Protokoll Motorsteuereinheit

Christian Schröder

17. November 2005

1 Protokoll

1.1 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.

1.1.1 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 1.2): Master -> Slave (Steuerung -> Motorsteuerung)
Nachricht:	Nachrichten-ID ist hier dezimal angegeben Nachrichten-ID $>$ **50** (siehe Tabelle 1.2): Slave -> Master (Motorsteuerung -> Steuerung, d.h. Antwort, Statusmeldung ...)
Param:	Parameter siehe Befehl-Spezifikation

Tabelle 1.1: 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 1.2: Protokoll - I^2C-Befehle bzw. -Nachrichten mit ID

1.1.2 Befehl-Spezifikation

Aufbau Befehl-Spezifikation

Im Folgenden wird für jeden Befehl bzw. jede Nachricht der Tabelle 1.2 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	**CALIBRATE_ADD_UPPER**
15	**CALIBRATE_ADD_MIDDLE**
16	**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.				