Benutzerhandbuch Motorsteuereinheit

Christian Schröder

17. November 2005

Technische Universität Braunschweig
Institut für Betriebssysteme und Rechnerverbund

Studienarbeit

Entwicklung einer Motorsteuereinheit für ein Fahrmodul

von
cand. informations-systemtechnik Christian Schröder

Aufgabenstellung und Betreuung:
Prof. Dr. Lars Wolf und Dipl.-Ing. Dieter Brökelmann

Braunschweig, den 17. November 2005

Inhaltsverzeichnis

1 Einleitung

In dieser Studienarbeit wird eine Motorsteuereinheit beschrieben, die in dem Fahrzeug (Fahrmodul) vom Praktikum „Ubiquitous Computing“ (Informations-Systemtechnik II) am Institut für Betriebssysteme und Rechnerverbund (IBR) der TU Braunschweig eingesetzt werden soll. Die Motorsteuerung wird über den I^2C-Bus von der Hauptplatine des Fahrzeugs gesteuert, welche von den Praktikanten programmiert wird.

Dieses Benutzerhandbuch soll es dem Praktikanten (Anwender) ermöglichen, die Motorsteuereinheit über den I^2C-Bus anzusteuern. Darüber hinaus ist es mit Hilfe des Handbuchs möglich, die Servos zu kalibrieren und das dazugehörende Programm mit FlashMagic in den P89C664-Mikrocontroller hochzuladen. Nur in Ansätzen kann ein Programmierer mit diesem Handbuch das Programm selbst erweitern. Zu diesem Zweck eignet sich besser das Arbeiten mit der vollständigen Studienarbeit[1]. Das Handbuch beschreibt die Ansteuerung der Motorsteuereinheit und Bedienung des Motor-Testprogramms, also der im Rahmen der Studienarbeit entwickelten Mikrocontroller-Programme. Zusätzlich wird in diesem Handbuch die Bedienung der dafür benötigten Programme angesprochen.

Für die diversen Nutzungsarten der Motorsteuereinheit sind verschiedene Programme notwendig:

- **Flash Magic** lädt das fertige Programm in den Mikrocontroller hoch
- **HyperTerminal** überwacht serielle Ausgaben der Motorsteuerung
- **μVision**
  - ist die Entwicklungsumgebung für die Programmentwicklung der Motorsteuerung
  - ist der Monitor für die Ausführung des Motor-Testprogramms

2 Ablaufbedingungen

Die Entwicklung der Motorsteuerung wurde unter dem Betriebssystem Windows 2000 durchgeführt. Sämtliche benötigten Programme sind für dieses Betriebssystem erhältlich. Neuere Windows-Betriebssysteme (z.B. Windows XP) sind ebenfalls unterstützt.

Die folgende Hardware und Programme werden für verschiedene Einsatzzwecke benötigt:

Motorsteuerung

Für die Ausführung der Motorsteuerung wird ein P89C664-Mikrocontroller der Firma Philips benötigt, welcher sich auf der Motorsteuer-Platine befindet. Mit Hilfe der Ausarbeitung der Studienarbeit[1] kann das Board nachgebaut werden. Weiterhin werden bis zu drei Modellbau-Servos benötigt, zwei modifizierte und ein handelsüblicher. Details sind ebenfalls der Studienarbeit zu entnehmen. Die Spannungsversorgung von 7,2 Volt Gleichstrom kann von einem Netzteil oder Modellbau-Akku geliefert werden. Ein PC mit seriellem COM-Port und der benötigten Programme lädt das Programm per In-System-Programming (ISP) in den Mikrocontroller hoch.

Motor-Testprogramm

Für die Ausführung des Motor-Testprogramms wird ein P89C552-Mikrocontroller der Firma Philips benötigt, welcher sich auf der Praktikums-Hauptplatine befindet. Dieses Board wird am IBR verwendet, um das Fahrmodul zu steuern, dessen Motoren die hier beschriebene Motorsteuerung steuert. Das Programm μVision führt das Motor-Testprogramm im Monitor-Modus auf einem PC mit seriellem COM-Port aus, der an die Hauptplatine angeschlossen ist.

Flash Magic

Flash Magic ist eine kostenlose ISP Software für Philips Flash Mikrocontroller. Sie kann frei runtergeladen werden von `http://www.esacademy.com/software/flashmagic/`. Flash Magic wird benötigt, um das Programm in den P89C664-Mikrocontroller der Motorsteuerung hochzuladen. Ein Programm muss (neu) hochgeladen werden, wenn es in μVision verändert wurde oder es sich um ein neues Board handelt, welches noch nicht mit dem Programm bestückt wurde. Die aktu-

elle Version 2.40 funktioniert unter den Betriebssystemen Windows 95, 98, ME, NT, 2000, XP. Für die Kommunikation mit dem Board wird ein serieller **COM-Port** benötigt.

HyperTerminal

Das Microsoft HyperTerminal ist in den Windows Betriebssystemen enthalten. Es dient der Anzeige der Kontroll-Ausgaben der Motorsteuerung. Alternativ kann jedes andere geeignete Terminal verwendet werden.

μVision

Die Entwicklungsumgebung für die Programmierung und den Monitor-Betrieb ist μVision von der Firma Keil Software, Inc. Eine kostenlose Evaluations-Version kann unter http://www.keil.com/uvision2/ runtergeladen werden. Diese Demo eignet sich aber nicht zur Verwendung in diesem Projekt, da die maximal erlaubte Code-Größe überschritten wird.

Ports

Für das Hochladen eines Programms in die Motorsteuerung und für die Kontrolle der Ausgaben am Terminal wird ein (nacheinander genutzter) serieller COM-Port (RS-232-Schnittstelle) am Computer benötigt. Soll zur gleichen Zeit wie die Kontrolle der Motorsteuerungs-Ausgaben am Terminal das Motor-Testprogramm mit der Monitor-Funktion von μVision ausgeführt werden, erfordert dies einen zweiten COM-Port.

3 Programminstallation und Programmstart

Zunächst müssen die benötigten Programme Flash Magic und μVision installiert werden. Dabei sind die Installationshinweise des jeweiligen Programms zu beachten. Das HyperTerminal befindet sich bei der Standart-Installation des Windows-Betriebssystems bereits im Startmenü im Untermenü Programme - Zubehör - Kommunikation.

Motorsteuerung für Anwender

Aus dem Verzeichnis der Studienarbeit[1] (von CD-ROM oder dem Archiv) wird zumindest aus dem Verzeichnis HEX-file Motorsteuerung die Datei motorst664.hex benötigt. Dabei handelt es sich um die originale Datei, die die Daten für den Flash-Speicher des Mikrocontrollers enthält. Sie kann mit Flash Magic hochgeladen werden. Wie das Hochladen funktioniert steht in Abschnitt 4.2.

Motor-Testprogramm

Für den Motortest wird das Verzeichnis Motortest benötigt. Es beinhaltet sämtliche notwendigen Dateien für ein μVision-Projekt. Nach der Installation von μVision kann das Projekt mit einem Doppelklick auf die Datei motor_test.Uv2 geöffnet werden. Die Bedienung wird in Abschnitt 4.4 beschrieben.

Motorsteuerung für Programmierer

Falls das Programm der Motorsteuerung verändert werden soll, wird das Verzeichnis Studiena664 benötigt, welches wie beim Motortest das vollständige μVision-Projekt enthält. Auf die Veränderung des Programms wird in diesem Handbuch nicht weiter eingegangen.

4 Bedienungsanleitung

4.1 Motorsteuerung

Die Motorsteuereinheit wird von der Hauptplatine über den I^2C-Bus mit Steuer-Befehlen versorgt. Diese Befehle sind in Abschnitt 5 spezifiziert. Die Motorsteuerung ist der Slave mit der Standart-Adresse 0x10. Die Hauptplatine übernimmt die Master-Funktion. Die verwendete Datenrate ist 100 kbit/s.

Im laufenden Betrieb gibt die Motorsteuerung Ausgaben an der seriellen Schnittstelle (COM-Port) aus. Auf jeden Fall werden bei einem Reset einige Zeilen mit Identifikation und Version ausgegeben. Wenn die Debug-Option aktiviert ist (siehe dazu Befehl **ENABLE_DEBUG**), werden zusätzliche Ausgaben gemacht, die die Fehlersuche erleichtern.

Diese zusätzlichen Ausgaben betreffen über den I^2C-Bus ankommende Befehle, gesendete Nachrichten, aufgetretene Fehler, die Kalibrierung, in die Warteschlange eingereihte sowie entfernte und ausgeführte Befehle. Der Aufruf der zuständigen Funktionen (`printf(...)` bzw. `puts(...)`) wird durch eine Mutex-ähnliche Variable geschützt, damit keine gleichzeitigen Aufrufe erfolgen.

Dadurch ist es möglich, dass verschiedene schnell aufeinander folgende Debug-Ausgaben einfach nicht erfolgen.

Bei aktiviertem Debug-Modus ist die Verarbeitungsgeschwindigkeit der Befehle geringer. Beim Debuggen ist darauf zu achten, dass die Befehle nicht zu schnell an die Motorsteuerung gesendet werden.

Auch im normalen Betrieb (deaktivierter Debug-Modus) ist darauf zu achten, dass die I^2C-Bus-Datagramme nicht zu schnell nacheinander gesendet werden, damit die Befehle in der Motorsteuerung bearbeitet werden können und keiner verworfen wird. Ein mit dem Motortest ermittelter Wert für den Abstand zweier **MOVE**-Befehle ist 2 ms. Als Hinweis für den Erfolg der Bearbeitung kann der Rückgabewert der Funktion `i2c_send(...)` verwendet werden.

Die in der Datei MOTOR_DEFS.H enthaltenen Strukturen (struct) und Konstanten für die IDs der Befehle können im eigenen Programm, welches mit der Motorsteuereinheit über den I^2C-Bus kommunizieren soll, verwendet werden.

4.2 Motorsteuerung mit Flash Magic hochladen

Eine mit μVision erzeugte HEX-Datei kann mit dem Programm Flash Magic in den P89C664-Mikrocontroller hochgeladen werden. Als Quelle kann entweder die Originaldatei motorst664.hex oder eine selbst modifizierte dienen, welche im Allgemeinen den gleichen Dateinamen haben wird, wenn sie mit dem Projekt der Studienarbeit erzeugt wurde.

Vor dem Start

Bevor Flash Magic gestartet wird, muss sichergestellt werden, dass die Motorsteuereinheit korrekt mit einem COM-Port des PCs verbunden ist. In dem hier dargestellten Beispiel ist dies COM 1. Außerdem darf dieser Port nicht von einem anderen Programm angesprochen werden, ein eventuell für diesen Port geöffnetes HyperTerminal ist zu beenden.

Einstellungen

Nach dem Start von Flash Magic werden alle Werte so eingestellt, wie sie in Abbildung 4.1 zu sehen sind.

- Unter Ziffer 1 muss unter Umständen ein anderer COM Port gewählt werden, falls die Motorsteuereinheit an einem anderen als COM 1 angeschlossen ist.
- Ziffer 2 beinhaltet die Möglichkeit, verschiedene Blöcke des Flash-Speichers zu löschen.
  - Die im Beispiel markierten Blöcke 0 und 1 sind auf jeden Fall zu löschen, da dort das Programm hochgeladen wird.
  - Block 2 und 4 werden nicht verwendet, sind also uninteressant.
  - Block 3 beinhaltet die Kalibrierungsdaten. Er kann gelöscht werden, um die Kalibrierung auf Standart-Werte zurückzusetzen. Wird ein Baustein das erste Mal mit dem Motorsteuer-Programm beschrieben, sollte dieser Block auf jeden Fall gelöscht werden. Mit dem Befehl **CALIBRATE_ERASE** des Protokolls (siehe Abschnitt 5) wird genau dieser Block gelöscht. Wenn die Kalibrierungsdaten erhalten bleiben sollen, darf dieser Block nicht gelöscht werden.
- Unter Ziffer 3 wird die HEX-Datei ausgewählt, die in den Speicher hochgeladen werden soll. In den meisten Fällen wird dies motorst664.hex sein.

  Es ist zu beachten, dass hier die zuletzt hochgeladene Datei voreingestellt ist. Es ist unbedingt notwendig, genau zu überprüfen, ob die hier angegebene Datei auch die gewünschte ist!

- Die Optionen unter Ziffer 4 werden nicht benötigt. Vor allem die Security Bits sollte man nicht setzen, ohne zu wissen was man tut. Diese verhindern unter Umständen das Neubeschreiben des Flash-Speichers. Näheres zu Security Bits ist im Datenblatt des Mikrocontrollers[2] zu finden.
- Ein Klick auf START unter Ziffer 5 beginnt den Schreibvorgang.

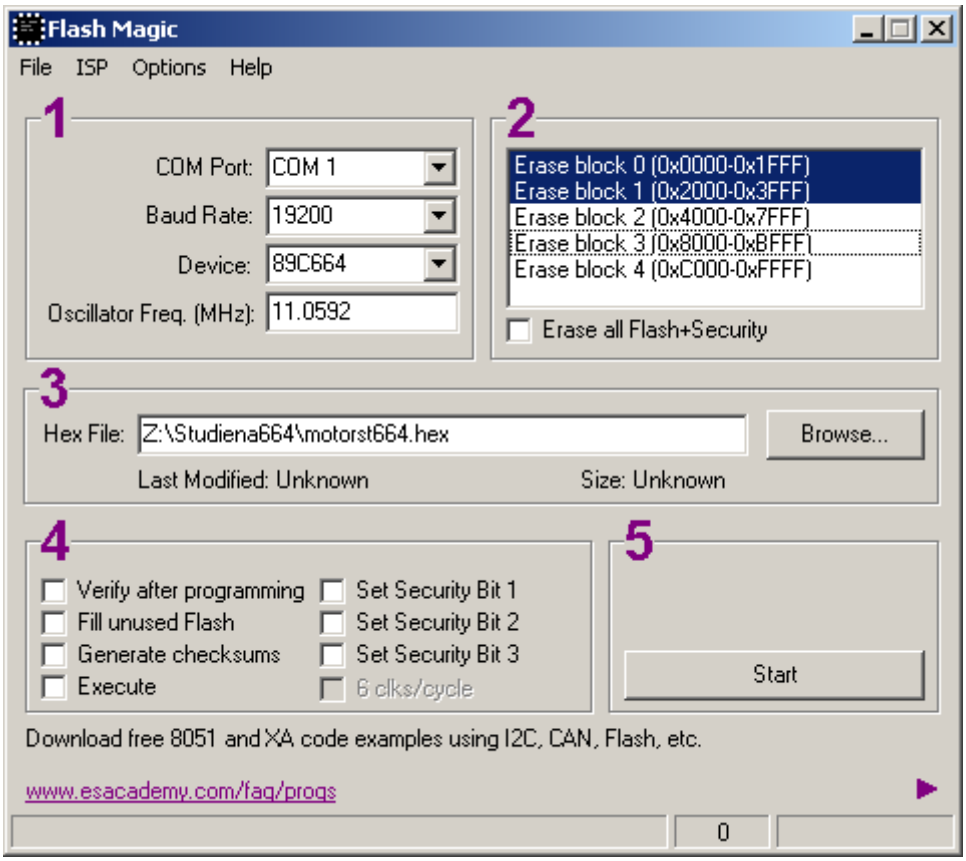

Abbildung 4.1: Einstellungen in Flash Magic

Zu beachten ist, dass dieses Verfahren mit Flash Magic nur für die Motorsteuereinheit nötig ist, nicht für das Hochladen eines Programms in die Hauptplatine. Letzteres wird vom μVision Monitor-Modus übernommen (siehe Abschnitt 4.4).

4.3 Motorsteuerung mit HyperTerminal überwachen

Die Ausgaben der Motorsteuereinheit können zu Kontroll- und Debug-Zwecken mit einem Terminal an der seriellen Schnittstelle gelesen werden. Hier ist die Verwendung des HyperTerminals von Microsoft beschrieben, da es im Betriebssystem enthalten und einfach zu bedienen ist. Es kann aber auch jedes andere Terminal (auch Hardware-Terminals) verwendet werden, das die entsprechenden Einstellungen zulässt.

Vor dem Start

Bevor das HyperTerminal gestartet wird, muss sichergestellt werden, dass die Motorsteuereinheit wie schon beim Hochladen eines Programms mit Flash Magic korrekt mit einem COM-Port des PCs verbunden ist. Dieser Port darf nicht schon von einem anderen Programm verwendet werden. Falls zuvor beispielsweise das Programm mit Flash Magic hochgeladen wurde, muss Flash Magic beendet werden.

Einstellungen

Nach dem Start des HyperTerminals muss eine neue Verbindung angelegt und benannt werden. Im anschließenden Fenster Verbinden mit wird im letzten Eintrag (Verbindung herstellen über:) der gewünschte COM-Port eingestellt, z.B. COM1. Nach Bestätigung durch OK sind die in Abbildung 4.2 gemachten Einstellungen einzutragen. Sobald wieder mit OK bestätigt ist, besteht die Verbindung und die Ausgaben der Motorsteuerung werden im Terminal angezeigt.

In der Standart-Einstellung gibt die Motorsteuerung nur nach einem Reset einige Begrüßungs-Zeilen an der seriellen Schnittstelle aus. Weitere Status-Meldungen werden erst nach Senden des Befehls **ENABLE_DEBUG** gesendet.

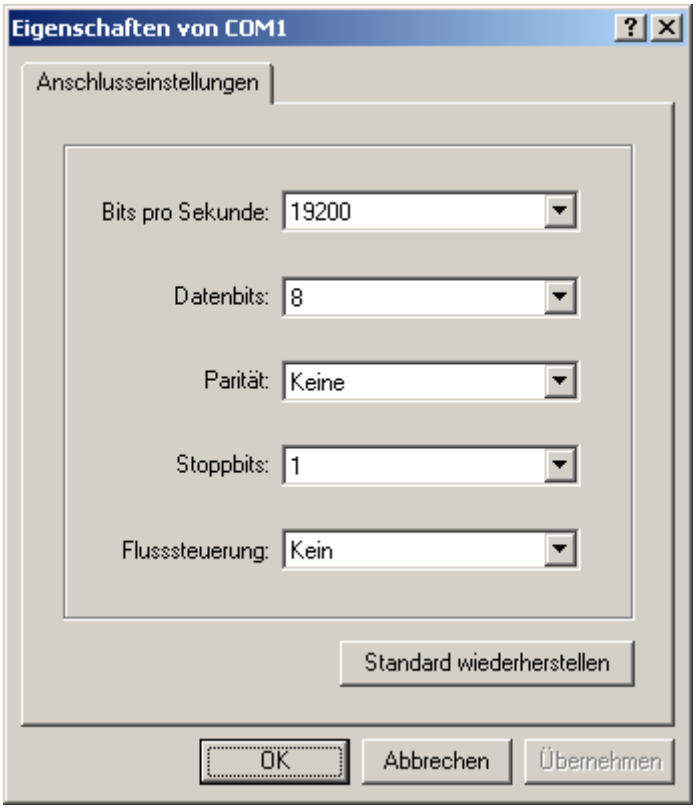

Abbildung 4.2: Verbindungs-Einstellungen im Terminal

4.4 Motor-Testprogramm

Zum Umfang dieser Arbeit gehört das Programm *Motortest*. Mit diesem Motor-Testprogramm können sämtliche Funktionen der Motorsteuereinheit gestestet und die Servos kalibriert werden. Es wird die Hauptplatine des Praktikum-Fahrmoduls benötigt. Auf dessen P89C552-Mikrocontroller wird das Testprogramm hochgeladen. Anschließend können Befehle über dessen serielle Schnittstelle gesendet werden, die umgesetzt werden in Befehle des Protokolls, die über den I^2C-Bus an die Motorsteuerung gesendet werden. Das Empfangen und Darstellen von Rückgabe-Nachrichten von der Motorsteuerung ist möglich. Die Nachrichten werden als Rohdaten und als aufgeschlüsselte Nachricht dargestellt.

Vor dem Start

Da beim Testen der Motorsteuereinheit deren Debug-Ausgaben am Hyperterminal beobachtet werden sollten (siehe Abschnitt 4.3), ist es sinnvoll, die Hauptplatine an einem zusätzlichen COM-Port des PCs anzuschließen, damit beide

Platinen gleichzeitig angesprochen werden können. Zu diesem Zweck bietet sich in dem hier dargestellten Beispiel der Port COM 2 an.

Funktionen des Motor-Testprogramms

Das Motor-Testprogramm ist ein Programm, welches auf einem P89C552-Mikrocontroller von Philips ausgeführt werden kann und dort über den I^2C-Bus kommuniziert. In dieser Anwendung wird es in die Praktikums-Platine hochgeladen, die über den genannten Controller verfügt. Gesteuert wird es über die serielle Schnittstelle (den COM-Port) aus dem Debug-Modus von μVision heraus. Dieser Modus stellt ein Terminal zur Verfügung, welches Ein- und Ausgaben über die serielle Schnittstelle erlaubt. Hierüber wird der Motortest gesteuert.

Das Motor-Testprogramm kann die im Protokoll in Abschnitt 5 vorgegebenen Befehle samt Parameter senden und empfangene Reaktionen (Nachrichten) verarbeiten und darstellen. Zusätzlich erlaubt es das Programm, beliebige Bytefolgen über den I^2C-Bus zu senden.

Debug-Modus starten

Nachdem das Projekt motor_test.Uv2 in μVision geöffnet ist, muss sichergestellt werden, dass als „Target“ der „Monitor“ ausgewählt ist (siehe Abbildung 4.3). Anschließend wird der Debug-Modus wie in Abbildung 4.3 zu sehen mit einem Klick auf Start/Stop Debug Session gestartet. μVision lädt nun automatisch das Motor-Testprogramm über die serielle Schnittstelle (z.B. COM 2) hoch. Das Programm befindet sich anschließend im RAM-Speicher an der Stelle 0x2000. Die dann automatisch ausgeführte Monitor-Initialisierungs-Datei initialization monitor.ini springt zu dieser Stelle (siehe Abbildung 4.4). Das Programm wird korrekt ausgeführt, wenn es sich mit der Zeile „Motortestprogramm“ im „Serial“-Fenster meldet. Dieses Fenster wird automatisch in den Vordergrund eingeblendet und zeigt die Kommunikation mit der Hauptplatine am COM-Port.

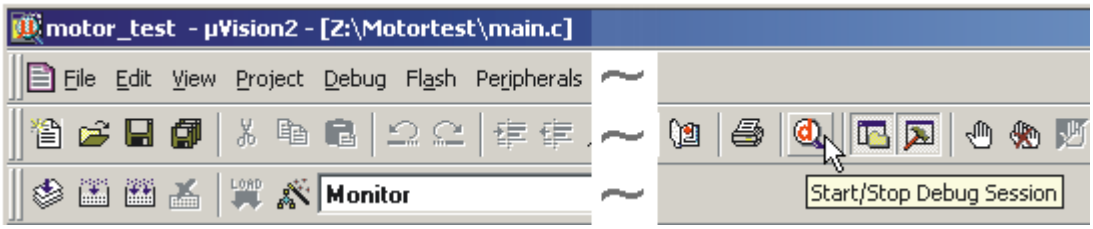

Abbildung 4.3: Starten des Debug-Modus

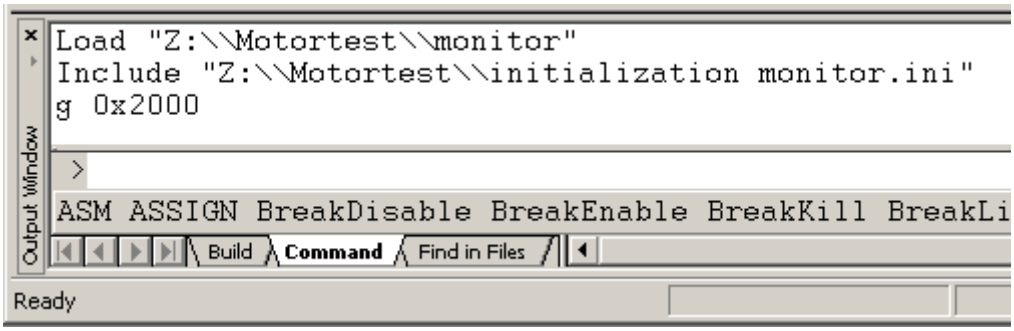

Abbildung 4.4: Automatischer Sprung an die Adresse 0x2000

Befehle im Motor-Testprogramm

Es folgt eine Tabelle der Eingaben, die das Motor-Testprogramm an der seriellen Schnittstelle akzeptiert. Es handelt sich dabei um Eingaben, die das Senden eines Protokoll-Befehls bewirken, das Empfangen von ebenfalls im Protokoll spezifizierten Nachrichten und das Senden und Empfangen von Rohdaten erlauben.

Befehl mit Parametern Parameter : Bereich	Funktion (gesendeter Befehl, ...) Funktion
enable_debug <d> <d> : 0|1	Sendet **ENABLE_DEBUG**: (De-)aktiviert den Debug-Modus 0: aktiviert, 1: deaktiviert
reset	Sendet **RESET**: Starte Modul neu (*ohne* Reset der Kalibrierungsdaten)
reset_all	Sendet **RESET_ALL**: Starte Modul neu (*mit* Reset der Kalibrierungsdaten)
set_speed <s> <sp> <s> : 0|1|2 <sp> : [-128, 127]	Sendet **SET_SPEED**: Setzt Geschwindigkeit <sp> für Servo <s> Servo Geschwindigkeit
cset_upper <s> <u> <s> : 0|1|2 <u> : [0, 65535]	Sendet **CALIBRATE_SET_UPPER**: Setzt *obere* Grenze <u> für Servo <s> Servo obere Grenze
cset_middle <s> <m> <s> : 0|1|2 <m> : [0, 65535]	Sendet **CALIBRATE_SET_MIDDLE**: Setzt *mittlere* Grenze <m> für Servo <s> Servo mittlere Grenze

cset_lower <s> <l> <s> : 0|1|2 <l> : [0, 65535]	Sendet **CALIBRATE_SET_LOWER**: Setzt *untere* Grenze <l> für Servo <s> Servo untere Grenze
cadd_upper <s> <u> <s> : 0|1|2 <u> : [-128, 127]	Sendet **CALIBRATE_ADD_UPPER**: Addiert zur *oberen* Grenze den Wert <u> für Servo <s> Servo addierter Wert
cadd_middle <s> <m> <s> : 0|1|2 <m> : [-128, 127]	Sendet **CALIBRATE_ADD_MIDDLE**: Addiert zur *mittleren* Grenze den Wert <m> für Servo <s> Servo addierter Wert
cadd_lower <s> <l> <s> : 0|1|2 <l> : [-128, 127]	Sendet **CALIBRATE_ADD_LOWER**: Addiert zur *unteren* Grenze den Wert <l> für Servo <s> Servo addierter Wert
calibrate_erase oder erase	Sendet **CALIBRATE_ERASE**: Löscht den Kalibrierungs-Speicher des Flash. Nach einem **CALIBRATE_ERASE** bitte ein Hardware-Reset der Motorsteuerung durchführen!
calibrate_write_flash oder flash	Sendet **CALIBRATE_WRITE_FLASH**: Sichert die aktuellen Kalibrierungswerte im Flash
calibrate_get_values <s> oder c_values <s> <s> : 0|1|2	Sendet **CALIBRATE_GET_VALUES**: Fordert die aktuellen Kalibrierungswerte für Servo <s> an. Danach receive 9 ausführen. Servo
calibrate_get_speed_value <s> oder c_svalue <s> <s> : 0|1|2	Sendet **CALIBRATE_GET_SPEED_VALUE**: Fordert die aktuellen Integer-Geschwindigkeit für Servo <s> an. Danach receive 5 ausführen. Servo

calibrate_set_speed oder c_set_speed <s> <sp> <s> : 0|1|2 <l> : [0, 65535]	Sendet **CALIBRATE_SET_SPEED**: Setzt die Geschwindigkeit <sp> für Servo <s> Servo Geschwindigkeit
move <sp0> <t0> <sp1> <t1> <sp0> : [-128, 127] <t0> : [0, 65535] <sp1> : [-128, 127] <t1> : [0, 65535]	Sendet **MOVE**: Setzt Geschwindigkeiten <sp0>, <sp1> und Ticks (Bewegungs-Einheiten) <t0>, <t1> für Servos 0 und 1 Geschwindigkeit Servo 0 Ticks Servo 0 Geschwindigkeit Servo 1 Ticks Servo 1
get_status	Sendet **GET_STATUS**: Fragt Status der Servos 0 und 1 ab. Danach receive 12 ausführen.
stop	Sendet **STOP**: Stoppt Servos 0 und 1, d.h. unterbricht die Steuersignale
start	Sendet **START**: Startet Servos 0 und 1
move_2 <sp> <sp> : [-128, 127]	Sendet **MOVE_2**: Setzt Geschwindigkeit <sp> für Servo 2 Geschwindigkeit/Position Servo 2
get_status_2	Sendet **GET_STATUS_2**: Fragt Status von Servo 2 ab. Danach receive 4 ausführen.
stop_2	Sendet **STOP_2**: Stoppt Servo 2, d.h. unterbricht das Steuersignal
start_2	Sendet **START_2**: Startet Servo 2
send <byte> <byte> ... oder s <byte> <byte> ...	Sendet eine Bytefolge (bis zu 10 Bytes) über den I^2C-Bus
help oder h	Zeigt eine Hilfe (wie diese) an
receive <bytes> oder r <bytes>	Empfange <bytes> Bytes vom I^2C-Bus. Siehe Daten Empfangen.

Tabelle 4.1: Eingaben Motortest

Das Motor-Testprogramm gibt nach der Eingabe des Befehls aus, wie viele Bytes über den I^2C-Bus gesendet wurden. Steht hier eine „0“, war die Eingabe des Befehls fehlerhaft.

Es findet keine Überprüfung der Parameter statt! Parameter im falschen Zahlenraum oder eine falsche Anzahl Parameter wird ignoriert. In dem Fall wird der Befehl trotzdem gesendet, allerdings mit sinnlosen Parametern.

Daten empfangen

Mit der Eingabe receive <bytes> (siehe Tabelle 4.1) liest das Motor-Testprogramm die angegebene Anzahl Bytes vom I^2C-Bus. Die empfangenen Bytes werden anschließend im Format *unsigned char* angezeigt. Falls das erste Byte (nach der Empfänger-Adresse) einer Nachricht zugeordnet werden kann, die im Protokoll spezifiziert ist, wird diese Nachricht dargestellt. Es ist zu beachten, dass eventuell mehrere nacheinander empfangene aber gleichzeitig abgefragte Nachrichten nicht als solche dargestellt werden (sondern nur die erste) und somit per Hand aus dem angezeigten Datenstrom extrahiert werden müssen. Aus diesem Grund ist es auch ratsam, mehr Bytes als erwartet anzeigen zu lassen. Überflüssige (nicht empfangene) Bytes werden mit Nullen gefüllt.

Empfangene Fehler-Nachrichten werden mit ihrem Fehler-Code dargestellt. Die Bedeutung ist der Auflistung in Abschnitt 6.3 zu entnehmen.

Der Anwender muss selbst dafür sorgen, potentiell empfangene Daten abzufragen. Das Motor-Testprogramm kündigt ankommende Daten nicht eigenständig an. Für angeforderte Daten ist dieses Vorgehen kein Problem, von der Motorsteuerung unerwartet gesendete Fehler-Nachrichten müssen aber ebenso abgefragt werden.

Debug-Modus beenden

Zum Beenden des Monitors und Debug-Modus ist ein Klick auf den roten Stop-Button nötig. Die zwei nun erscheinenden Fenster sind jeweils mit Stop Debugging zu bestätigen.

4.5 Kalibrierung der Servos

Nutzen der Kalibrierung

Jeder im Handel erhältliche Servo unterscheidet sich leicht (bei verschienen Modellen auch stark) von anderen. Das betrifft vor allem das in dieser Anwendung wichtige Verhalten aufgrund der Steuersignale, die die Motorsteuereinheit erzeugt. Die Motorsteuerung soll die Geschwindigkeits-Befehle (bei den Motoren 0 und 1) immer gleich ausführen. Das Protokoll sieht hierfür eine Skala von -128 bis +127 vor. Der Wert +127 steht für die maximale Vorwärts-Geschwindigkeit, der Wert -128 für die maximale Rückwärts-Geschwindigkeit und der Wert 0 für das Stehenbleiben des Motors.

Die Kalibrierung erlaubt es, diese drei Werte für jeden (originalen oder modifizierten) Servo separat anzupassen.

Es ist somit möglich, die gemeinsame Maximalgeschwindigkeit der zwei Antriebsmotoren festzulegen, und diese in die Motorsteuerung flüchtig (in den RAM) oder dauerhaft (in den Flash) zu speichern. Kann beispielsweise einer der beiden Servos schneller als der andere drehen, ist es mit Hilfe der Kalibrierung möglich, den schnelleren auf die Maximalgeschwindigkeit des langsameren zu drosseln. Das führt zu einer Geradeausbewegung des Fahrmoduls, wenn beide Motoren mit dem Wert +127 bzw. -128 angesprochen werden.

Da die Servos im Fahrmodul spiegelverkehrt zueinander angebracht werden können, erlaubt die Kalibrierung die Umkehrung der Drehrichtungen. Werden die Kalibrierungswerte für maximale Vorwärts- und Rückwärtsgeschwindigkeit vertauscht, dreht sich der Motor andersrum. Jetzt können beide Motoren mit einem identischen Wert angesprochen werden (z.B. +128), was zu einer entgegengesetzten Drehung mit gleicher Geschwindigkeit führt. Das wiederum bewirkt durch die spiegelverkehrte Anbringung der Motoren eine Geradeausbewegung des Fahrmoduls.

Der dritte wichtige Kalibrierungswert neben den Maximalgeschwindigkeiten für Vorwärts- und Rückwärtsbewegung ist die Mittelposition. Damit ein Motor beim Geschwindigkeitswert 0 stehen bleibt, wird diese während der Kalibrierung ebenfalls festgelegt.

Die Kenntnis von Maximalgeschwindkeit und Mittelposition erlaubt die Interpolation von Zwischenstufen. Die 8-Bit-Auflösung des Geschwindigeitswertes stellt 256 verschiedenen Geschwindigkeitsstufen zu Verfügung. Bei gleichem Geschwindkeitswert drehen sich zwei aufeinander abgestimmte (kalibrierte) Motoren mit der gleichen Geschwindigkeit in die gleiche Richtung. Das Fahrmodul führt dann eine Geradeausbewegung durch.

Zusätzlich zur Abstimmung zweier Motoren aufeinander kann die Kalibrierung mehrere Fahrmodule gleich auf Geschwindigkeitsbefehle reagieren lassen. Das er-

fordert allerdings eine sehr genaue Kalibrierung und den Vergleich vieler Servo-Motoren. In diesem Fall ist es nötig, sämtliche verwendeten Motoren auf die Maximalgeschwindigkeit des langsamsten zu drosseln.

Kalibrierungswerte

Bisher wurden Geschwindigkeiten immer in Werten zwischen -128 und +127 angegeben. Diese Werte sind die für den End-Anwender sichtbaren. Da sie aber nur einen Ausschnitt der tatsächlich zur Verfügung stehenden Auflösung der Geschwindigkeitsregelung der Motorsteuereinheit darstellen (deswegen wird kalibriert), werden bei der Kalibrierung andere Werte verwendet. Diese Werte werden später im laufenden Betrieb in die vom Anwender übermittelten Geschwindigkeiten umgerechnet.

Die bei der Kalibrierung verwendeten 16-Bit-Werte reichen jeweils von 0 bis 65535. In diesem Bereich liegen für jeden Servo die Maximalgeschwindigkeiten und Mittelwerte.

Es können für jeden Servo separat drei verschiedene Werte festgelegt werden:

- UPPER: Wert für die Maximalgeschwindkeit in Vorwärts- bzw. Rückwärtsrichtung
- MIDDLE: Wert für die Mittelstellung, in der sich der Servo nicht bewegt
- LOWER: Wert für die Maximalgeschwindgkeit in Rückwärts- bzw. Vorwärtsrichtung

Für Servo 1 sind UPPER und LOWER vertauscht.

Wenn bisher keine Werte in den Flash-Speicher geschrieben wurden, bzw. der Speicher gelöscht wurde, sind die oben genannten Werte nach einem Hardware-Reset oder dem Software-Befehl **RESET_ALL** für jeden Servo auf Default-Werte des Programmcodes gesetzt.

Diese Default-Werte sind wie folgt festgelegt:

Servo 0 und 2:

- UPPER: 60927
- MIDDLE: 62770
- LOWER: 64613

Servo 1:

- UPPER: 64613
- MIDDLE: 62770
- LOWER: 60927

Die Default-Werte sehen für die Servos 0 und 2 die gleiche Drehrichtung vor. Bestimmt wird sie durch die Werte UPPER > MIDDLE > LOWER. Durch die entgegengesetzte Einstellung LOWER > MIDDLE > UPPER für Servo 1 ist seine Drehrichtung wegen spiegelverkehrter Platzierung am Fahrmodul umgekehrt. Bei

der eigenen Kalibrierung kann die Drehrichtung durch Änderung der Verhältnisse LOWER >< UPPER eingestellt werden.

Kalibrierungswerte ermitteln

Die Ermittlung der richtigen Kalibrierungswerte UPPER, MIDDLE und LOWER für jeden Servo kann auf verschiedene Weise vorgenommen werden.

Zuerst folgen Möglichkeiten, wie ein gewünschter Wert zu Testzwecken übermittelt und zur Ausführung gebracht werden kann.

- **CALIBRATE_SET_SPEED**:
  Die direkte Art und Weise, einen Servo in der gewünschten Geschwindigkeit (Achtung: 16-Bit-Geschwindigkeit!) drehen zu lassen, ist der Befehl **CALIBRATE_SET_SPEED**.

- **SET_SPEED**:
  Ist eine grobe Kalibrierung bereits vorgenommen, kann auch der Befehl **SET_SPEED** verwendet werden, um eine Geschwindigkeit zu setzen. Hier ist zu beachten, dass keine Werte jenseits von den aktuellen Werten für UPPER und LOWER möglich sind. Außerdem ist die Genauigkeit unter Umständen sehr viel schlechter als bei Verwendung des oben genannten Befehls. Sind die Werte UPPER und LOWER schon in den RAM geschrieben, bietet sich diese Methode zum Finden der Mittelposition an. Ist die gewünschte Geschwindigkeit eingestellt, kann der Kalibrierungswert anschließend mit dem Befehl **CALIBRATE_GET_SPEED_VALUE** angefordert werden.

- **CALIBRATE_ADD_xxx**:
  Falls eine Anwendung es erfordert, können die Befehle **CALIBRATE_ADD_UPPER**, **CALIBRATE_ADD_MIDDLE** und **CALIBRATE_ADD_LOWER** mit einem anschließenden **SET_SPEED** verwendet werden.

Welche Werte nach obigen Methoden ausprobiert werden sollten, bleibt dem Anwender überlassen, der die Kalibrierung durchführt.

Eine schnelle aber auch sehr ungenaue Methode ist beispielsweise die Ermittlung der Maximalwerte nach Gehör. In dem Fall wird der Wert für UPPER oder LOWER erhöht oder verringert bis keine Geräusch-Änderung mehr stattfindet.

Eine sehr viel genauere Methode ist das Zählen der zurückgelegten Entfernungsschritte (Pulse). Zu diesem Zweck sind die Befehle **MOVE** und **GET_STATUS** nützlich. Mit **MOVE** kann eine unendliche Bewegung einer bestimmten Geschwindigkeit initiiert werden, woraufhin nach einer festen Zeit mit Hilfe des Befehls

GET_STATUS die zurückgelegten Entfernungsschritte abgefragt werden. Ein Vergleich zwischen den Servos liefert Informationen, welcher Servo schneller oder langsamer ist und kalibriert werden muss. Eine detailliertere Programm-Idee ist in Abschnitt 4.5.1 dargestellt.

Kalibrierungswerte setzen und speichern

Sind die Kalibrierungswerte bekannt, müssen sie in der Motorsteuereinheit gespeichert werden. Dafür stehen zwei verschiedene Verfahren zu Verfügung:

- Flüchtiges Speichern im RAM
- Permanentes Speichern im Flash

Die Vorteile des **flüchtigen Speicherns im RAM** des Mikrocontrollers sind folgende: Die Werte sind jederzeit änderbar und überschreibbar. Deswegen können die Werte auch während dem oben beschriebenen Kalibriervorgang temporär gesetzt werden. Außerdem ist diese Art flexibel für individuelle Kalibrierung, falls z.B. verschiedene Kalibrierungswerte für verschiedene Programme angewendet werden sollen oder verschiedene Praktikumsgruppen eine eigene Kalibrierung verwenden möchten. Es ist zu beachten, dass die im RAM gespeicherten Werte bei jedem Reset (Hardware-Reset und Reset-Befehl **RESET_ALL**) verloren gehen. Das betrifft nicht den Befehl **RESET**.

Die Befehle zum Speichern der Werte in den RAM sind die folgenden:

- **CALIBRATE_SET_UPPER**
- **CALIBRATE_SET_MIDDLE**
- **CALIBRATE_SET_LOWER**

Das **permanente Speichern im Flash** setzt voraus, dass alle Werte zuvor im RAM stehen. Dort nicht explizit gesetzte Werte bleiben dann auf den Default-Werten. Die Vorteile des Speicherns im Flash sind folgende: Die gespeicherten Werte bleiben dauerhaft erhalten. Da ein Fahrmodul üblicherweise im Betrieb nicht baulich verändert wird, ist eine Kopplung der Motorsteuereinheit mit den angeschlossenen Servos sinnvoll. Anwender (Programmentwickler für die Hauptplatine) müssen die Werte nicht setzen und können mit den 8-Bit-Geschwindigkeiten arbeiten. Im Flash stehende Kalibrierungswerte können jederzeit temporär überschrieben werden, indem sie im RAM geändert werden. Nach einem **RESET_ALL** oder Hardware-Reset sind wieder die Werte aus dem Flash gesetzt. Die Werte können nur für alle Servos gleichzeitig gespeichert werden. Bei einer Änderung müssen sämtliche Kalibrierungsdaten mit dem Befehl **CALIBRATE_ERASE** gelöscht, ein Reset durchgeführt werden und der normale

Speichervorgang durchgeführt werden (zuerst Werte in RAM schreiben, dann alle gleichzeitig in den Flash-Speicher schreiben). Statt diesen Befehl anzuwenden, können die Kalibrierungsdaten mit Flash Magic gelöscht werden. Zu diesem Zweck muss dort der Block 3 (ab der Adresse 0x8000) gelöscht werden.

Der Befehl zum Speichern der Werte vom RAM in den Flash ist:

- **CALIBRATE_WRITE_FLASH**

Übersicht Kalibrierungswerte setzen und speichern

Es folgt eine Übersicht über das Vorgehen beim Kalibrieren und Speichern der Werte im Flash:

Diese Werte werden bei einem **RESET_ALL** oder Hardware-Reset in den RAM geladen:

- Wenn der Flash-Speicher leer ist:
  - Default-Werte aus dem *Programmcode* werden in den RAM geladen
- Wenn der Flash-Speicher beschrieben ist:
  - Default-Werte aus dem *Flash* werden in den RAM geladen

So kann bei einer Kalibrierung vorgegangen werden:

1. Eventuell vorhandene Werte notieren, falls diese für einen der Servos übernommen werden sollen.
2. Den Flash löschen.
3. Neue Kalibrierungswerte ermitteln und evtl. notierte wieder setzen.
   Anschließend stehen sie im RAM.
4. Die Werte vom RAM in den Flash speichern.

Auslesen der Kalibrierungswerte

Sollen die aktuellen Kalibrierungswerte ausgelesen werden, kann der Befehl **CALIBRATE_GET_VALUES** verwendet werden.

4.5.1 Anwendungsvorschlag Kalibrierung

In diesem Abschnitt wird ein Vorschlag beschrieben, wie ein Programm bei der Kalibrierung zweier Motoren vorgehen könnte. Besonderer Wert wird auf genaue Abstimmung der maximalen Geschwindigkeiten der zwei Fahr-Servos gelegt, damit das Fahrzeug nach der Kalibrierung bei gleichem angesteuerten Geschwindigkeitswert der zwei Motoren auch geradeaus fährt.

Das Prinzip hinter dieser Idee ist die Verwendung der Puls-Zählung in Verbindung mit einer genau festgelegten Messzeit dieser Entfernungsschritte. Der Kern ist ein Algorithmus, der Befehle in bestimmter Reihenfolge und exaktem Timing über den I^2C-Bus sendet. Dabei müssen folgende Schritte eingehalten werden:

- **Kalibrierungsdaten setzen**: Die zu testenden Kalibrierungswerte für die Servos sind zu setzen. Dafür sind die Befehle **CALIBRATE_ADD_xxx** oder **CALIBRATE_SET_xxx** geeignet. Hier ist darauf zu achten, dass sich die Servos bei spiegelverkehrter Montage in die gleiche Richtung bewegen (siehe Abschnitt 4.5).

- **Zeit nehmen**: Zunächst muss die Zeit genommen werden, da die tatsächliche Geschwindigkeit der Servos von der Software nur über die Kombination aus zurückgelegter Strecke (Entfernungsschritte bzw. Pulse) und der Zeit ermittelt werden kann.

- **Servos starten**: Mit dem Befehl **MOVE** werden anschließend beide Motoren mit einer unendlichen (oder hinreichend weiten) Bewegung (welche keinesfalls vor Ablauf der Zeit erreicht sein darf) und maximaler Geschwindigkeit gestartet. Je nach zu testender Richtung ist das +127 für vorwärts oder -128 für rückwärts.

- **Zeit abgelaufen**: Sobald eine bestimmte Zeit abgelaufen ist (z.B. 2 Sekunden), wird mit dem Befehl **GET_STATUS** der Zählerstand der beiden Servos abgefragt. Die Servos laufen während dieser Abfrage noch. Anschließend können sie gestoppt werden.

- **Zählerstände auswerten**: Diese beiden Zählerstände sind auszuwerten. Ziel ist es, bei gleicher vorgegebener Geschwindigkeit an dieser Stelle gleiche Werte für beide Servos zu erhalten. Dann bewegen sie sich gleich schnell. Weiteres Ziel ist es, hier möglichst hohe Werte für die fortgelegten Entfernungsschritte zu erhalten, weil das einer hohen Geschwindigkeit entspricht. Für die Kalibrierung der Mittelstellung muss der Wert Null sein.

Mit dieser Methode können beide Motoren gleichzeitig gestartet und ausgewertet werden. Das hat den Vorteil, dass keine unterschiedlichen Be-arbeitungszeiten der Befehle die Messung beeinträchtigen.

Ein Algorithmus um diesen gerade beschriebenen Kernalgorithmus herum muss die Kalibrierungswerte sinnvoll wählen und die Auswertung der erhaltenen Zählerstände vornehmen.

Beispielsweise kann sich der Algorithmus von zwei Seiten mit immer kleineren Schritten an den Maximalwert herantasten. Die Vorgehensweise kann wie folgt aussehen:

- Mittelposition (Stand) kalibrieren:
  - Servo 0:
    - Den Wert schrittweise (z.B. 1000er-Schritte) von LOWER an UPPER annähern (oder umgekehrt) und die Zählerstände vergleichen. Sobald sich der bisherige Trend umkehrt (z.B. Wert wird zuerst kleiner, dann wieder größer), die Schrittweite verkleinern und zwischen den letzten beiden 1000er-Schritten diesen Punkt mit kleineren Schritten wiederholen.
    - *Alternativ* den Mittelwert zwischen LOWER und UPPER berechnen (oder Default-Wert 62770) und als Ausgangspunkt verwenden.
    - Eventuell ist es sinnvoll zu beachten, dass je nach Annäherung aus Richtung UPPER oder LOWER verschiedene (aber nahe beieinander liegende) Werte zum Stehen des Servos führen. In dem Fall kann der Mittelwert zwischen den beiden verwendet werden, um sicher gegenüber kleinen Veränderungen der Servos zu sein (z.B. Temperaturschwankungen, die zu einer Verschiebung der Mittelposition führen).
    - Das Ergebnis ist der Wert für MIDDLE von Servo 0.
  - Servo 1:
    - Die Mittelstellung wie bei Servo 0 ermitteln.
    - Das Ergebnis ist der Wert für MIDDLE von Servo 1.
- Vorwärts-Kalibrierung:
  - Servo 0:
    - Mit Default-Wert 60927[1] beginnend in 1000er-Schritten größer werden .
    - Sobald der Servo zwischen zwei Messungen langsamer wird in 100er-Schritten vom kleineren Wert ausgehend erhöhen.
    - Sobald der Servo zwischen zwei Messungen langsamer wird, den letzten Punkt mit immer kleineren Schritten wiederholen.

[1] Default-Wert 60927: falls nötig kleineren Wert verwenden, d.h. falls der Servo die Maximalgeschwindigkeit bei diesem Standart-Wert noch nicht erreicht hat.

- Das Ergebnis ist die Vorwärts-Maximalgeschwindigkeit (UPPER bzw. LOWER je nach Drehrichtung) für Servo 0. Zusätzlich den Zählerstand zwischenspeichern.

- Servo 1:
  - Entsprechend Servo 0 die Vorwärts-Maximalgeschwindigkeit ermitteln. Falls der Servo spiegelverkehrt angebracht ist, also andersrum drehen soll, wie bei der Rückwärts-Maximalgeschwindigkeit von Servo 0 vorgehen.

- Gemeinsame Vorwärts-Maximalgeschwindigkeit:
  - Die Zählerstände bei Maximalgeschwindigkeit der Servos 0 und 1 vergleichen. Die Maximalgeschwindigkeit des Servo mit dem höheren (niedrigeren) Wert muss verringert (vergrößert) werden (je nach Drehrichtung), damit sich beide Servos in ihrer Maximalstellung gleich schnell bewegen.
  - Zu diesem Zweck den Geschwindigkeitswert des schnelleren Servos Richtung Mittelposition (MIDDLE) schrittweise verändern, bis der Zählerstand gleich dem des langsameren Servos ist. Hierbei kann sich wie bereits beschrieben in veränderbaren Schrittweiten an den gewünschten Wert herangetastet werden.

- Rückwärts-Kalibrierung:

  - Servo 0:
    - Die drei ersten Punkte der Vorwärts-Kalibrierung mit dem Default-Wert 64613[2] wiederholen. Dabei den Wert jedoch jeweils verkleinern.
    - Das Ergebnis ist die Rückwärts-Maximalgeschwindigkeit (LOWER bzw. UPPER je nach Drehrichtung) für Servo 0.

  - Servo 1:
    - Entsprechend Servo 0 die Rückwärts-Maximalgeschwindigkeit ermitteln. Falls der Servo spiegelverkehrt angebracht ist, also andersrum drehen soll, wie bei der Vorwärts-Maximalgeschwindigkeit von Servo 0 vorgehen.

  - Gemeinsame Rückwärts-Maximalgeschwindigkeit:
    - Identisch zur Vorgehensweise bei der Ermittlung der gemeinsamen Vorwärts-Maximalgeschwindigkeit.

[2] Default-Wert 64613: falls nötig größeren Wert verwenden

Nach dem Durchlaufen dieses Algorithmus sind die Kalibrierungswerte der Servos 0 und 1 bekannt. Servo 2 besitzt keine Puls-Zählung und kann nicht automatisch kalibriert werden, da die Software keinerlei Information über die aktuelle Position des Servos erhalten kann.

5 Protokoll

5.1 Spezifikation

Im Folgenden wird das Protokoll spezifiziert, mit dem die Servomotoren über den I^2C-Bus gesteuert und die Statusinformationen abgerufen werden können.

Die I^2C-Adresse der Motorsteuerung (Slave) ist voreingestellt auf 0x10. Die verwendete I^2C-Bus-Datenrate ist 100 kbit/s.

5.1.1 Aufbau der Befehle

Die Befehle setzen sich wie folgt byteweise zusammen:

Adresse	Befehl (Param0)	Param1 ... Paramn

bzw.

Adresse	Nachricht (Param0)	Param1 ... Paramn

Adresse:	I^2C-Adresse der Motorsteuerung (Slave)
Befehl:	Befehls-ID ist hier dezimal angegeben Befehls-ID < 50 (siehe Tabelle 5.2): Master -> Slave (Steuerung -> Motorsteuerung)
Nachricht:	Nachrichten-ID ist hier dezimal angegeben Nachrichten-ID $>$ **50** (siehe Tabelle 5.2): Slave -> Master (Motorsteuerung -> Steuerung, d.h. Antwort, Statusmeldung ...)
Param:	Parameter siehe Befehl-Spezifikation

Tabelle 5.1: Protokoll - Byteweiser Aufbau eines Befehls bzw. einer Nachricht

Steuerung allgemein	
RESET	1
RESET_ALL	2
ENABLE_DEBUG	3
Kalibrieren	
CALIBRATE_SET_UPPER	11
CALIBRATE_SET_MIDDLE	12
CALIBRATE_SET_LOWER	13
CALIBRATE_ADD_UPPER	14
CALIBRATE_ADD_MIDDLE	15
CALIBRATE_ADD_LOWER	16
CALIBRATE_GET_VALUES	17
CALIBRATE_VALUES	**51**
CALIBRATE_ERASE	18
CALIBRATE_WRITE_FLASH	19
CALIBRATE_SET_SPEED	31
CALIBRATE_GET_SPEED_VALUE	30
CALIBRATE_SPEED_VALUE	**55**
Steuerbefehle für die Servos	
MOVE	21
GET_STATUS	22
STATUS	**52**
STOP	23
START	24
MOVE_2	25
GET_STATUS_2	26
STATUS_2	**53**
STOP_2	27
START_2	28
SET_SPEED	29
Fehler	
ERROR	**54**

Tabelle 5.2: Protokoll - I^2C-Befehle bzw. -Nachrichten mit ID

5.1.2 Befehl-Spezifikation

Aufbau Befehl-Spezifikation

Im Folgenden wird für jeden Befehl bzw. jede Nachricht der Tabelle 5.2 der Aufbau und die Befehl-Spezifikation erläutert.

ID	**NAME_DES_BEFEHLS_ODER_DER_NACHRICHT**			
Beschreibung was dieser Befehl bewirkt bzw. die Nachricht enthält				
PARAMx	Funktion	Datentyp	erlaubte Werte	Beschreibung

Der Datentyp „unsigned char“ entspricht dem Datentyp „byte“.
ID ist als Dezimal-Wert angegeben.

Befehle Steuerung allgemein

1	**RESET**
Setzt alle Werte in den Ursprungszustand. Das beinhaltet Löschen der Warteschlange, Zurücksetzen aller Geschwindigkeiten und Ticks, Stoppen der Timer. Die Kalibrierung wird *nicht* zurückgesetzt.	
kein Parameter	

2	**RESET_ALL**
Wie **RESET**, setzt zusätzlich die Kalibrierung zurück auf die Standart-Werte aus dem Flash.	
kein Parameter	

3	**ENABLE_DEBUG**			
Aktiviert oder deaktiviert den Debug-Modus. Aktionen der Motorsteuerung werden über die serielle Schnittstelle bestätigt, falls der Debug-Modus aktiviert ist. Bei deaktiviertem Debug-Modus gibt die serielle Schnittstelle nur eine Begrüßung aus.				
Byte Nr.	Funktion	Datentyp	Werte	Beschreibung
PARAM1		unsigned char	1|0	1: aktivert, 0: deaktiviert (default)

Befehle zum Kalibrieren

11 12 13	**CALIBRATE_SET_UPPER** **CALIBRATE_SET_MIDDLE** **CALIBRATE_SET_LOWER**

Setzt den Wert für den oberen/mittleren/unteren Wert (maximale Vorwärts-Geschwindigkeit / keine Bewegung / maximale Rückwärts-Geschwindigkeit) für den angegebenen Motor. Dieser Befehl kann verwendet werden, um bereits bekannte Werte für die vorliegende Kombination Platine - Motoren in der Initialisierungsphase zu setzen. Zur Kalibrierung selbst können die Befehle **CALIBRATE_SET_UPPER**, -**MIDDLE**, -**LOWER** verwendet werden.
Achtung: Der hier gesetzte Wert wirkt erst bei der nächsten Geschwindigkeitsänderung des Motors (z.B. nächster Befehl der Warteschlange oder neues **SET_SPEED**).

Byte Nr.	Funktion	Datentyp	Werte	Beschreibung
PARAM1		unsigned char	0|1|2	Motor, der kalibriert wird
PARAM2	HI-Byte	unsigned integer	[0, 65535]	Wert
PARAM3	LOW-Byte			

Hier kann das vorgegebene struct `CALIBRATE_SET_CMD` verwendet werden.

14 15 16	**CALIBRATE_ADD_UPPER** **CALIBRATE_ADD_MIDDLE** **CALIBRATE_ADD_LOWER**

Verändert den Wert für den oberen/mittleren/unteren Wert (maximale Vorwärts-Geschwindigkeit / keine Bewegung / maximale Rückwärts-Geschwindigkeit) um den angegeben Wert für den angegebenen Motor.
Es sind Schritte in maximal -128- bzw. +127-Sprüngen möglich. Dieser Befehl kann zur Fein-Justierung verwendet werden.
Achtung: Der hier gesetzte Wert wirkt erst bei der nächsten Geschwindigkeitsänderung des Motors (z.B. nächster Befehl der Warteschlange oder neues **SET_SPEED**).

Byte Nr.	Funktion	Datentyp	Werte	Beschreibung
PARAM1		unsigned char	0|1|2	Motor, der kalibriert wird
PARAM2		signed char	[-128, 127]	Wert, um den der Kalibrierungs-Wert geändert werden soll

17	**CALIBRATE_GET_VALUES**

Fordert die aktuellen Kalibrierungswerte für den angegebenen Motor an. Die Antwort ist die Nachricht **CALIBRATE_VALUES**.

Byte Nr.	Funktion	Datentyp	Werte	Beschreibung
PARAM1		unsigned char	0|1|2	Motor, dessen Werte angefordert werden

51	**CALIBRATE_VALUES**			
Rückantwort Slave -> Master (Motorsteuerung -> Steuerung) Liefert die von **CALIBRATE_GET_VALUES** angeforderten Kalibrierungswerte für den entsprechenden Motor zurück.				
Byte Nr.	Funktion	Datentyp	Werte	Beschreibung
PARAM1		unsigned char	0|1|2	Motor, dessen Werte folgen
PARAM2	HI-Byte	unsigned integer	[0, 65535]	UPPER-Wert
PARAM3	LOW-Byte			
PARAM4	HI-Byte	unsigned integer	[0, 65535]	MIDDLE-Wert
PARAM5	LOW-Byte			
PARAM6	HI-Byte	unsigned integer	[0, 65535]	LOW-Wert
PARAM7	LOW-Byte			
Hier kann das vorgegebene struct `CALIBRATE_VALUES_MSG` verwendet werden.				

18	**CALIBRATE_ERASE**
Löscht den Kalibrierungs-Variablen-Speicher des Flash, damit neue Werte geschrieben werden können. **Achtung:** Nach Ausführung dieses Befehls bitte per Hand einen Reset durchführen! **Achtung:** Nach Ausführung dieses Befehls müssen die Kalibrierungs-Variablen neu gesetzt (**CALIBRATE_SET_xxx**) und neu gespeichert (**CALIBRATE_WRITE_FLASH**) werden, sonst werden die Default-Werte verwendet.	
kein Parameter	

19	**CALIBRATE_WRITE_FLASH**
Schreibt die aktuell eingestellten Kalibrierungs-Werte in den Flash-Speicher, so dass sie nach einem Neustart nicht verloren gehen. Wenn alte Werte überschrieben werden sollen, muss vorher unbedingt ein **CALIBRATE_ERASE** durchgeführt werden, da sonst die neuen Werte nicht korrekt gespeichert werden (die 0-Bits der alten Werte würde erhalten bleiben).	
kein Parameter	

31	**CALIBRATE_SET_SPEED**			
Setzt die Geschwindigkeit für den angegebenen Motor direkt auf den angegebenen Wert. Dieser Befehl umgeht die sonst angewendeten [-128, 127], die im normalen Betrieb innerhalb der Motorsteuerung in den hier direkt übermittelten Wert umgerechnet werden. Dieser Befehl kann verwendet werden, um Kalibrierungswerte auszuprobieren. **Achtung:** Mit diesem Befehl können die Kalibrierungsgrenzen (UPPER und LOWER) übergangen werden.				
PARAM1		unsigned char	0|1|2	Motor, dessen Wert gesetzt wird
PARAM2	HI-Byte	unsigned integer	[0, 65535]	Wert
PARAM3	LOW-Byte			
Hier kann das vorgegebene struct `CALIBRATE_SET_CMD` verwendet werden.				

30	**CALIBRATE_GET_SPEED_VALUE**			
Fordert den Wert (vom Datentyp „unsigned integer“) der aktuellen Geschwindigkeit vom angegebenen Motor an. Dieser Wert kann für die Kalibrierung nützlich sein, wenn die Geschwindigkeit beispielsweise mit **SET_SPEED** gesetzt wurde (siehe auch **CALIBRATE_SET_SPEED**). Die Antwort ist die Nachricht **CALIBRATE_SPEED_VALUE**.				
Byte Nr.	Funktion	Datentyp	Werte	Beschreibung
PARAM1		unsigned char	0|1|2	Motor, dessen Wert angefordert wird

55	**CALIBRATE_SPEED_VALUE**			
Rückantwort Slave -> Master (Motorsteuerung -> Steuerung) Liefert den von **CALIBRATE_GET_SPEED_VALUE** angeforderten Geschwindigkeits-Wert für den entsprechenden Motor zurück.				
Byte Nr.	Funktion	Datentyp	Werte	Beschreibung
PARAM1		unsigned char	0|1|2	Motor, dessen Wert folgt
PARAM2	HI-Byte	unsigned integer	[0, 65535]	Geschwindigkeits-Wert
PARAM3	LOW-Byte			
Hier kann das vorgegebene struct `CALIBRATE_SET_CMD` verwendet werden.				

Steuerbefehle für die Motoren

21	**MOVE**			
Steuerbefehl für die Motoren 0 und 1. Diese Motoren können nur gleichzeitig angesprochen werden. Diese Befehle werden in die Warteschlange eingereiht. Die Geschwindigkeit ist ein Wert zwischen -128 und 127, der den kalibrierten Maximalgeschwindigkeiten entspricht. Die Mittelposition ist 0 (der Motor steht). Die Ticks sind die Anzahl der Bewegungsschritte, die sich der Motor fortbewegen soll.				
Byte Nr.	Funktion	Datentyp	Werte	Beschreibung
PARAM1		signed char	[-128, 127]	Geschwindigkeit Motor 0
PARAM2	HI-Byte	unsigned integer	[0, 65535]	Ticks Motor 0
PARAM3	LOW-Byte			
PARAM4		signed char	[-128, 127]	Geschwindigkeit Motor 1
PARAM5	HI-Byte	unsigned integer	[0, 65535]	Ticks Motor 1
PARAM6	LOW-Byte			
Hier kann das vorgegebene struct `MOVE_CMD` verwendet werden.				

22	**GET_STATUS**
Fordert von der Motorsteuerung den Befehl der Warteschlange, der sich aktuell in der Ausführung befindet, und die vergangenen Ticks an. Die Antwort ist die Nachricht **STATUS**. Die Ausführung der Warteschlange wird nicht beeinflusst.	
kein Parameter	

52	**STATUS**

Rückantwort Slave -> Master (Motorsteuerung -> Steuerung)
Liefert den aktuellen Befehl der Warteschlange, der sich in der Ausführung befindet. Die Parameter 1 bis 6 liefern genau den Befehl wie er mit dem Befehl **MOVE** in Auftrag gegeben wurde. Die folgenden Werte sind die aktuellen Zählerstände, wie viele Ticks schon vergangen sind, das heißt wie viele Entfernungs-Schritte der Motor seit Beginn des Befehls schon zurückgelegt *hat*. Die Ausführung der Warteschlange wird nicht beeinflusst.
Wenn die Motoren mit dem Befehl **STOP** gestoppt wurden, werden an dieser Stelle für Geschwindigkeiten und Ticks Nullen zurückgegeben. Die vergangenen Ticks bleiben durch das **STOP** unbeeinflusst.

Byte Nr.	Funktion	Datentyp	Werte	Beschreibung
PARAM1		signed char	[-128, 127]	Geschwindigkeit Motor 0
PARAM2	HI-Byte	unsigned integer	[0, 65535]	Ticks Motor 0
PARAM3	LOW-Byte			
PARAM4		signed char	[-128, 127]	Geschwindigkeit Motor 1
PARAM5	HI-Byte	unsigned integer	[0, 65535]	Ticks Motor 1
PARAM6	LOW-Byte			
PARAM7	HI-Byte	unsigned integer	[0, 65535]	vergangene Ticks Motor 0
PARAM8	LOW-Byte			
PARAM9	HI-Byte	unsigned integer	[0, 65535]	vergangene Ticks Motor 1
PARAM10	LOW-Byte			

Die Parameter 1 bis 6 sind identisch mit denen des **MOVE**-Befehls.
Hier kann das vorgegebene struct `STATUS_MSG` verwendet werden.

23	**STOP**

Pausiert die Ausführung der Befehle in der Warteschlange für die Motoren 0 und 1 und verwirft den aktuell ausgeführten Befehl. Das Signal an den Motorausgängen 0 und 1 wird unterbrochen. Der in diesem Zustand zurückgegebene Status liefert für die Geschwindigkeiten und Ticks Nullen, die vergangenen Ticks bleiben unbeeinflusst. Motor 2 bleibt unbeeinflusst.

kein Parameter

24	**START**

Führt die Ausführung der Warteschlange bei dem Befehl fort, der dem mit **STOP** unterbrochenen folgt. Motor 2 bleibt unbeeinflusst.

kein Parameter

25	**MOVE_2**

Steuerbefehl für den Motor 2. Dieser Befehl wird sofort ausgeführt, also nicht in der Warteschlange der Motoren 0 und 1 geführt. Der aktuell vom Motor 2 ausgeführte Befehl wird unterbrochen. Der aktuell ausgeführte Befehl kann durch **GET_STATUS_2** abgefragt werden.
Für diesen Motor können keine Entfernungs-Schritte (Ticks) angegeben werden! Die Position (eines nicht modifizierten Servos) kann kann nur über die MOVE_2-Befehle gesetzt werden.

Byte Nr.	Funktion	Datentyp	Werte	Beschreibung
PARAM1		signed char	[-128, 127]	Position Servo 2

26	**GET_STATUS_2**

Fordert von der Motorsteuerung den aktuellen von Motor 2 ausgeführten Befehl an. Die Antwort ist die Nachricht **STATUS_2**. Die Ausführung der Warteschlange (der Motoren 0 und 1) und des Befehls für Motor 2 werden nicht beeinflusst.

kein Parameter

53	**STATUS_2**

Rückantwort Slave -> Master (Motorsteuerung -> Steuerung)
Liefert den von **GET_STATUS_2** angeforderten aktuellen von Motor 2 ausgeführten Befehl. Die Ausführung der Warteschlange (der Motoren 0 und 1) und des Befehls für Motor 2 werden nicht beeinflusst. Falls sich der Motor im STOP-Zustand befindet, wird Geschwindigkeit Null zurückgegeben.

Byte Nr.	Funktion	Datentyp	Werte	Beschreibung
PARAM1		signed char	[-128, 127]	Geschwindigkeit Motor 2

Die Parameter sind identisch mit denen des **MOVE_2**-Befehls.

27	**STOP_2**

Pausiert die Ausführung des Befehls für Motor 2. Das Signal am Motorausgang 2 wird unterbrochen. Die Motoren 0 und 1 bleiben unbeeinflusst. Der in diesem Zustand zurückgegebene Status liefert die Geschwindigkeit Null.

kein Parameter

28	**START_2**

Führt die Ausführung des aktuellen Befehls von Motor 2 fort. Die Motoren 0 und 1 bleiben unbeeinflusst.

kein Parameter

29	**SET_SPEED**			
Aktiviert sofort den entsprechenden Motor mit der angegebenen Geschwindigkeit und unendlicher Bewegung (also ohne Zählung von Entfernungs-Schritten). **Achtung:** Dieser Befehl sollte *nicht* gleichzeitig mit der Warteschlange verwendet werden (die mit dem Befehl **MOVE** gefüllt wird), da er die Befehle dort überholt und deren Ausführung verhindert. Der Befehl **SET_SPEED 2** entspricht dem Befehl **MOVE_2**.				
Byte Nr.	Funktion	Datentyp	Werte	Beschreibung
PARAM1		unsigned char	0|1|2	Motor, der gestartet wird
PARAM2		signed char	[-128, 127]	Geschwindigkeit

Der leere Befehl für keine Aktion (siehe Befehle **MOVE**, **STATUS**, **MOVE_2**, **STATUS_2**) ist durch die Parameter Geschwindigkeit=0 und Ticks=0 möglich. Eine unendliche Bewegung der Motoren 0 oder bzw. und 1 wird durch Setzen der Ticks = 0 erreicht. Diese Bewegung wird unterbrochen sobald der andere Motor seine Ticks abgearbeitet hat. Wenn beide Motoren unendliche Befehle haben (Ticks=0), wird die Ausführung abgebrochen sobald ein nachfolgender Befehl in die Warteschlange eingereiht wird.

54	**ERROR**			
Zeigt einen Fehler in der Motorsteuerung an. Die Behandlung der Fehler ist im Benutzerhandbuch beschrieben. Fehler-Code 61 = `ERROR_WRITE_FLASH`, Fehler-Code 62 = `ERROR_FLASH_IS_NOT_ERASED`, Fehler-Code 63 = `ERROR_FIFO_FULL`				
Byte Nr.	Funktion	Datentyp	Werte	Beschreibung
PARAM1		unsigned char	61|62|63	Fehler-Code
Hier kann das vorgegebene struct `ERROR_MSG` verwendet werden.				

6 Fehler und Fehlermeldungen

6.1 Fehler bei Flash Magic

Flash Magic bekommt keine Verbindung zur Motorsteuereinheit

Meldung: Flash Magic meldet „Unable to connect at the specified baud rate. [...]“.

Maßnahme:

- Der DIP-Schalter auf der Motorsteuer-Platine muss mit beiden Schaltern auf „ON“ („programm“) stehen.
- Die Baud-Rate unter Ziffer 1 muss 19200 betragen.
- Die Meldung erscheint trotz richtig eingestellten DIP-Schalter: Den Reset-Taster betätigen und in Flash Magic den Verbindungsversuch wiederholen.
- Überprüfen, ob das Kabel zwischen PC und Motorsteuer-Platine fest verbunden ist.

Falsches Programm hochgeladen

Meldung: Änderungen am Programm scheinen trotz wiederholtem Hochladen mit Flash Magic keine Wirkung zu zeigen.

Maßnahme: Flash Magic trägt unter Ziffer 3 die zuletzt hochgeladene HEX-Datei ein. Eventuell ist es die falsche.

6.2 Fehler beim Motortest

Motor-Testprogramm startet nicht automatisch

Meldung: Nach dem Starten des Debug-Modus wird das Programm ohne Fehlermeldung in den Mikrocontroller der Hauptplatine hochgeladen, das Programm meldet sich dann aber nicht im Fenster „Serial“ mit der Zeile „Motortestprogramm“.

Maßnahme: Die Initialisierungsdatei wird nicht ausgeführt. Das Programm lässt sich per Hand starten mit der Eingabe `g 0x2000` in das Eingabefeld von „Output Window“.

Fehler-Code empfangen

Meldung: Nach Eingabe von receive <anzahl> zeigt der Motortest einen Fehler (**ERROR**) mit einem Fehler-Code an.

Maßnahme: Einer der Fehlerfälle der Motorsteuerung ist aufgetreten. Die Behandlung der Fehler ist in Abschnitt 6.3 beschrieben.

6.3 Fehler und Fehlernachrichten der Motorsteuerung

Fehler-Code 61 ERROR_WRITE_FLASH

Meldung: Nach der Übermittlung des Befehls **CALIBRATE_WRITE_FLASH** über den I^2C-Bus an die Motorsteuerung wird der Fehler (**ERROR**) mit dem Fehler-Code 61 (**ERROR_WRITE_FLASH**) empfangen.

Maßnahme: Das Schreiben von neuen Kalibrierungswerten ist fehlgeschlagen. Es liegt ein Fehler im Flash bzw. der In-Application-Programming-Methode vor. Mit Flash Magic sollte ein vollständiges Löschen des Flash-Speichers (inklusive Block 3) und neues Hochladen des Programms durchgeführt werden. Erscheint der Fehler wieder, liegt eventuell ein fehlerhafter Mikrocontroller vor oder der Programmcode ist defekt.

Fehler-Code 62 ERROR_FLASH_IS_NOT_ERASED

Meldung: Nach der Übermittlung des Befehls **CALIBRATE_WRITE_FLASH** über den I^2C-Bus an die Motorsteuerung wird der Fehler (**ERROR**) mit dem Fehler-Code 62 (**ERROR_FLASH_IS_NOT_ERASED**) empfangen.

Maßnahme: Vor dem Schreiben von neuen Kalibrierungswerten müssen die bisherigen im Flash-Speicher gelöscht werden. Folglich muss zuerst der Befehl **CALIBRATE_ERASE** ausgeführt werden. Anschließend werden die Werte neu übertragen und erst dann mit **CALIBRATE_WRITE_FLASH** dauerhaft gespeichert.

Fehler-Code 63 ERROR_FIFO_FULL

Meldung: Nach der Übermittlung des Befehls **MOVE** über den I^2C-Bus an die Motorsteuerung wird der Fehler (**ERROR**) mit dem Fehler-Code 63 (**ERROR_FIFO_FULL**) empfangen.

Maßnahme: Der Warteschlangenspeicher des Mikrocontrollers der Motorsteuerung ist belegt, der letzte übermittelte **MOVE**-Befehl wurde verworfen. Die Anwendung muss mit der weiteren Übermittlung warten bis Befehle ausgeführt wurden. In der Standart-Einstellung fasst die Warteschlange 100 Einträge.

Motorsteuerung reagiert nicht

Meldung: Die Motorsteuerung reagiert nicht auf gesendete I^2C-Bus-Befehle.

Maßnahme: Durch Drücken des Reset-Tasters an der Motorsteuer-Platine lässt sich überprüfen, ob das Programm ausgeführt wird: Beim Betätigen führen die Servos kurze Initialisierungs-Bewegungen durch. Die Aktionen der Motorsteuerung lassen sich über die serielle Schnittstelle mit einem Terminal überprüfen, wenn zuvor der Befehl **ENABLE_DEBUG** gesendet wird.

Motorsteuerung führt einige Befehle nicht aus

Meldung: Von einer Reihe schnell nacheinander übertragener Befehle werden einige nicht ausgeführt.

Maßnahme: Eventuell werden die Befehle zu schnell nacheinander übertragen, so dass die Einreihung in die Warteschlange (bei **MOVE**-Befehlen) oder die Bearbeitung (bei anderen Befehlen) nicht abgeschlossen werden kann, bevor der nachfolgende Befehl über den I^2C-Bus eintrifft. Der Mindestabstand zwischen zwei **MOVE**-Befehlen und Kriterien, anhand denen die erfolgreiche Übermittlung eines Befehl festgestellt werden kann, sind Abschnitt 4.1 zu entnehmen.

6.4 Fehler im HyperTerminal

Keine Verbindung zum COM-Port

Meldung: COMx konnte nicht geöffnet werden.

Maßnahme:

- Wenn ein anderes Programm (z.B. Flash Magic) den gleichen COM-Port verwendet, muss dieses vor dem Start des HyperTerminals beendet werden.
- Der in den Einstellungen des HyperTerminals angegebene COM-Port muss der richtige sein.

Keine Ausgabe der Motorsteuereinheit

Meldung: Obwohl das HyperTerminal ohne Fehlermeldung gestartet hat, zeigt es keine Ausgaben der Motorsteuereinheit an.

Maßnahme:

- Der DIP-Schalter auf dem Board der Motorsteuerung muss auf „run“ eingestellt sein. Anschließend wird zum Starten der Reset-Schalter betätigt.
- Wenn sich die Motorsteuerung nicht im Debug-Modus befindet (siehe Befehle Abschnitt 4.4 und 5), erscheint nur eine Begrüßung mit Version und Datum nach einem Reset. Zur Kontrolle den Reset-Taster betätigen und Ausgabe überprüfen.

7 Board und Anschlüsse

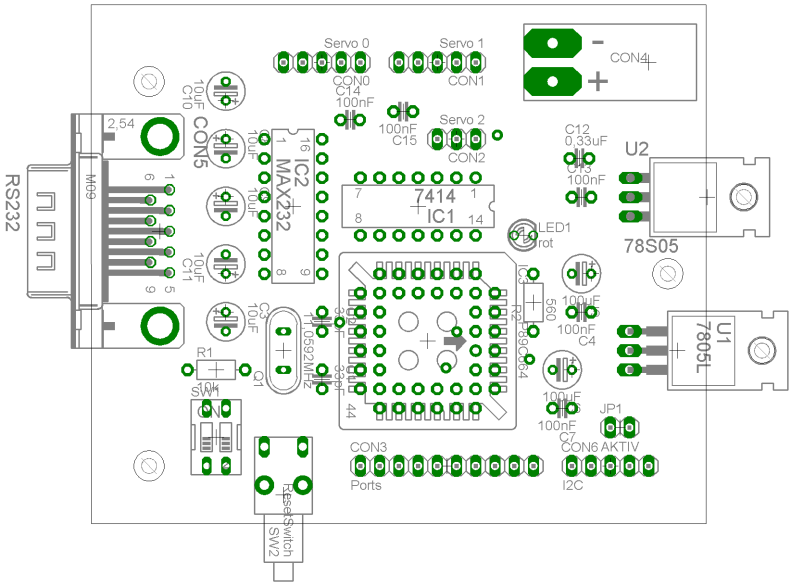

Abbildung 7.1: Bauteile-Ansicht

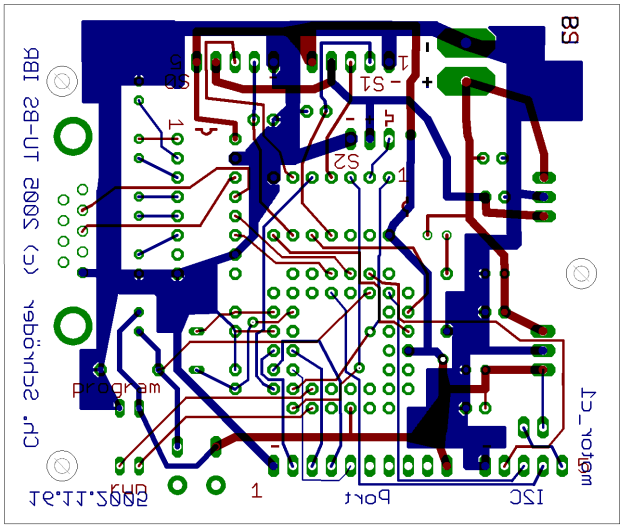

Abbildung 7.2: Löt- und Bestückungsseiten

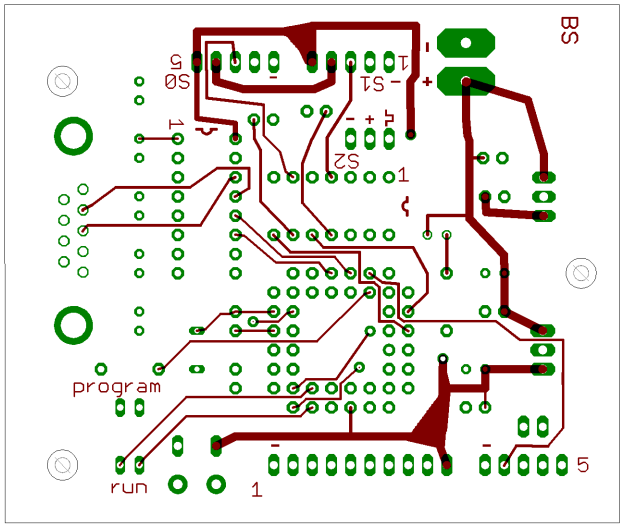

Abbildung 7.3: Bestückungsseite

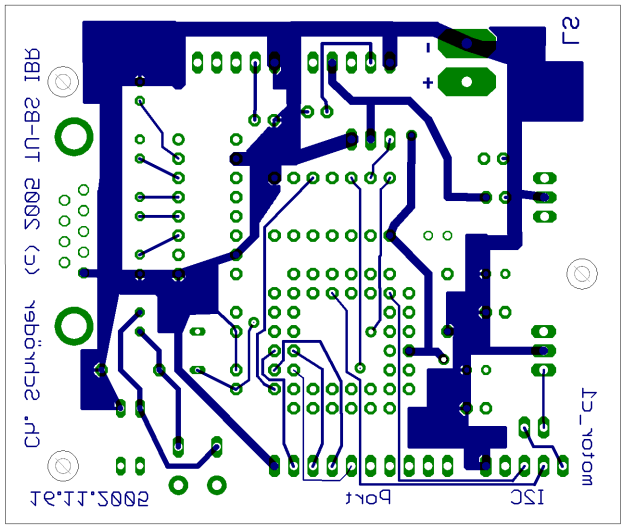

Abbildung 7.4: Lötseite

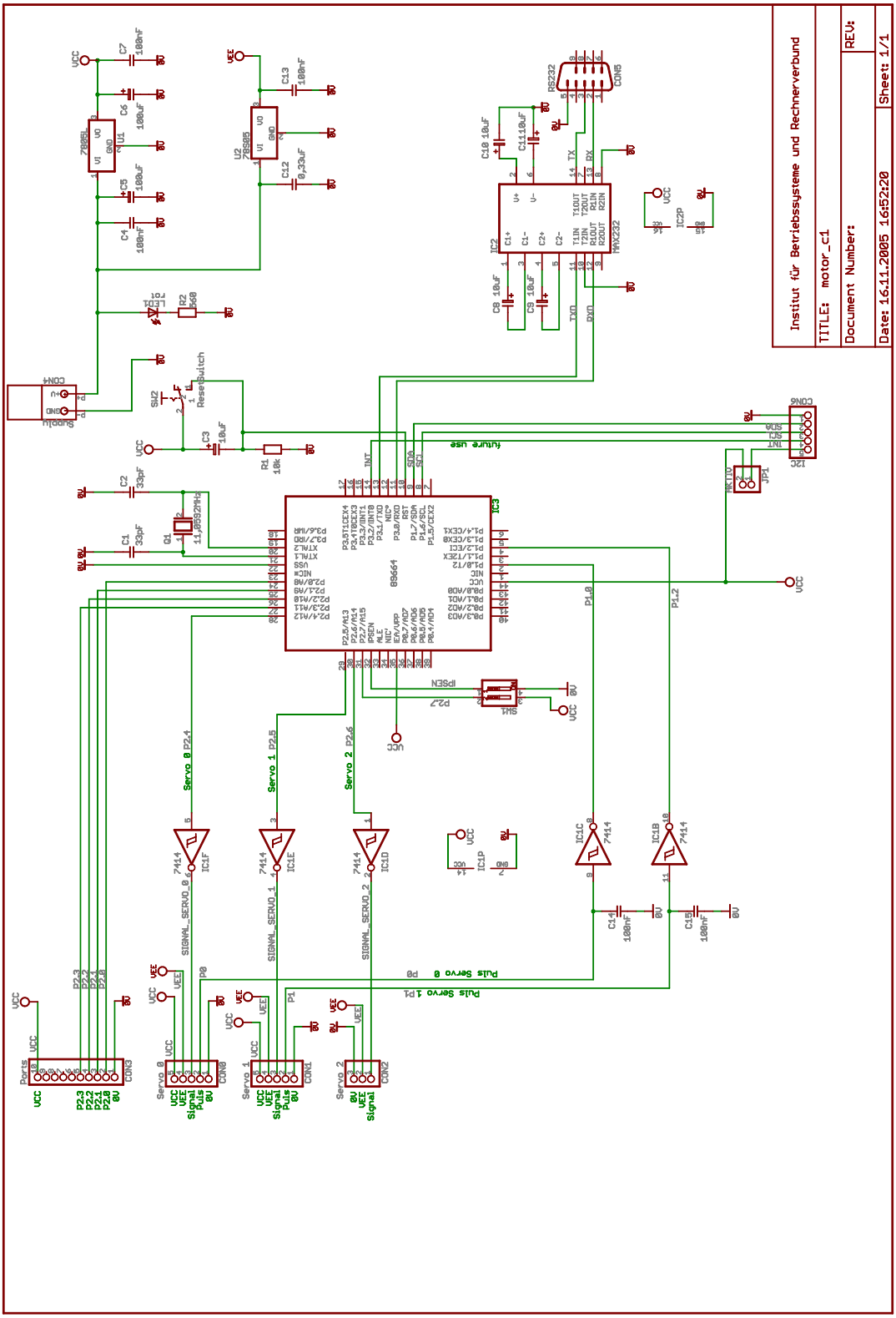

Abbildung 7.5: Schaltbild [Diese Seite wird ersetzt durch faltbare DIN A3-Seite mit Schaltbild im Querformat]

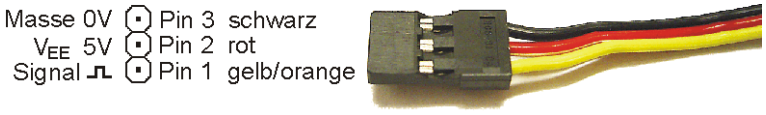

Abbildung 7.6: Anschluss-Schema nicht modifizierter Servo (Servo 2)

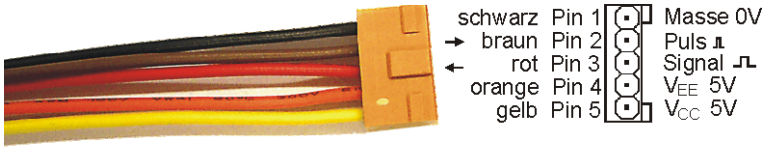

Abbildung 7.7: Anschluss-Schema modifizierter Servo (Servo 0 oder Servo 1)

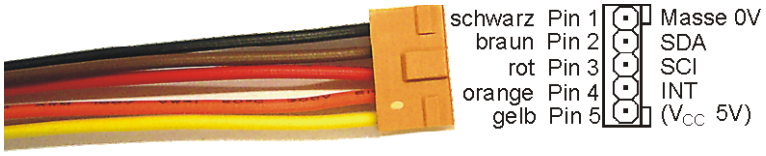

Abbildung 7.8: Anschluss-Schema I^2C-Steckverbinder

Details und Beschreibungen zu sämtlichen Abbildungen in diesem Kapitel sind der Studienarbeit „Entwicklung einer Motorsteuereinheit für ein Fahrmodul“[1] zu entnehmen.

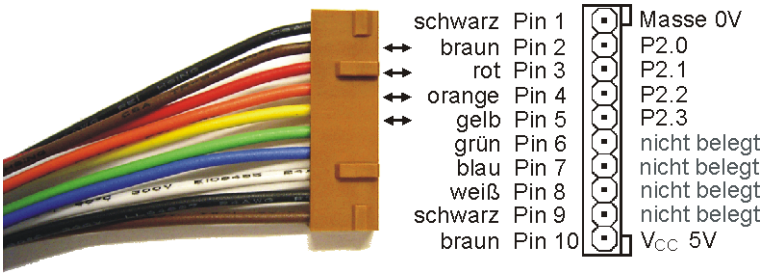

Abbildung 7.9: Anschluss-Schema Port-Steckverbinder

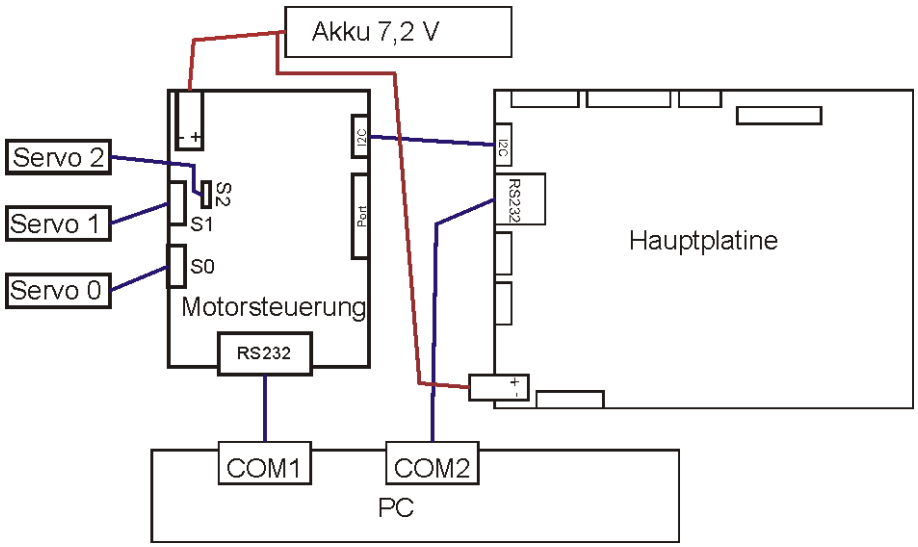

Abbildung 7.10: Schematischer Anschluss der Motorsteuereinheit an das Fahrmodul

Literaturverzeichnis

[1] Christian Schröder: Entwicklung einer Motorsteuereinheit für ein Fahrmodul. Institut für Betriebssysteme und Rechnerverbund, Technische Universität Braunschweig, 2005.
http://www.ibr.cs.tu-bs.de/theses/broeke/SA_Motorsteuereinheit_Prakt/Dokumente.html

[2] Philips Semiconductors: Data Sheet P89C660/P89C662/P89C664/P89C668. Oktober 2002, U.S.A.

Index