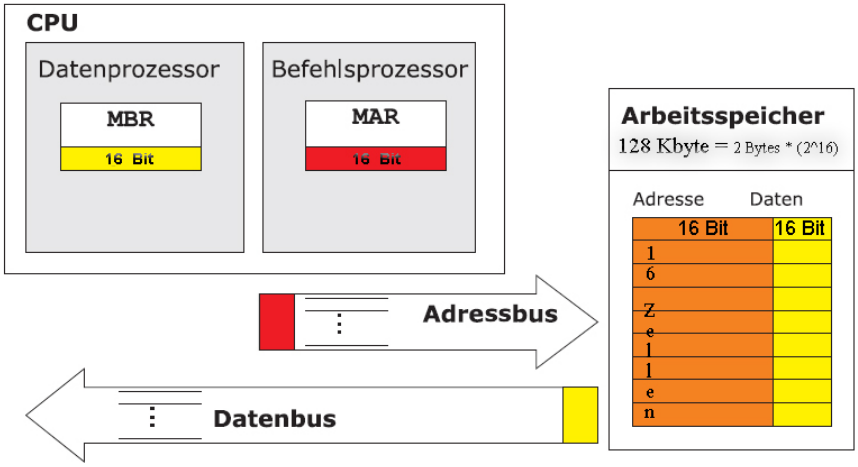
3.2 Primärspeicher

Der Hauptspeicher umfasst lediglich 128 Kbyte ($2Bytes * 2^16$) Speicher. Er wird gewöhnlich über MAR und MDR mit dem Systembus verbunden.

Adressiert werden 16 Bit Daten mittels 16-bittigen Adressen. Offensichtlich erfolgt die Datenübertragung ebenfalls über 16 Bit (16 Datenleitungen).

Speicherbelegungsplan des Primärspeichers

Hexadezimaler Adressbereich		**Dezimaler Adressbereich**		**Größe (MByte)**	**Reserviert für**
0x0000	0x3FFF	0	16383	32	Systemcode
0x4000	0x7FFF	16384	32767	32	Programmcode
0x8000	0xBFFF	32768	49151	32	Derzeitiges Programm
0xC000	0xFFFF	49152	65535	32	Stack



3.3 Sekundärspeicher

Es wird eine beliebige SD-Speicherkarte ohne Dateisystem als Sekundärspeicher eingesetzt. Sie enthält System-, Putzprogrammcode und Daten (z.B. Debuglogs). Die mögliche Speicherkapazität von derzeit mehr als 4 GB ist für unsere Zwecke daher mehr als ausreichend.

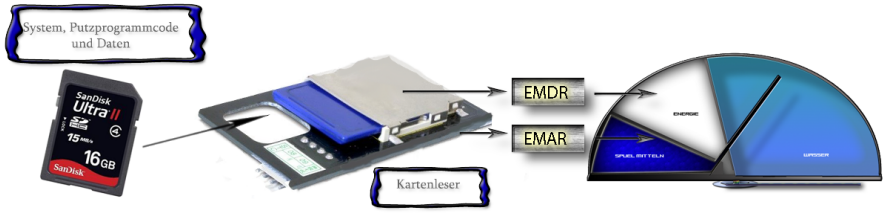
Die Einbindung geschieht über einen Kartenleser. Der Einfachheit gehen wir davon aus, dass das Lesegerät am Systembus angeschlossen ist und intern mit 32 Bit arbeitet. Dies impliziert 32 bittige Adressen sowieso eine Datenbreite von 4 Byte (32 Datenleitungen), also maximal 16Gb Speicher ($4Bytes * (2^32)$).

Die speziellen Register, die als Schnittstellen zwischen Systembus und Lesegerät funktionieren, nennen wir **EMDR** (Extern MDR) und **EMAR** (Extern MAR) und sind in Assembler über EMDR[L], EMDR[H] und EMAR[L], EMAR[H](Low, High) direkt ansprechbar, indirekt kann man mittels **MVLH**

Daten auf die Register EMAR oder EMDR legen. Im Prä-Systemcode finden sich roboterspezifische Einstellungen, beispielsweise Konstanten wie die Geräte-ID.

Speicherbelegungsplan des Sekundärspeichers

Hexadezimaler Adressbereich		**Dezimaler Adressbereich**		**Größe (MByte)**	**Reserviert für**
0x0000 0000	0x1FFF FFFF	0	536870911	2	Bootcode
0x2000 0000	0x3FFF FFFF	536870912	1073741823	2	Prä-Systemcode
0x4000 0000	0x7FFF FFFF	1073741824	2147483647	4	Systemcode
0x8000 0000	0xFFFF FFFF	2147483648	4294967295	16	Programmcode
unbekannt				unbekannt	Beliebig



3.4 Sensoren und Aktuatoren

Die folgende Tabelle listet die Kanäle der Sensoren und Aktuatoren aus Abschnitt 2.3 auf und erklärt die vergebenen Bitmuster. Es wird das allgemein übliche Little-Endian-Format genutzt.

Kanal	Bitmuster	Bedeutung
1-2	00 bis 11	Behälter leer bis Behälter voll
3	00 bis 11	Akku leer bis Akku voll
4	0000 bis 1111	Je nach Putzprogramm 16 verschiedene Mischmöglichkeiten
5	000 bis 011 100 bis 110	Pumpen von Putzmittel- und Wasserbehälter zum Gerät mit variabler Stärke und Dauer Pumpen vom Schmutzwasserbehälter zum Wasserbehälter mit variabler Dauer
6	0/1	Loslösen/ Ansaugen
7 bis 10	0/1	Kein Glas/ Glas
12 bis 19	00 bis 11	4 Positionen, 00 ist links, bzw. oben und 11 ist rechts bzw. unten

20	0000 bis 1111	1.Bit 0/1 → Wasser leer/ ausreichend 2.Bit 0/1 → Putzmittel leer/ ausreichend 3.Bit 0/1 → Akku leer/ ausreichend 4.Bit 0/1 → Gerät an/ aus
22	0/1	Behälter muss nicht entleert werden/ muss entleert werden
23 bis 24	00000 bis 11111	1. - 4.Bit verschiedene Putzprogramme 5.Bit 0/1 beende Putzen und fahre Gerät zum Boden/ fortfahren mit Putzen
25 bis 48	00 bis 11	1.Bit 0/1 Motor an/ aus 2.Bit 0/1 Drehrichtung des Motors (links / rechts)
49	0/1	Aktuator an/ aus
50	0/1	Sensor (Markierung detektiert/ keine Markierung detektiert)
51-66	0/1	0 (zu, Druckausgleich mit Kompressordruck, Saugnapf ist fest) 1 (auf, Druckausgleich mit Außendruck, Saugnapf kann gelöst werden)
67-98	0/1	0 →kein Kontakt mit Saugnapf 1 → Kontakt
99-114	00 bis 11	1. Bit → 0 Richtung vorwärts, 1 rückwärts 2. Bit → 0 Motor aus/ 1 an

4 Mikroprozessor

4.1 Spezifikation

Unser Prozessor nutzt drei Adressen (3-Adressmaschine), nach CISC-Prinzip. Er hat sechs Adressierungsarten (4 (+2) → 2 Bit Adressmodus) und unterstützt maximal 64 Assemblerbefehle (→ 6 Bit Instruktionen), wovon 33 genutzt werden, keine Pipeline, keine (hardware) Exceptions, keine hardware Interrupts (stattdessen in Assembler modellieren) und keine Caches.

Es ist ein Stack vorgesehen zur Verwaltung von stacknutzenden Branches und Return (RET). Der direkte Zugriff auf den Stack ist möglich.

Wortbreite und Registergröße ist 16 Bit, die Befehlsbreite 64. Auch stehen eine reiche Anzahl von 128 Register R0. . . R127 zur Verfügung (→ 7 Bit für Registeradressierung).

R0. . . R120 sind Mehrzweckregister (**GPR**), die somit helfen, Timingprobleme zu vermeiden, indem Speicherzugriffe minimiert werden. Das Register R121 ist der Stackpointer, die Register R122. . . R127 sind vorgesehen, intern vom Mikroprozessor (R122. . . R124) und dem Präprozessor des Compilers (R125. . . R127)

verwendet zu werden.

Weiterhin verfügt der Prozessor über folgende *spezielle Register:*

Nur-Lese-Register	**PC**, **IR**, **CCR** (**C**ondition-**C**ode-**R**egister), **O0. . . O2** (Operandenregister)
Nur-Schreib-Register	**EMAR** und **MAR**
Lese-Schreib-Register	**EMDR** und **MDR**

Der Hauptspeicher wird wort-, der externe Speicher doppelwortadressiert. Es wird das allgemein übliche Little-Endian-Format genutzt.

Unserer Einschätzung nach reicht eine Taktfrequenz von wenigen Megahertz aus, um die in Assembler geschriebenen, optimierten, meist sensorbasierten Programme zeitnah ausführen zu können (der Commodore 64 kam auch mit etwa 1 Mhz aus).

Folgende Adressierungsarten werden unterstützt, wobei (m[Adresse] für Hauptspeicherzugriff steht):

Bit	Typ	Notation	Funktion
00	Registerdirekt	Rn	EA=Rn OP=m[EA]
01	Registerindirekt	@Rn	EA=m[Rn] OP=m[EA]
10	Registerdirekt indiziert	(R1+Offset)	EA=R1+Offset OP=m[EA]
11	Registerindirekt indiziert	@(R1+Offset)	EA=M[R1+Offset] OP=m[EA]
00	Unmittelbar, alle außer erstem Operand	#n	EA=PC OP=m[EA], (INC PC)
01	Speicherunmittelbar	[n]	EA=m[PC] OP=m[EA], (INC PC)
10	n.a.	n.a.	n.a.
11	n.a.	n.a.	n.a.

4.2 Spezifikation der ALU

Die ALU arbeitet auf 16-Bit Werten. Im Folgenden werden die Funktionen der ALU bekannt, wobei die Steuersignale nicht schaltungstechnisch evaluiert sind, Änderungen vorbehalten:

Funktionen der ALU:

Steuer-signal	F=	Funktion
0000	A+B	Addition
0001	A-B	Subtraktion
0010	B	Transfer
0011	B-1	Dekrementierung
0100	B+1	Inkrementierung
0101	A == B	Relation: Gleichheit
0110	A != B	Relation: Ungleich
0111	A > B	Relation: Größer
1000	A >= B	Relation: Größer oder Gleich
1001 - 1011	n.a.	n.a.
1100	A XOR B	Bitweise-XOR-Operation
1101	NOT B	Bitweise-Negation
1110	A AND B	Bitweise-Und-Verknüpfung
1111	A OR B	Bitweise-Oder-Verknüpfung

Drei Flaggen werden im **CCR** (Condition-Code-Register) gespeichert. Es wird garantiert, dass die genau diese drei Flaggen der ALU nach einer Operation wie erwartet gesetzt bleiben.

Operationen, die auf sonst typische ALU-Flaggen zugreifen wie EQUAL oder GREATERTHAN, werden innerhalb des Befehls angegeben (konditionale Branches). Es werden keine Ausnahmen bei Zahlenbereichsüber/ -unterschreitungen geworfen. Der Zugriff auf die einzelnen Bitstellen ist schaltungstechnisch realisiert (teste Bit).

Übersicht zu den Flaggen im CCR:

Position	Flagge	Funktion
0	CARRY	Gesetzt bei arithmetischer Operation mit Übertrag
1	OVERFLOW	Wenn Wert der Operation zu groß, um 16 Bit zu genügen
2	UNDERFLOW	Wenn Wert der Operation zu klein, um 16 Bit zu genügen
3	EQUAL	Gesetzt durch Vergleich (CMP), wenn Operanden gleich
4	GTE	Gesetzt durch Vergleich (CMP), wenn erster Operand größer oder gleich als Zw
5	STE	Gesetzt durch Vergleich (CMP), wenn erster Operand kleiner oder gleich als Zw
6	GT	Gesetzt durch Vergleich (CMP), wenn erster Operand größer als der Zweite
7	ST	Gesetzt durch Vergleich (CMP), wenn erster Operand kleiner als der Zweite

Das CCR ist intern als 8-Bit Register implementiert.

4.3 Realisierung der ALU

Die 16-Bit ALU besteht intern aus sechzehn **1-Bit ALU**, die jedes Bit separat verarbeiten und das Addierwerk (als Backend) bilden. Dazu sind die 1-Bit-

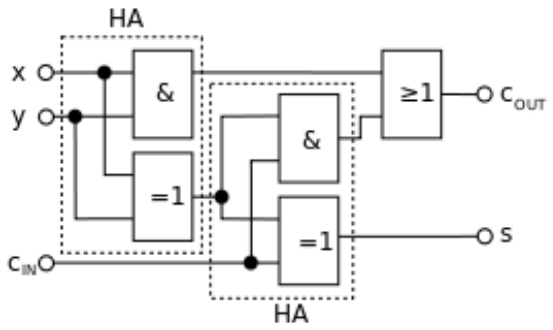


Abbildung 1: Ein Volladdierer (Quelle: Wikipedia)

ALU an die Steuerleitungen der **CU** (Control Unit) der ALU angeschlossen und höherwertige Bit verarbeitende ALU über deren Carry-In mit dem Carry-Out der ALU, die niederwertige Bit verarbeiten, verbunden. Jede 1-Bit ALU bearbeitet eine Binärstelle der Eingänge A und B. Die Ausgänge aller 1-Bit ALU entsprechen Z. Jede 1-Bit ALU ist hauptsächlich ein Volladdierer (hier S= Z0, x= A0,y= B0).

Zur Subtraktion (Frontend) werden wie gewöhnlich die Eingänge A negiert zu $\bar{A}$. Ein Multiplexer nimmt jeweils $\bar{A}$ und A an, der an ein gesondertes Subtraktionssteuersignal der CU angeschlossen ist, der $\bar{A}$' an den jeweiligen Volladdierer liefert. Zusätzlich wird bei einer Subtraktion (Subtraktionssteuersignal ist 1) ebenso das CarryIn der ALU (C0) auf 1 gesetzt, um ein Zweierkomplement in der Gleichung A+B von der Zahl B zu bilden, sodass in der Tat A+(-B), genauer A+($\bar{B}$-1) gerechnet wird, wobei $\bar{B}$ die Negation meint. Jede 1-Bit ALU ist neben einem Volladdierer darüberhinaus mit einem 16-Bit-Komparator ausgestattet, der die Vergleichsoperationen ausführt.

Für jede Stelle gilt dabei:

H_i=(A_i or B_i) and ($\bar{A}_i$ or $\bar{B}_i$)= $\overline{XOR\ A\ B}$

EQ=(A==B)= H_0 and H_1 and H_2 and ... and H_{15}

GT=(A>B)=$\bar{A}_{15}$ or B_{15} or () AND $\bar{A}_{14}$ or B_{14} or (H_{15}) ... $\bar{A}_3$ or B_3 or (H_{15} OR H_{14} OR H_{13} ... OR H_4) AND (... H_3) OR $\bar{A}_2$ or B_2 AND (... H_3 OR H_2) OR $\bar{A}_1$ OR B_1 AND (... H_3 OR H_2 OR H_1) OR $\bar{A}_0$ OR B_0

ST=(A<B)=$\overline{GT}$ AND $\overline{EQ}$

Weitere Operationen wie NOT, OR, AND, XOR sind durch bitweise Anwendungen (ohne Carry) von An und Bn als Eingänge an den Gattern Nicht, Oder, Und, XOR zu realisieren.

Die Ink- und Dekrementiereinheiten jeweils funktionieren über das Addierwerk oder das Addierwerk nutzende Subtrahierwerk mit 1 (A_0= 1, A_1... A_{15}= 0)

oder +(-1) als Eingangssignal.

Die Schiebebefehle werden hauptsächlich mittels einem Schieberegister nach der ALU umgesetzt.

4.4 Befehlsformat

Legende

Instruktion	OPC
Operand X	ADX
Adressmodus für Operand X	OPX

Jeder Operand hat 16 Bits zur Verfügung. Registeradressen benötigen 7 Bit. Die Befehlsbreite beträgt immer 64 Bit. Im folgenden bezeichnet #NEXT eine Sprungmarke.

Allgemeiner 3-Operandenbefehl (bei Arithmetisch):

Name	OPC	AD1	OP1	AD2	OP2	AD3	OP3	(null)
Bits	6	2	16	2	16	2	16	4

ADD R0 R1 @R2 | ADD R0 #22 #44

Allgemeiner 2-Operandenbefehl (bei Transfer, Logisch):

Name	OPC	AD1	OP1	AD2	OP2	(null)
Bits	6	2	16	2	16	22

MOV R0 @R2
MOV R0 #44

2-Operanden, konditionaler Sprungbefehl:

Name	OPC	AD1	OP1	AD2	OP2	AD3	Adresse	(null)
Bits	6	2	16	2	16	2	16	4

BEQ R1,R2,#NEXT

1-Operandensprungbefehl:

Name	OPC	(null)	AD3	Adresse	(null)
Bits	6	18	2	16	18

BRA #NEXT
BRA R7

0-Operandenbefehl

Name	OPC	(null)
Bits	6	58

RET

Operanden

Modus	Bits			Beschreibung
	Adress/-modus	Operand	Typ	
Registerdirekt/-indirekt	1	1+7	Register	1 Bit Flagge für Auswahl (GPR oder spezielle Register) 7 Bit für GPR
Immediate	1	16	Ganzzahliger Wert	

Registertabelle für 1+Bitmuster:

Selektorflagge	**Bitmuster**
1	R0...R127

Registertabelle für 0+Bitmuster:

Selektorflagge	**Bitmuster**	**Register**
0	0000000	PC
	0000001	IR
	0000010	O0
	0000011	O1
	0000100	O2
	0000101	EMDR[L]
	0000110	EMDR[H]
	0000111	EMAR[L]
	0001000	EMAR[H]
	0001001	EMDR
	0001010	EMAR
	0001011	MDR
	0001100	MAR

4.5 Befehlssatz

Legende

Quellwert (Source) unmittelbar, aus Register direkt/indirekt	S
Ergebniswert (Target) aus Register direkt/indirekt	T
Adresse (absolut)	A
Kanal (Channel)	C

Instruk/-tions/-code	**Mnemonic**	**Argumente (A B C)**	**Befehlsart**	**Funktion**
000000	MOV	T S	Transferiert T:=S (Move)	Transfer

000001	BLT	S1 S2 A	**B**ranch if S1 **l**esser **t**han S2	Sprung (nutzt nicht den Stack)
000010	BGT	S1 S2 A	**B**ranch if S1 **g**reater **t**han S2	
000011	BEQ	S1 S2 A	**B**ranch if S1 **eq**ual S2	
000100	BNQ	S1 S2 A	**B**ranch if S1 **n**ot **e**qual S2	
000101	BOF	A	**B**ranch if **o**ver**f**low	
000110	BUF	A	**B**ranch if **u**nder**f**low	
000111	BLE	S1 S2 A	**B**ranch if S1 **l**esser or **e**qual than S2	
001000	BGE	S1 S2 A	**B**ranch if S1 **g**reater or **e**qual than S2	
001001	BRA	A	**Bra**nch to A	
001010	JSR	A	**J**ump to **S**ub**r**outine A	Sprung (nutzt Stack)
001011	RET		Springt zurück (**Ret**urned) zur Adresse auf TOS (Top Of Stack)	
001011	RET	N	wie RET, entfernt aber N Elemente vom Stack	
001011	PUSH	S	Fügt auf den Stack hinzu	
001011	POP	T	Entfernt oberstes Element vom Stack	
001100	JLT	S1 S2 A	**J**ump if S1 **l**esser **t**han S2	
001101	JGT	S1 S2 A	**J**ump if S1 **g**reater **t**han S2	
001110	JEQ	S1 S2 A	**J**ump if S1 **eq**ual S2	
001111	JNQ	S1 S2 A	**J**ump if S1 not **eq**ual S2	
010000	JOF	A	**J**ump if **o**ver**f**low	
010001	JUF	A	**J**ump if **u**nder**f**low	
010010	JLE	S1 S2 A	**J**ump if S1 **l**esser or **e**qual than S2	
010011	JGE	S1 S2 A	**J**ump if S1 **g**reater or **e**qual than S2	
010100	INC	T S	Inkrementiert (**inc**rement)	Arithmetisch
010101	DEC	T S	Dekrementiert (**dec**rement)	
100001	SL	T S	logical left shift	
100010	ASL	T S	logical right shift	
100011	ASR	T S	arihthm. right shift (höchswertige bit unverändert)	
010110	ADD	T S S	**Add**ition	
010111	SUB	T S1 S2	**Sub**traktion S1-S2 = T	
010110	ADDC	T S S	**Add**ition mit Carry	

010111	SUBB	T S1 S2	**Sub**traktion S1-S2 = T mit Borrow	
011000	NOT	T S	Negiere S (**NOT**)	Logisch, bitweise
011001	OR	T S S	**OR**-Operation	
011010	AND	T S S	**AND**-Operation	
011011	XOR	T S S	**XOR**-Operation	
011011	CLC	T S S	clear carry	
011011	STC	T S S	STC set carry	
011011	CMC	T S S	CMC komplement carry	
011100	GET	T C	Empfängt Daten über Kanal C, speichert in T	
011101	SET	C S	Sendet Daten von S über Kanal C	
011110	MVLH	T S S	**M**o**v**e **l**ow **h**igh; konkateniert zwei 16-Bit-Adressen zu einer 32-Bit-Adresse zwecks Kommunikation mit 32-bittigen Geräten	Adressen und Daten nach Ziel T (EMAR oder EMDR)
011111	LOADLH	A	Lädt von Adresse A in den EMDR	
100000	STORELH	A	Speichert von EMDR an Adresse A	
100100	INIT	T	Initialisiert mit 0	
100100	CMP	S S	Vergleiche Operanden	
100100	CBRA	A1 A2 A3	(bedingt CMP) Springe zu A1, wenn kleiner, zu A2, wenn größer oder zu A3 wenn gleich (ST GT EQ), nutzt Stack	
100100	CJTS	A1 A2 A3	(bedingt CMP) Springe zu A1, wenn kleiner, zu A2, wenn größer oder zu A3 wenn gleich (ST GT EQ), nutzt nicht den Stack	
111111	Hardwareabsturz			

4.6 Datentypen- und Formate

Es werden ausschließlich Ganzzahlen benutzt. Die interne Repräsentation erfolgt im Zweierkomplement.

4.7 Beschreibung der Routinen

Die Kompressordruckregulation ist bei aktiviertem Kompressor aktiv und sorgt für einen gleichförmigen Innendruck. Tatsächlich läuft der aktivierte Kompressor nur bei Druckabfall an und deaktiviert sich bei gleichmäßigen Druck für eine Zeit von 2 Sekunden. Daher ist keine Routine zu diesem Zwecke vorgesehen.

Auch andere Funktionalitäten - wenn aktiviert - des Roboters laufen sensorbasiert ohne Interaktion mit dem Mikroprozessor ab, wie z.B. die automatische Wasserreinigung oder der Putzmechanismus.

Routine der Fortbewegung (hier beispielhaft: horizontale Bewegung nach rechts):

Bewegung	Pseudocode
	Solange Kanal 7/8 null ausgibt:
Saugnapfbewegung der diagonalen Saugnäpfe 1 der Schienen links oben und rechts unten	Sende Signal zum Loslösen der Saugnäpfe l.o. 1 (33), r.u. 1 (36) Kompressor der Saugnäpfe an und Saugnäpfe lösen Motor der Saugnäpfe an Verschieben der Saugnäpfe bis Sensoren der Saugnäpfe das Stoppsignal senden
Saugnapfbewegung der diagonalen Saugnäpfe 1 der Schienen rechts oben und links unten	Signal zum Loslösen der Saugnäpfe r.o. 1 (34), l.u. 1 (35) Kompressor der Saugnäpfe an Motor der Saugnäpfe an Verschieben der Saugnäpfe bis Sensoren der Saugnäpfe das Stoppsignal senden
Verschieben der Schienen nacheinander	Motor der jeweiligen Schiene an Verschieben der jeweiligen Schiene Stoppen des jeweiligen Motors
Saugnapfbewegung der diagonalen Saugnäpfe 2 der Schienen rechts oben und links unten	...analog
Saugnapfbewegung der diagonalen Saugnäpfe 2 der Schienen links oben und rechts unten	...analog
Beenden der Schleife	Kompressor aus

Routine des Putzvorganges

Vorgang	Pseudocode

	Solange Füllhöhe des Wassers und des Putzmittels ausreichend:
Putzmittel mischen	Putzmittelmischmotor an und entsprechend des Putzprogrammes mischen
Sprühvorgang und Wasserreinigung (automatisch)	Kompressor an (vorgegebene Zeit im Programm festgehalten) Gemisch wird aufgesprüht und aufgefangenes Wasser automatisch gereinigt und wieder zurückgepumpt, solange der Kompressor läuft Kompressor aus

4.8 Assemblercode der Routinen

Ein für unseren Mikroprozessor geeigneter Assemblercompiler erfordert einen Präprozessor, der die Sprungmarken zu Speicheradressen übersetzt und Namensräume beherrscht. Alle Sprungmarken in einem Namensraum sind standardmäßig lokal gebunden (**.namespace NSNAME**). Somit sind Sprungmarken im Systemkode immer von anderen Namensräumen (hier: Speicherbereichen) getrennt, z.B. ein Putzprogramm (hier: Sprungmarke) im Namensraum der Putzprogramme vom Systemcode. Mit dem Schlüsselwort **.public** wird der Zugriff auf Sprungmarken **NAME** erlaubt (z.B. **.public NAME**). Diese Voraussetzungen folgen dem Wunsche, Adressarithmetik komplett zu vermieden (leider fanden wir nicht die Zeit, die praktischen Probleme, die Entstehen, wenn Programmcode kopiert, die Adressen aber nicht relativ zur Speicherungsadresse korrigiert werden, zu lösen)

Eine weitere Anforderung an einen Assemblercompiler ist eine Spezialform zum Umgang mit dem Kommunikationsbus bei konditionalen Verzweigungen (Branches) und weiteren Operationen.

Die Direktive ←**N**, wobei **N** für eine Kanalzahl steht, wird zu **GET RX N** expandiert, wobei RX eines der reservierten Register R125. . . R127 sein kann, vor der Operation ausgeführt wird und ausschließlich Quelle sein kann. Ebenso wird die Direktive →N zu **SET N RX** expandiert, nach der Operation ausgeführt, und kann ausschließlich Ziel einer Operation sein. Der Vorgang STORE OPERANDS arbeitet offensichtlich mit den Verzweigungen und Sprüngen zusammen. Ein für unseren Mikroprozessor geeigneter Assemblercompiler besitzt einen Präprozessor, der die Sprungmarken zu Speicheradressen übersetzt.

Zur Verdeutlichung folgende Beispiele:

BEQ ←7 0 #NEXT	GET R127 7 BEQ R127 0 #NEXT	Branche zu #NEXT, wenn Sensorwert von Kanal 7 gleich 0 ist
BEQ 0 ←8 #NEXT	GET R127 7 BEQ 0 R127 #NEXT	Branche zu #NEXT, wenn Sensorwert von Kanal 8 gleich 0 ist
BEQ ←7 ←8 #NEXT	GET R127 7 GET R126 8 BEQ R127 R126 #NEXT	Branche zu #NEXT, wenn Sensorwerte von Kanal 7 gleich Sensorwerte von Kanal 8
ADD →20 #10 #101	ADD R127 #10 #101 SET 20 R127	Addiere 10 und 101, sende Ergebnis 111 über den Kanal 20
ADD →20 ←#10 ←#110	ADD R127 #10 #101 SET 20 R127	Addiere 10 und 101, sende Ergebnis 111 über den Kanal 20
ADD →20 ←7 ←8	GET R127 7 GET R126 8 AND R125 R126 R127 SET 20 R125	Sind Sensorwerte von Kanal 7 und Sensorwerte von Kanal 8 beide 1, sende 1 über den Kanal 20, ansonsten 0.

4.8.1 Assemblercode für den Startvorgang

Der Hauptspeicher ist beim Start des Roboters leer, d.h. nicht erfasst. Im Bootcode wird ein Teil des externen Speichers in den Hauptspeicher kopiert. Der Halt-Befehl schaltet das Gerät ab.

Assemblercode	Erläuterungen
.org 0 .namespace BOOT .public BOOTSTRAP	
:BOOTSTRAP MOV R0 0 MOV R1 0x8000	:Kopiere von Flashdisk nach Hauptspeicher den ersten Teil des Bootcodes
:BOOT_TEST_COND BGE R0 16383 BOOT_DONE MOV EMDR @R0 MOV @R1 EMDRL ADD R1 R1 1	:Kopiere Doppelworte in MAR von Adressbereich 0-16383 Schleife, breche bei Adressen größer gleich 16383 ab Lese Hauptspeicher Kopiere LOW aus EMDR nach Adresse in R1 nächste Adresse im Hauptspeicher A=A+1

MOV @R1 EMDRH	... behandle HIGH ebenso
ADD R0 R0 1	nächste Adresse; A=A+1 (Doppelwort)
BRA BOOT_TEST_COND	Schleifenbedingung prüfen
:BOOT_DONE	Gehe zurück zum letzten Aufruf
RET	

4.8.2 Assemblercode für die Systemschleife:

Assemblercode	
.org 200	Zugehörig zum Systemraum
.namespace SYSTEM	
:MAIN_LOOP	:Wiederhole
JTS SSYSTEM_LOOP	Führe Unterprogramme (nicht aufgeführt) aus
JTS SMOVEMENT_LOOP	
JTS SREMOTE_LOOP	
JTS SMOVEMENT_LOOP	
[...]	Weitere Unterprogramme
MOV R0 0x3000	(Ab 0x3000 befinden sich Variablen für das System)
BEQ @R0 1 SHUTDOWN	Prüfe, ob 0x3000 1 entspricht, dann fahre runter
BRA MAIN_LOOP	
:SHUTDOWN	:Beende
HALT	

4.8.3 Assemblercode für eine horizontale Bewegung vorwärts:

Anmerkung: Horizontale Bewegungen rückwärts oder vertikale funktionieren quasi analog und sind daher nicht weiter aufgeführt.

Assemblercode	Erläuterungen
.org 500	Zugehörig zum System
.namespace SYSTEM	Öffentliche Sprungmarken
.public MOVEH	
.public MOVEH_LOOP	
:MOVEH_LOOP	:Wenn Kanal 7/8 0 ausgibt, ...
JEQ ←7 0 MOVEH	
JEQ ←8 0 MOVEH	... gehe vorwärts horizontal schrittweise
RET	
:MOVEH	:gehe einen einzelnen Schritt vorwärts horizontal
SET 6 1	Kompressor an
JSR MOVEH_A0	

SET 6 RET	Kompressor aus
:MOVEH_A0 BEQ ←83 0 MOVEH_A1a BEQ ←95 0 MOVEH_A1b BRA MOVEH_B0	Kompressor aus . . . wiederhole Schleife
:MOVEH_A1a SET 59 1 SET 33 1 BRA MOVEH_A0 :MOVEH_A1b SET 62 1 SET 36 1/ BRA MOVEH_A0	:Signal zum Loslösen der Saugnäpfe l.o. 1 (33), r.u. 1 (36) Saugnapfklappen öffnen Verschieben der Saugnäpfe
:MOVEH_B0 SET 59 0 SET 62 0 BEQ ←87 0 MOVEH_B1a BEQ ←91 0 MOVEH_B1b BRA MOVEH_C0	:solange nicht Endposition Saugnapfklappen wieder schließen . . . wiederhole Schleife
:MOVEH_B1a SET 60 1 SET 34 1 BRA MOVEH_B0 :MOVEH_B1b SET 61 1 SET 35 1 BRA MOVEH_B0	:Signal zum Loslösen der Saugnäpfe r.o. 1 (34), l.u. 1 (35) Saugnapfklappen öffnen Verschieben der Saugnäpfe
:MOVEH_C0 SET 60 0 SET 61 0 GET R0 16 BEQ R0 11 MOVEH_C0T BNQ R0 11 MOVEH_C1 SET 25 1 BRA MOVEH_C1	:bewege zuständige Schiene Saugnapfklappen wieder schließen Position zu Ende → stoppe Schiene Position nicht zu Ende → w. m. mit nächster Schiene Schalte Motor ein
:MOVEH_C0T SET 25 0 BRA MOVEH_C1	:stoppe zuständige Schiene Schalte Schienenmotor aus
:MOVEH_C1 GET R0 17 BEQ R0 11 MOVEH_C1T	:bewege zuständige Schiene Position zu Ende → stoppe Schiene

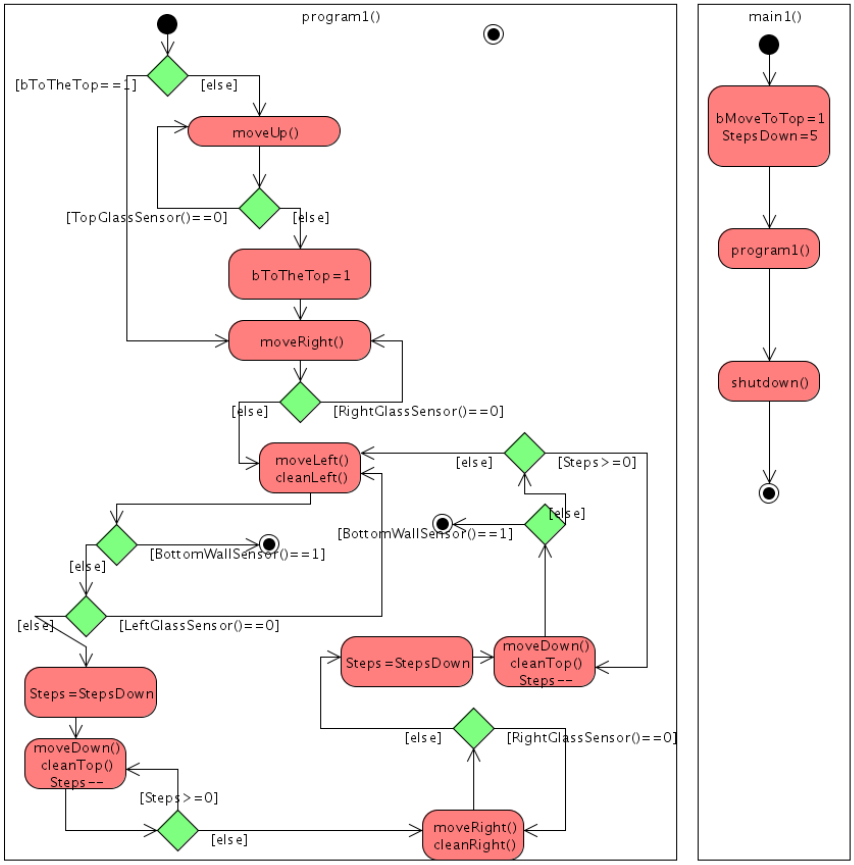
BNQ R0 11 MOVEH_C2	Position nicht zu Ende → w. m. mit nächster Schiene
SET 26 1	Schalte Motor ein
BRA MOVEH_C2	
:MOVEH_C1T	:stoppe zuständige Schiene
SET 26 0	Schalte Schienenmotor aus
BRA MOVEH_C2	
:MOVEH_C2	:bewege zuständige Schiene
GET R0 18	
BEQ R0 11 MOVEH_C2T	Position zu Ende → stoppe Schiene
BNQ R0 11 MOVEH_C3	Position nicht zu Ende → w. m. mit nächster Schiene
SET 27 1	Schalte Motor ein
BRA MOVEH_C3	
:MOVEH_C2T	:stoppe zuständige Schiene
SET 27 0	Schalte Schienenmotor aus
BRA MOVEH_C3	
:MOVEH_C3	:bewege zuständige Schiene
GET R0 19	
BEQ R0 11 MOVEH_C3T	Position zu Ende → stoppe Schiene
BNQ R0 11 MOVEH_C4	Position nicht zu Ende → w. m. mit nächster Schiene
SET 28 1	Schalte Motor ein
BRA MOVEH_C4	
:MOVEH_C3T	:stoppe zuständige Schiene
SET 27 0	Schalte Schienenmotor aus
BRA MOVEH_C3	
:MOVEH_C4	:Und-Verknüpfung
GET R0 16	Teste alle Schienen, ob zu Ende und stoppe ggf. Motor
BNQ R0 11 MOVEH_C0	
GET R0 17	
BNQ R0 11 MOVEH_C1	
GET R0 18	fahre fort
BNQ R0 11 MOVEH_C2	
GET R0 19	
BNQ R0 11 MOVEH_C3	
BRA MOVEH_D0	
:MOVEH_D0	:solange nicht Endposition
BEQ ←85 0 MOVEH_D1a	
BEQ ←96 0 MOVEH_D1b	... wiederhole Schleife
BRA MOVEH_E0	
:MOVEH_D1a	:Signal zum Loslösen der Saugnäpfe l.o. 2 (37), r.u. 2 (40)
SET 63 1	Saugnapfklappen öffnen

SET 37 1 BRA MOVEH_D0 :MOVEH_D1b SET 66 1 SET 40 1 BRA MOVEH_D0	Verschieben der Saugnäpfe
:MOVEH_E0 SET 63 0 SET 66 0 BEQ ←89 0 MOVEH_E1a BEQ ←93 0 MOVEH_E1b BRA MOVEH_F0	:solange nicht Endposition Saugnapfklappen wieder schließen ...wiederhole Schleife
:MOVEH_E1a SET 65 1 SET 38 1 BRA MOVEH_E0 :MOVEH_E1b SET 64 1 SET 39 1 BRA MOVEH_E0	:Signal zum Loslösen der Saugnäpfe r.o. 2 (38), l.u. 2 (39) Saugnapfklappen öffnen Verschieben der Saugnäpfe
:MOVEH_F0 SET 65 0 SET 64 0 RET	:Ende Saugnapfklappen wieder schließen

4.9 Putzprogramme

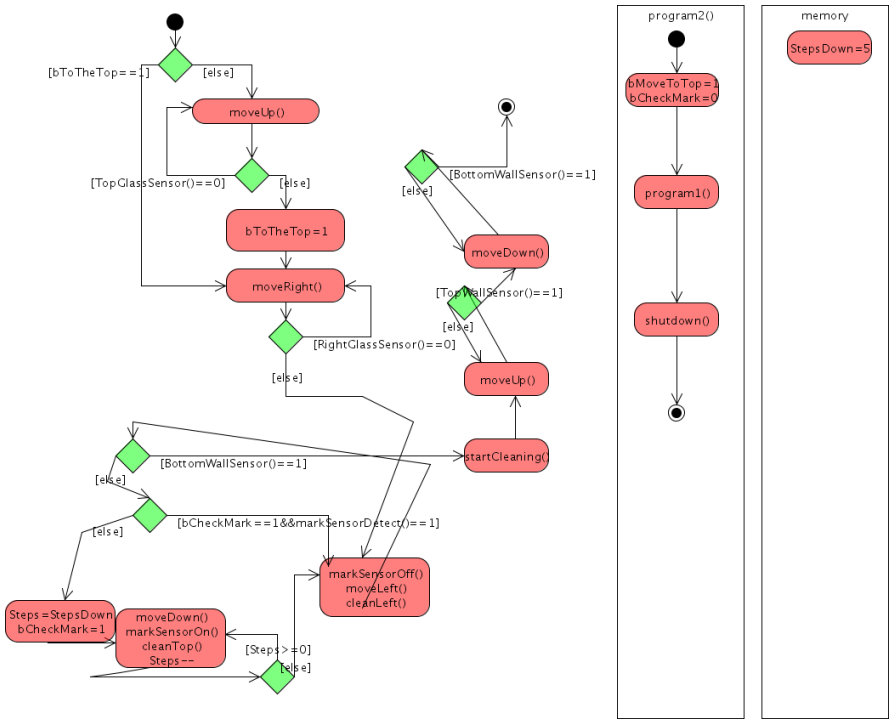
4.9.1 Programm1(): Ebene Flächen

Das Gerät wird irgendwo an der Fläche gestartet, fährt zum oberen Wandende, justiert sich nach rechts (bewegt sich nach rechts), beginnt dann mäandernd zu Säubern, von rechts nach links, Versetzung nach unten, links nach rechts, Versetzung nach unten, so lange wiederholend, bis am Boden angekommen.



4.9.2 Programm2(): Zylindrische, ebene Flächen

Ähnlich zu Programm1() wird sich bewegt, aber ein Markierung jeweils beim Versetzen nach unten angebracht und sich ausschließlich von links nach rechts bewegt. Bei dem ersten halben Durchgang des Mäanderschema wird sofort markiert, die folgenden Male wird bei der Fortbewegung fortwährend geprüft, ob Markierung am Sensor. Wenn ja, versetzt das Gerät nach unten und wiederholt das Schema. Am Ende, am Boden angekommen, fährt das Gerät an der Markierung hoch, sie dabei entfernend, und fährt nach unten.



Assemblercode

4.10 Steuereinheit

Erklärung der einzelnen Signale:

Signal	Bedeutung
PC_{in}	Liest die Daten vom Datenbus und schreibt sie in den PC
PC_{out}	Liest den PC und schreibt die Daten auf den Datenbus
CCR_{in}	Liest die Daten vom Bus und schreibt sie in das CCR
CCR_{out}	Liest die Daten aus dem CCR und legt sie auf den Bus
MAR_{in}	Liest die Daten vom Datenbus und schreibt sie in das MAR
MDR_{out}	Holt Daten aus dem MDR und legt sie auf den Datenbus
MDR_{in}	Schreibt die Daten vom Datenbus ins MDR
$EMAR[H]_{in}$	Schreibt den High-Teil der Daten in das EMAR
$EMAR[L]_{in}$	Schreibt den Low-Teil der Daten in das EMAR
$EMDR[H]_{in}$	Schreibt den High-Teil der Daten in das EMDR
$EMDR[L]_{in}$	Schreibt den Low-Teil der Daten in das EMDR
$EMDR_{out}$	Holt die Daten aus dem EMDR und legt sie auf den Datenbus

ALU_{in}	Übergibt die Daten des Busses der ALU
Y_{in}	Übergibt die Daten des Busses dem Zwischenspeicher der ALU
Z_{out}	Legt das Ergebnis aus dem Zwischenspeicher Z auf den Datenbus
WMFC	Warten, bis memory fetch complete
MEM_{read}	Liest den Hauptspeicher
MEM_{write}	Schreibt den Hauptspeicher
$EMEM_{read}$	Liest den Sekundärspeicher
$EMEM_{write}$	Schreibt den Sekundärspeicher
IR_{in}	Liest die Daten vom Datenbus und schreibt sie in das IR
KS_{out}	Liest die Daten aus der Kommunikationsschnittstelle und schreibt sie auf den Bus
KS_{in}	Liest die Daten vom Bus und gibt sie der Kommunikationsschnittstelle
OP_{in}	Liest Daten aus angegebenen Operanden (Register etc.) und legt sie auf den Bus
OP_{out}	Liest Daten vom Bus und schreibt sie in den Operand

STORE OPERANDS

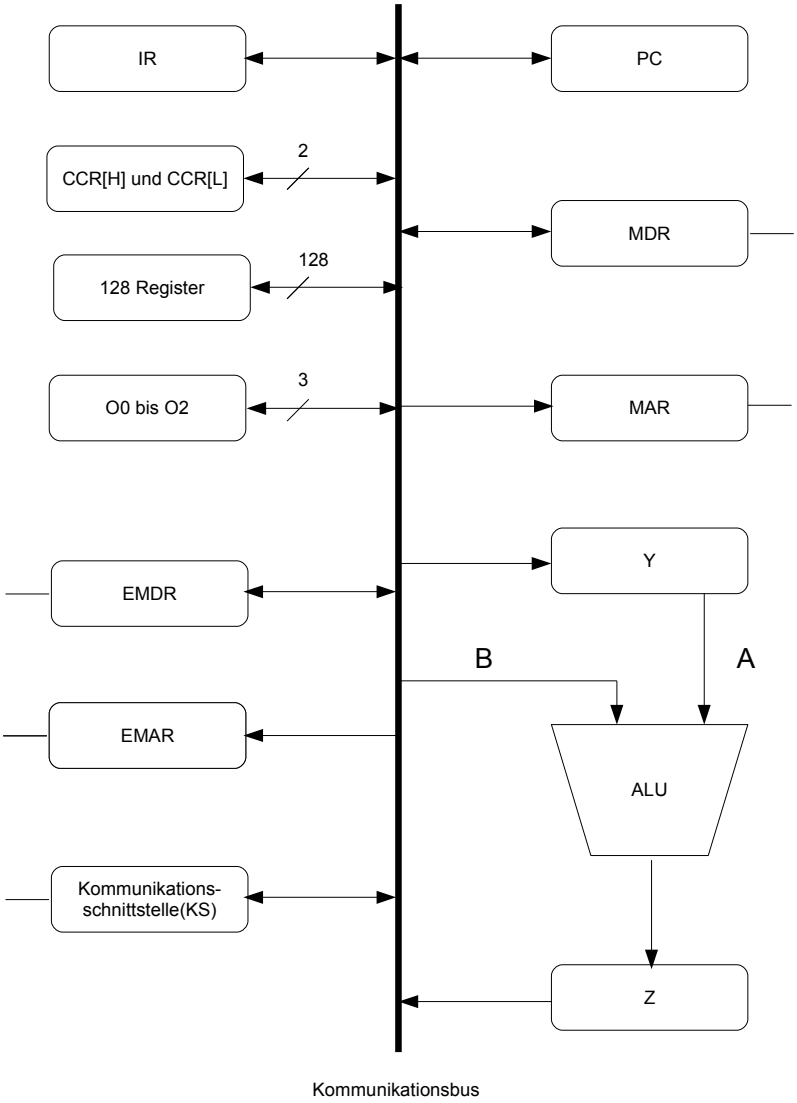
Der Wert der Operanden wird unaufgefordert in die Operandenregister O0...O2 (O0 erster, O1 zweiter, O3 dritter Operand) gespeichert. Bei Operationen mit weniger Operanden wird 0 stattdessen im jeweiligen Operandenregister gespeichert. Dies erlaubt nach Branches nachträglichen Zugriff auf die beim Branch ausgewerteten Operanden.

OPCODE FETCH (vor jedem Befehl)	PC_{out}, MAR_{in}, ALU_{in}, F=B+1, Z_{in}, MEM_{read} Z_{out}, PC_{in} IR_{in}, MDR_{out}, DECODE, WMFC
STORE OPERANDS	$OP1_{out}$, $O0_{in}$ $OP2_{out}$, $O1_{in}$ $OP3_{out}$, $O2_{in}$
MOV OP1, OP2	$OP2_{out}$, $OP1_{in}$
INC OP1, OP2	$OP2_{out}$, ALU_{in}, F=B+1 Z_{out}, $OP1_{in}$
DEC OP1, OP2	$OP2_{out}$, ALU_{in}, F=B-1 Z_{out}, $OP1_{in}$
ADD OP1, OP2, OP3	$OP2_{out}$, Y_{in} $OP3_{out}$, ALU_{in}, F=A+B Z_{out}, $OP1_{in}$
SUB OP1, OP2, OP3	$OP2_{out}$, Y_{in} $OP3_{out}$, ALU_{in}, F=A-B Z_{out}, $OP1_{in}$
NOT OP1, OP2	$OP2_{out}$, ALU_{in}, F = 1st complement Z_{out}, $OP1_{in}$

OR OP1, OP2, OP3	$OP2_{out}$, Y_{in} $OP3_{out}$, ALU_{in}, F=A OR B Z_{out}, $OP1_{in}$
AND OP1, OP2, OP3	$OP2_{out}$, Y_{in} $OP3_{out}$, ALU_{in}, F=A AND B Z_{out}, $OP1_{in}$
XOR OP1, OP2, OP3	$OP2_{out}$, Y_{in} $OP3_{out}$, ALU_{in}, F=A XOR B Z_{out}, $OP1_{in}$
GET OP1, C	KS_{out}, $OP1_{in}$
SET C, OP1	$OP1_{out}$, KS_{in}
BRA OP1	$OP1_{out}$, PC_{in}
BLT OP1, OP2, OP3	$OP1_{out}$, Y_{in} $OP2_{out}$, ALU_{in}, F=B > A bei Z=1: $OP3_{out}$, PC_{in} bei Z=0: run
BGT OP1, OP2, OP3	$OP1_{out}$, Y_{in} $OP2_{out}$, ALU_{in}, F=A > B bei Z=1: $OP3_{out}$, PC_{in} bei Z=0: run
BEQ OP1, OP2, OP3	$OP1_{out}$, Y_{in} $OP2_{out}$, ALU_{in}, F=A == B bei Z=1: $OP3_{out}$, PC_{in} bei Z=0: run
BNQ OP1, OP2, OP3	$OP1_{out}$, Y_{in} $OP2_{out}$, ALU_{in}, F=A /= B bei Z=1: $OP3_{out}$, PC_{in} bei Z=0: run
BGE OP1, OP2, OP3	$OP1_{out}$, Y_{in} $OP2_{out}$, ALU_{in}, F=A >= B bei Z=1: $OP3_{out}$, PC_{in} bei Z=0: run
BLE OP1, OP2, OP3	$OP1_{out}$, Y_{in} $OP2_{out}$, ALU_{in}, F=A >= B bei Z=1: $OP3_{out}$, PC_{in} bei Z=0: run
BOF OP1	CCR_{out}, ALU_{in}, F= B == 1 bei Z=1: $OP1_{out}$, PC_{in} bei Z=0: run
BUF OP1	CCR_{out}, ALU_{in}, F= B == 0 bei Z=1: $OP1_{out}$, PC_{in} bei Z=0: run
JSR OP1 JLT OP1, OP2, OP3	$OP1_{out}$, PC_{in}, $R121_{in}$ $OP1_{out}$, Y_{in}

	$OP2_{out}$, ALU_{in}, F=B > A bei Z=1: $OP3_{out}$, PC_{in}, $R121_{in}$ bei Z=0: run
JGT OP1, OP2, OP3	$OP1_{out}$, Y_{in} $OP2_{out}$, ALU_{in}, F=A > B bei Z=1: $OP3_{out}$, PC_{in}, $R121_{in}$ bei Z=0: run
JEQ OP1, OP2, OP3	$OP1_{out}$, Y_{in} $OP2_{out}$, ALU_{in}, F=A == B bei Z=1: $OP3_{out}$, PC_{in}, $R121_{in}$ bei Z=0: run
JNQ OP1, OP2, OP3	$OP1_{out}$, Y_{in} $OP2_{out}$, ALU_{in}, F=A /= B bei Z=1: $OP3_{out}$, PC_{in}, $R121_{in}$ bei Z=0: run
JOF OP1	CCR_{out}, ALU_{in}, F= B == 1 bei Z=1: $OP1_{out}$, PC_{in}, $R121_{in}$ bei Z=0: run
JUF OP1	CCR_{out}, ALU_{in}, F= B == 0 bei Z=1: $OP1_{out}$, PC_{in}, $R121_{in}$ bei Z=0: run
JLE OP1, OP2, OP3	$OP1_{out}$, Y_{in} $OP2_{out}$, ALU_{in}, F=A >= B bei Z=1: $OP3_{out}$, PC_{in}, $R121_{in}$ bei Z=0: run
JGE OP1, OP2, OP3	$OP1_{out}$, Y_{in} $OP2_{out}$, ALU_{in}, F=A >= B bei Z=1: $OP3_{out}$, PC_{in}, $R121_{in}$ bei Z=0: run
MVLH OP1 OP2 OP3	$OP2_{out}$, $OP1[L]_{in}$ $OP3_{out}$, $OP1[H]_{in}$
RET	$R121_{out}$, PC_{in}
LOADLH OP1	$OP1[H]_{out}$, $EMAR[H]_{in}$ $OP1[L]_{out}$, $EMAR[L]_{in}$, $EMEM_{read}$,WMFC
STORELH OP1	$EMAR[H]_{out}$, $OP1[H]_{in}$ $EMAR[L]_{out}$, $OP1[L]_{in}$, $EMEM_{write}$,WMFC

4.11 Schaltungsentwurf



5 Erweiterungen: Rückruffunktion

5.1 Inhalt

Die Rückruffunktion ist in einer Fernsteuerung implementiert. Diese Fernsteuerung kann jedoch über die initiale Idee mehr – wie Putzprogramme starten, den Wasser-, Putzmittel- oder Akkustand anzeigen.

Die Möglichkeit, das Funkprotokoll anstelle der Host-CPU in einer eigenen dezidierten CPU abarbeiten zu lassen, wurde nicht betrachtet. Ebenso wie die parallele Ausführung der Sende/Empfangseinheit zum Programm des Roboters.

5.2 Datenformat

Die Wortbreite beträgt immer 18 Bit.

Im 1. Wort wird eine Prüfsumme über die GeräteID, den Instruktionscode und die Multiwortflagge errechnet. Ist die Multiwortflagge 1, so folgt ein weiteres Wort.

		ECC über		
	ECC	GeräteID	Instruktionscode	Multiwortflagge
Bit(s)	5	8	4	1

Im Multiwortmodus wird desweiteren mindestens ein Datenwort übertragen. Die Multiwortflagge kündigt auch hier ein weiteres Wort an. Jeder Befehl (aus mehren Worten) endet mit einer gesetzten Multiwortflagge. Daten werden von links mit Nullen gefüllt, falls der Wertebereich von 2^(12) nicht ausgenutzt wird.

		ECC über	
	ECC	Daten	Multiwortflagge
Bit(s)	5	12	1

5.3 Gemeinsamer Befehlscode: Roboter und Fernbedienung

Instruktionscode	Mnemonic	Funktion	Multiwort (nein)	Empfängeraktion
0000	F_ACK	Nachricht erhalten		Nichts
0001	FREQ_TIMEOUT	Nachricht erwartet – nicht erhalten.		Fehler anzeigen
0010	FREQ_ECC_ERROR	Nachricht erhalten – falsche Prüfsumme.		Fehler anzeigen, starte letzte Operation erneut
0011	FREQ_ERROR	Spezifische Nachricht erwartet, andere Nachricht erhalten oder Übertragung fehlgeschlagen, erwarte keine weiteren Versuche, Operation durchzuführen		Fehler anzeigen, Systemreset

Bedeutung der Aktionen

Fehler anzeigen: wie FRES_OP_FAILED behandeln, d.h. Fehler durch LED an Gerät und Fernbedienung signalisieren

Systemreset: Reset des Fernbedienungssystems

5.4 Befehlscode: Roboter

Instruktionscode	Mnemonic	Funktion	Multiwort (nein)
1011	FRES_DEV_ID	Sendet Gerätekennung	Ja, 1
1100	FRES_FILL_STAT	Sendet Füll-/Ladestandinformationen	Ja, 3
1101	FRES_OP_STARTED	Operation gestartet	
1110	FRES_OP_FAILED	Operation fehlgeschlagen	
1111	FRES_OP_FINISHED	Operation beendet	

5.4.1 Datenworte im Multiwortmodus:

FRES_DEV_ID:

Wort	ECC	Daten (8 Bits)	Multiwortflagge (1 Bit)
1.	n.a.	Geräteinterne ID	0

FRES_FILL_STAT:

Wort	ECC	Daten (12 Bits)	Multiwortflagge (1 Bit)
1.	n.a.	Akkuladung	1
2.	n.a.	Wasserfüllstatus	1
3.	n.a.	Spülmittelfüllstatus	0

Die Daten sind Zahlen in einem Bereich von 0. . . 2519 (100% entspricht 2520, eine besonders hochzusammengesetzte Zahl).

5.5 Befehlscode: Fernsteuerung

Opcode	Mnemonic	Funktion	Multiwort (nein)	Implizites F_ACK erwartet (nein)
0100	FREQ_MVR	Deaktiviere automatischen Putzmodus, bewege rechts		
0101	FREQ_MVL	s.o., bewege links		
0110	FREQ_MVF	s.o., bewege vorwärts		
0111	FREQ_MVB	s.o., bewege rückwärts		
1000	FREQ_-START_-CLEANING	Aktiviere automatischen Putzmodus		
1001	FREQ_STOP	Falls in Bewegung oder im automatischen Putzmodus, stoppe		
1010	FREQ_-SHUTDOWN	Stoppe, bewege Gerät zum Boden		
1011	FREQ_-START_-PROGRAM	Starte Programm (für automatischen Putzmodus)	ja	
1100	FREQ_-ENTER_-DEBUG_-MODE	Aktiviere Debugmodus		
1101	FREQ_-LEAVE_-DEBUG_-MODE	Deaktiviere Debugmodus		
1110	FREQ_FILL_-STATUS	Verlange Füllstatus		Ja, FRES_FILL_STAT

1111	FREQ_-ESTABLISH_-CONNECTION	Baue Verbindung auf		Ja, FRES_DEV_ID

5.5.1 Datenworte im Multiwortmodus:

FREQ_START_PROGRAM:

Wort	ECC	Daten (8 Bits)	Multiwortflagge (1 Bit)
1.	n.a.	Programmnummer	0

5.6 Fehlererkennung

Es wird ein verkürzter Hamming(18,13)-Code benutzt, der suboptimal im Vergleich zu Hamming(32,27), der das beste Verhältnis von Paritäts- und Datenbits aufweist, ist. E1...E5 bezeichnet die ECC-Informationsstellen, D1...D13 die Datenstellen; $\oplus$ steht für eine XOR-Operation.

Nachrichtenformat:

D14...D27(bzw. C19...C32) sind offensichtlich immer 0, weil sie nicht genutzt werden. Dies ist der Grund für die Codewortverkürzung.

C1	C2	C3	C4	C5	C6	C7	C8	C9
E1	E2	D1	E3	D2	D3	D4	E4	D5
C10	C11	C12	C13	C14	C15	C16	C17	C18
D6	D7	D8	D9	D10	D11	E5	D12	D13

Da die Reihenfolge für die Abarbeitung der Verknüpfung von mehreren $(0 \oplus 0)$-Operationen nicht relevant ist und weiterhin $(0 \oplus A)=(A \oplus 0)=A$ gilt, werden im folgenden D14...D27 nicht mehr verwendet (Codewortverkürzung).

Vor dem Senden wird die Parität wie folgt errechnet:

$E1=D1 \oplus D2 \oplus D4 \oplus D5 \oplus D7 \oplus D9 \oplus D11 \oplus D12$

$E2=D1 \oplus D3 \oplus D4 \oplus D6 \oplus D7 \oplus D10 \oplus D11 \oplus D13$

$E3=D2 \oplus D3 \oplus D4 \oplus D8 \oplus D9 \oplus D10 \oplus D11$

$E4=D5 \oplus D6 \oplus D7 \oplus D8 \oplus D9 \oplus D10 \oplus D11$

$E5=D12 \oplus D13$

Der Empfänger überprüft die Parität erneut wie folgt:

(E1:) $C1=C3 \oplus C5 \oplus C7 \oplus C9 \oplus C11 \oplus C13 \oplus C15 \oplus C17$

(E2:) $C2=C3 \oplus C6 \oplus C7 \oplus C10 \oplus C11 \oplus C14 \oplus C15 \oplus C18$

(E3:) C4=C5 $\oplus$ C6 $\oplus$ C7 $\oplus$ C12 $\oplus$ C13 $\oplus$ C14 $\oplus$ C15

(E4:) C8=C9 $\oplus$ C10 $\oplus$ C11 $\oplus$ C12 $\oplus$ C13 $\oplus$ C14 $\oplus$ C15

(E5:) C16=C17 $\oplus$ C18

Dabei gilt stets:

Jede Parität E0...E5 ist korrekt.	Kein Fehler oder Mehrbitfehler (>2)
Paritäten aus E0...E5 sind falsch.	Datenfehler (mindestens zwei Paritäten) oder Paritätsdatenfehler (bei genau einer Parität)

5.7 Kommunikation

Jeder Anfrage (**Req**uest) erwartet eine Antwortbestätigung (**Res**ponse) F_ACK (**Ack**nowledge) vom Empfänger. Anfragen, die Daten zurückliefern, senden stattdessen implizite ACK.

Fernbedienung	Roboter	Funktion
FREQ_XXX	→	
FREQ_TIMEOUT (optional)	→	Im Zeitfenster wurde nicht geantwortet.
	← 1.FRES_XXX ← 2.F_ACK ← 3.F_ECC_ERR ← 4.FRES_ERROR	1.(implizites F_ACK) 2.(explizites F_ACK) 3.Fehlerhafte Übertragung 4.Anderweitiger Fehler
1.F_ACK 2.F_ACK 3./ 4./	→ → → →	1.implizites F_ACK 2.implizites F_ACK 3.beginne von vorne N mal,N>0,sonst 4. 4.behandle wie FRES_OP_FAILED
	← 1./ ← 2./	1.Bereit für nächsten Befehl 2.Bereit für nächsten Befehl

Beispiel: Verbindungsaufbau

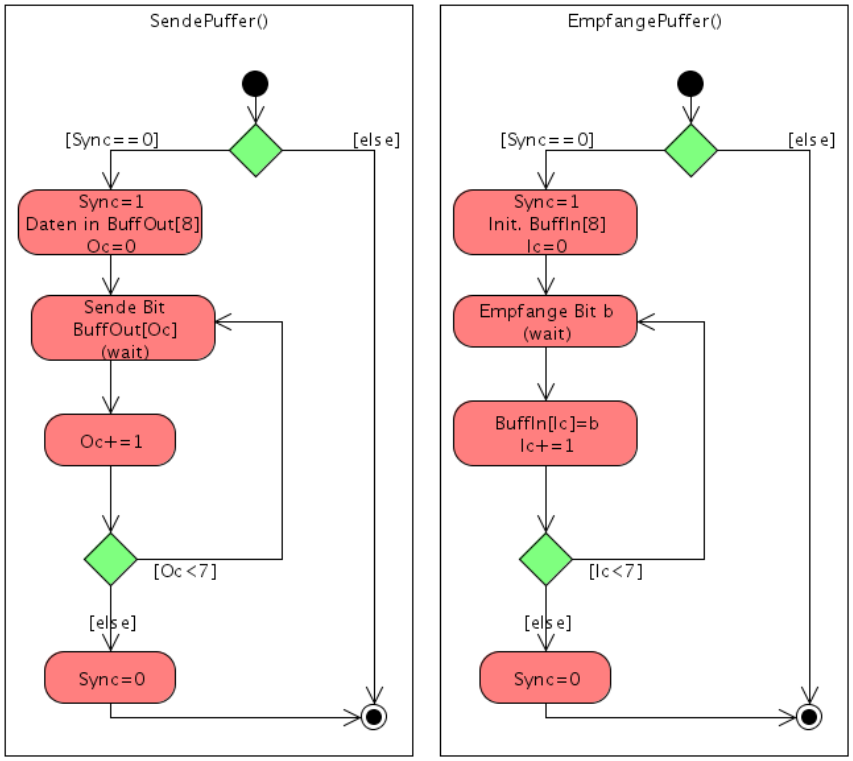
Fernbedienung	Roboter	Funktion
FREQ_ESTAB-LISH_-CONNECTION	→	Baue Verbindung auf
FREQ_TIMEOUT (optional)	→	Im Zeitfenster wurde nicht geantwortet.
	← 1.FRES_DEV_ID	1.aufgebaut, sende Geräte-ID (implizites F_ACK)
	← 2.FREQ_ECC_ERROR	2.ECC-Fehler
	← 3.FREQ_ERROR	3.Anderweitiger Fehler
1.F_ACK	→	1.implizites F_ACK
2./	→	2.beginne von vorne N mal,N>0,sonst 4.
3./	→	3.behandle wie FRES_OP_FAILED
	← 1./	1.Bereit für nächsten Befehl

Beispiel: Füllstatus (Verbindung besteht bereits)

Fernbedienung	Roboter	Funktion
FREQ_FILL_-STATUS	→	Erwarte Status
FREQ_TIMEOUT (optional)	→	Im Zeitfenster wurde nicht geantwortet.
	← 1.FRES_FILL_STAT	1.Sende 4 Datenworte (implizites F_ACK)
	← 2.FREQ_ECC_ERROR	2.ECC-Fehler
	← 3.FREQ_ERROR	3.Anderweitiger Fehler
1.F_ACK	→	1.implizites F_ACK
2./	→	2.beginne von vorne N mal,N>0,sonst 4.
3./	→	3.behandle wie FRES_OP_FAILED

5.8 Darstellung der Routinen

Daten aus Puffer senden / Daten aus Puffer empfangen:



5.9 Funknetz

Die Übertragung der Daten erfolgt drahtlos. Inwiefern sich die vorangegangene Ausführungen in bereits vorhandene Funkübertragungstechnologien integrieren lassen oder eine eigene Technik auf einem bestimmten Frequenzband vorzuziehen wäre, wird aus Platz- und Zeitgründen nicht weiter erörtert.

5.10 Aussehen der Fernbedienung

Um die Benutzung des Gerätes zu erleichtern, kann man unser Gerät per Fernbedienung steuern. So kann man bequem vom Boden aus die Putzprogramme eingeben oder wichtige Informationen über die verschiedenen Füllzustände erlangen.
Die Fernbedienung ist natürlich Wasserfest, damit sie bei jeder Wetterlage benutzt werden kann.Sie hat ein durchsichtiges Alphanumerisches Display, das Zahlen und Wörter ausgeben kann. Des weiteren gibt es ein Liquid Crystal Display, einen Funksender-/empfänger Adapter und LED beleuchtete Tasten. Die Fernbedienung wird mit einem Akkumulator der Stromstärke 4,8V betrieben. Um den Stromverbrauch zu senken, gibt das Display Farben in Graustufen aus.
Durch die integrierte Software gibt es verschiedene Ausgabemöglichkeiten auf der Fernbedienung. Dazu zählen zwei verschiedene Optionen des Ausschaltmoduses, die man mit „OK“ Bestätigen kann, eine kurze Beschreibung der Programme bei der Programmauswahl und die Rückgabe der Füllzustände mittels des Info-Knopfes auf der Fernbedienung.

Natürlich kann man ebenso das Gerät mittels dieser Fernbedienung steuern.

Die weiteren Tasten 1-9 benötigt man, um das gewünschte Programm aus dem Menü auch aus der Ferne auszuwählen.

Um die Bedienung weiterhin zu erleichtern und die Sicherheit zu verbessern, gibt es in der Fernbedienung einen Vibrationsalarm, der ausgelöst wird, wenn einer der Füllstände Wasser, Putzmittel oder Batterie unter 8% sinkt. Zusätzlich blinkt eine Lampe in der Fernbedienung auf.

Eine weitere Funktion der Fernbedienung ist die Möglichkeit des Abschaltens des Gerätes bei schlechtem Wetter oder ähnliches. Dazu gibt es zwei Varianten; entweder das Gerät schaltet sich einfach ab und bleibt an der Scheibe gesaugt oder es kommt zurück zur Startposition und schaltet sich dort automatisch ab.

Nichts desto trotz gibt es natürlich auch die vorinstallierte Möglichkeit des Gerätes, selbst zur Startposition zurückzukehren, falls man einmal die Fernbedienung verlegt hat.

6 Schlussbemerkungen

6.1 Bewertung der Einsatzmöglichkeiten

Putzi wurde entwickelt, um an großen ebenen Scheibenflächen zu putzen. Bürohäuser oder verglaste Hotels, wie sie immer häufiger entstehen bieten da eine gute Einsatzmöglichkeit. Man könnte andenken, ob es eventuell möglich sein wird, Putzi auch auf anderen ebenen Flächen wie Kunststoff oder polierten Steinen putzen zu lassen, wenn man die Sensoren dementsprechend erweitert. Auch kann man das Gerät zu Hause in Eigennutzung betreiben, wobei man da Aufwand und Nutzen abwägen sollte.

6.2 Bewertung der Einschränkungen

Physikalische Betrachtungen spielen bei diesem Projekt eine große Rolle. Ohne die Aufarbeitung von den in der Einleitung erwähnten notwendigen Betrachtungen (Gewicht, Tragfähigkeit der Scheiben, Aerodynamik, Windlast, Energieverbrauch,Putzleistung, finanzielle Prüfung) ist beispielsweise nicht daran zu denken, von einer Bank die zum Bau dieses Roboters erforderlichen Geldsummen zu erhalten. Der Roboter ist derzeit nicht praxistauglich. Durch weitere Forschung mag dieser Stolperstein ausgemerzt werden.

6.3 problemabhängige Sicherheits- bzw. Fehlerbetrachtungen

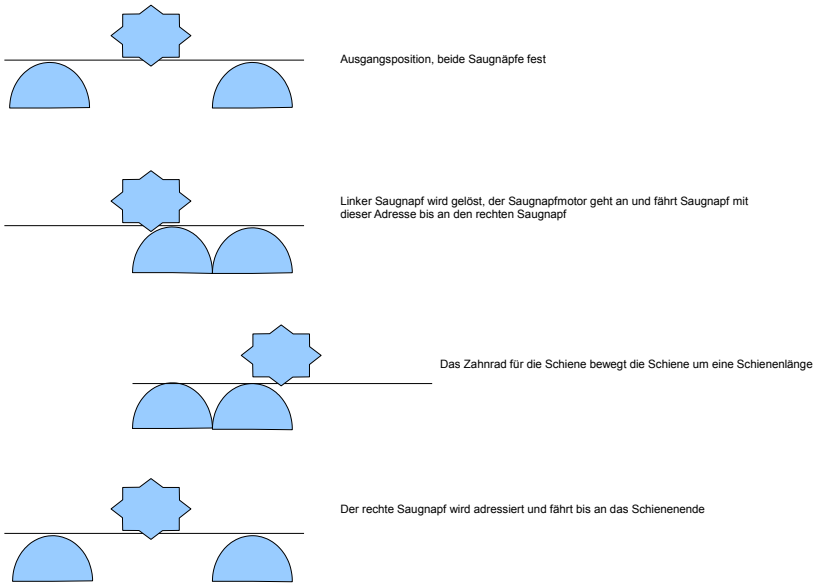
Das offensichtlichste Sicherheitsproblem dieses Projektes ist natürlich die Fixierung. Wie bei den Einschränkungen erwähnt, müssen diverse physikalische Gegebenheiten berücksichtigt werden. Wie schwer darf das Gerät inklusive Wasserzusatzlast werden, um von den Saugnäpfen gehalten zu werden? Man muss den Wind betrachten, die Angriffspunkte, die Belastbarkeit der Scheiben etc. All dieses machen das Gerät momentan noch zu einer Gefahrenquelle für sämtliche Personen, die sich in Reichweite des möglichen Abfallens befinden.

Wir haben auch keine technische Analyse durchgeführt, das heißt, dass es sein kann, dass dieses Gerät aufgrund von technischen Gegebenheiten gar nicht wie angedacht gebaut werden könnte oder wie oben erwähnt einfach viel zu schwer wird.

Des weiteren wissen wir natürlich nicht ob die angedachte Reinigungsweise mit Wasserdruck ein befriedigendes Ergebnis bei verschmutzten Scheiben liefert. All das müsste ausgiebig getestet und evtl. verbessert werden.

6.4 Timingbedingungen bei zeitkritischen Abläufen

Der einzige zeitkritische Ablauf in unserem Gerät ist die Fortbewegung. Denn die Stabilität und die Festigkeit muss in jeder Situation gewährleistet sein, es dürfen sich immer nur so viele Saugnäpfe von der Scheibe lösen, sodass die Maschine noch sicher an der Scheibe befestigt ist. Wir haben vier Schienen mit je zwei Saugnäpfen für die Bewegung in horizontale Richtung und ebenso viele für die vertikale Richtung. Um die Stabilität zu gewährleisten, befinden sich in jeder Position immer sechs Saugnäpfe an der Scheibe. Es werden immer nur zwei diagonal liegende Saugnäpfe parallel bewegt, während die anderen sechs Saugnäpfe befestigt bleiben. Bei der horizontalen bzw. vertikalen Bewegung gibt es vier Zustände einer Schiene, die nacheinander eingenommen werden, um die Schiene zu versetzen. Nacheinander mit den Diagonal liegenden Schienen ausgeführt, ergibt sich die Fortbewegung des Gerätes um eine Schienenlänge. Diese Zustände sind:



Beim dem Wechsel von horizontaler in vertikaler Richtung bzw. vertikaler in horizontaler Richtung gibt es einen weiteren Zustand. Es werden mit einem Mal alle 16 Saugnäpfe an der Scheibe befestigt und die nicht benötigten acht Saugnäpfe in die entgegengesetzte Richtung daraufhin gelöst, sodass nur noch acht an der Scheibe sind. Die genaue Beschreibung findet sich in Abschnitt 4.7 wieder.

6.5 Fehlerbetrachtungen und Tests

Das Modul müsste entsprechend unter extremen Bedingungen auf seine Praktikabilität hin geprüft werden. Windgeschwindigkeiten könnten Beispielsweise eine Barriere zu korrekter Funktionalität sein. Die Form dieser Maschine sollte maximal aerodynamisch und flach sein. Bei schlechtem Wetter könnte es trotz allem geschehen, dass die Saugnäpfe mangelnde Ansaugstärke besitzen und das Gerät dadurch abfällt.

Bei dem von uns betrachteten „Roboter“ können auch Konflikte auftreten, wenn die Fensterscheiben zu klein und schmal sind.

6.6 Erweiterungen in Aussicht

Es ist vorzuziehen, in Puncto Datensicherheit und Auslesbarkeit, Speicherkarten mit einem Dateisystem wie etwa FAT16/32 zu verwenden. Dazu müssten jedoch Routinen in Assembler im Systemcode vorliegen, die schreibend zugreifen können.

Weitere Putzprogramme hinzuzufügen ist ein leichtes, – sofern der Bestand der Sensoren und Aktuatoren ausreicht– da die hohe Registeranzahl und der reiche Befehlssatz flexible Assemblerroutinen ermöglicht und der Zugriff auf 256 verschiedene Putzprogramme, auch über die Fernbedienung vorgesehen ist.

Offensichtlich wurde die Fernsteuerungsfunktion nur theoretisch behandelt. Eine detaillierte schaltungstechnische Definition und Evaluation steht daher noch aus. In der Praxis wird sich die Fernsteuerung zwecks Fehlerbehebung als unverzichtbar erweisen. Daher wurde nicht unversucht gelassen, die theoretische Praktikabilität dadurch zu erhöhen.

Auf der technischen Ebene könnte man die ALU effizienter implementieren, indem man sie um eine „Carry-lookahead“-Einheit erweitert. Somit würde die Gatterlaufzeit von etwa ($3 * 16 = 48$, 16 Volladdierer mit maximaler Gatterlaufzeit von 3) stark reduziert. Desweiteren könnte man untersuchen, inwiefern Pipelining und L1-Caches, wie auch das Caching des Hauptspeichers die Leistung steigern könnten.

7 Literatur- und Quellenverzeichnis

1. http://en.wikipedia.org/wiki/Hamming_code, Stand 03.06.2009
2. http://en.wikibooks.org/wiki/Microprocessor_Design/Register_File, Stand 20.06.2009
3. http://www2.informatik.hu-berlin.de/rok/ca/SS03/, Stand 30.06.2009
4. Volladdierer aus Wikipedia (de.wikipedia.org/wiki/Volladdierer), Stand 30.06.09
5. http://rab.ict.pwr.wroc.pl/ kreczmer/wds/prezentacje/przeglad.pdf
6. Folien und Übungen der Vorlesung Technische Informatik 2, SS09
7. Andrew S. Tanbenbaum: Computerarchitektur, 5. Auflage
8. Sensors and Actuators, Volume 102, Number 3, 1 January 2003
9. Projekt „Meerwasserentsalzungsmodul SS04“

8 Bearbeitungsmatrix

Aufgabenteil	Hauptbearbeiter		
	Justin	Thomas	Marianne
1.1 – 1.3			X
1.4	X		
2.1 – 2.2		X	
2.3			X
2.4	X		
2.5		X	
2.6	X		
3.1			X
3.2 – 3.3	X		
3.4			X
4.1 – 4.5	X		
4.6			X
4.7 – 4.9	X		X
4.10 – 4.11			X
5.1 – 5.9	X		
5.10		X	
6.1 – 6.2			X
6.3	X		
6.3 – 6.4			X
6.5		X	
6.6	X		
Schaubilder		X	
LaTeX			X
Präsentation			X
Gesamt	25	9	15