# Specification for Communication Protocol

## Contents


**1 Packet Structur** **1**
- 1.1 Command Structur . . . 1
- 1.2 Response Structur . . . 2

**2 Command List** **3**
- 2.1 Generic Commands . . . 3
- 2.2 Specific Commands . . . 4
- 2.3 InitMove (MOTOR, DIR) . . . 5
- 2.4 MoveTo (MOTOR, DIR, ABS_POS, SPEED, ACC, DEC) . . . 5
- 2.5 WaitMoved (MOTOR, TIMEOUT) . . . 6
- 2.6 IsReady (MOTOR) . . . 6
- 2.7 Move (MOTOR, DIR, SPEED, ACC, DEC) . . . 6
- 2.8 StopMove (MOTOR, IS_HARDSTOP) . . . 7
- 2.9 GetAbsPos (MOTOR) . . . 7
- 2.10 SetPin (PIN_NR,IS_HIGH) . . . 8
- 2.11 GetPin (PIN_NR) . . . 8
- 2.12 ConfigPin (PIN_NR, IS_OUTPUT) . . . 8
- 2.13 SaveHome (MOTOR) . . . 9
- 2.14 GoHome (MOTOR) . . . 9
- 2.15 SaveWayPoint (MOTOR) . . . 9
- 2.16 MoveToWayPoint (MOTOR, WAY_POINT, SPEED, ACC, DEC) . . . 9

**3 Errorcodes** **10**


## Definitions

FALSE := 0x00
TRUE := 0x01..0xFF

# 1 Packet Structur

## 1.1 Command Structur

| Length (1 Byte) | Payload (0..13 Bytes) | Padding (13 Bytes - Length) | Checksum (1 Byte) |
|---|---|---|---|

Length: Länge des Befehls in Byte
Payload: Befehlsfolge
Padding: Füllbytes
*Checksum*: *Prüfsumme CRC8(Length + Payload) (optional)*

## 1.2 Response Structur

| Ack<br>(1 Byte) | Length<br>(1 Byte) | Payload<br>(0..13 Bytes) | Padding<br>(13 Bytes - Length) | Checksum<br>(1 Byte) |
|---|---|---|---|---|

Ack: Acknowledge (TRUE wenn Befehl gültig)
Length: Länge der Antwort in Byte
Payload: Antwort
Padding: Füllbytes
*Checksum*: *Prüfsumme CRC8(Length + Payload) (optional)*

# 2 Command List

## 2.1 Generic Commands

| Command mnemonic | Command Bytes (Payload) | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 | Byte 8 | Byte 9 | Byte 10 | Byte 11 | Byte 12 |
| InitMove | *0x??* | Motor | Dir | | | | | | | | | | |
| MoveTo | *0x??* | Motor | Dir | AbsPos (HB) | AbsPos | AbsPos (LB) | Speed (HB) | Speed | Speed (LB) | Acc (HB) | Acc (LB) | Dec (HB) | Dec (LB) |
| WaitMoved | *0x??* | Motor | Timeout (HB) | Timeout (LB) | | | | | | | | | |
| IsReady | *0x??* | Motor | | | | | | | | | | | |
| Move | *0x??* | Motor | Dir | Speed (HB) | Speed | Speed (LB) | Acc (HB) | Acc (LB) | Dec (HB) | Dec (LB) | | | |
| StopMove | *0x??* | Motor | IsHard Stop | | | | | | | | | | |
| GetAbsPos | *0x??* | Motor | | | | | | | | | | | |
| SetPin | *0x??* | PinNr | IsHigh | | | | | | | | | | |
| GetPin | *0x??* | PinNr | | | | | | | | | | | |
| ConfigPin | *0x??* | PinNr | IsOutput | | | | | | | | | | |

## 2.2 Specific Commands

| Command mnemonic | Command Bytes (Payload) | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 | Byte 8 | Byte 9 | Byte 10 | Byte 11 | Byte 12 |
| SaveHome | *0x??* | Motor | | | | | | | | | | | |
| GoHome | *0x??* | Motor | | | | | | | | | | | |
| SaveWayPoint | *0x??* | Motor | | | | | | | | | | | |
| MoveToWayPoint | *0x??* | Motor | WayPoint | Speed (HB) | Speed | Speed (LB) | Acc (HB) | Acc (LB) | Dec (HB) | Dec (LB) | | | |

## 2.3 InitMove (MOTOR, DIR)

Command code: **0x??**

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Byte 0 | | | | | | | | |
| Byte 1 | MOTOR | | | | | | | |
| Byte 2 | DIR | | | | | | | |

**Description**
Der gewählte MOTOR fährt zur Initialisierung in der angegebenen Richtung (DIR) bis an eine Endposition (zB. mechanischer Anschlag). Sobald diese Position erreicht ist, bleibt der Motor stehen und ist bereit.

## 2.4 MoveTo (MOTOR, DIR, ABS_POS, SPEED, ACC, DEC)

Command code: **0x??**

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Byte 0 | | | | | | | | |
| Byte 1 | MOTOR | | | | | | | |
| Byte 2 | DIR | | | | | | | |
| Byte 3 | ABS_POS (High Byte) | | | | | | | |
| Byte 4 | ABS_POS (Middle Byte) | | | | | | | |
| Byte 5 | ABS_POS (Low Byte) | | | | | | | |
| Byte 6 | SPEED (High Byte) | | | | | | | |
| Byte 7 | SPEED (Middle Byte) | | | | | | | |
| Byte 8 | SPEED (Low Byte) | | | | | | | |
| Byte 9 | ACC (High Byte) | | | | | | | |
| Byte 10 | ACC (Low Byte) | | | | | | | |
| Byte 11 | DEC (High Byte) | | | | | | | |
| Byte 12 | DEC (Low Byte) | | | | | | | |

**Description**
Der gewählt MOTOR fährt zu eine bestimmten absoluten Position (ABS_POS). Dabei kann dem Motor eine Richtung (DIR) sowie eine Geschwindigkeit (SPEED in $\frac{Steps}{s}$) vorgegeben werden. Des weiteren kann eine Beschleunigungsrampe (ACC in $\frac{Steps}{s^2}$) und eine Verzögerungsrampe vorgegeben werden (DEC in $\frac{Steps}{s^2}$).
Falls Standardwerte für Geschwindigkeit, Beschleunigung und Verzögerung verwendet werden sollen, können die jeweiligen Parameter auf 0x00 gesetzt werden.

## 2.5 WaitMoved (MOTOR, TIMEOUT)

Command code: **0x??**

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Byte 0 | | | | | | | | |
| Byte 1 | MOTOR | | | | | | | |
| Byte 2 | TIMEOUT (High Byte) | | | | | | | |
| Byte 3 | TIMEOUT (Low Byte) | | | | | | | |

**Description**
Bei diesem Befehl wird die Antwort erst zurück gesendet, wenn der spezifizierte MOTOR still steht. Zusätzlich muss ein TIMEOUT (in $ms$) mitgegeben werden, nach welchem die Antwort auf jeden Fall gesendet wird. Falls das Timeout abgelaufen ist, wird ein Errorcode mitgesendet.

## 2.6 IsReady (MOTOR)

Command code: **0x??**

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Byte 0 | | | | | | | | |
| Byte 1 | MOTOR | | | | | | | |

**Description**
Sendet mit der Antwort, ob der spezifizierte MOTOR bereit ist. Die Antwort ist TRUE, wenn der Motor keinen aktuellen Befehl am ausführen ist, andernfalls ist sie FALSE.

## 2.7 Move (MOTOR, DIR, SPEED, ACC, DEC)

Command code: **0x??**

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Byte 0 | | | | | | | | |
| Byte 1 | MOTOR | | | | | | | |
| Byte 2 | DIR | | | | | | | |
| Byte 3 | SPEED (High Byte) | | | | | | | |
| Byte 4 | SPEED (Middle Byte) | | | | | | | |
| Byte 5 | SPEED (Low Byte) | | | | | | | |
| Byte 6 | ACC (High Byte) | | | | | | | |
| Byte 7 | ACC (Low Byte) | | | | | | | |
| Byte 8 | DEC (High Byte) | | | | | | | |
| Byte 9 | DEC (Low Byte) | | | | | | | |

**Description**
Der gewählt MOTOR fährt mit der angegebenen Geschwindigkeit (SPEED in $\frac{Steps}{s}$) in die Richtung DIR. Es kann eine Beschleunigungsrampe (ACC in $\frac{Steps}{s^2}$) und eine Verzögerungsrampe vorgegeben werden (DEC in $\frac{Steps}{s^2}$).
Falls Standardwerte für Geschwindigkeit, Beschleunigung und Verzögerung verwendet werden sollen, können die jeweiligen Parameter auf 0x00 gesetzt werden.

## 2.8 StopMove (MOTOR, IS_HARDSTOP)

Command code: **0x??**

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Byte 0 | | | | | | | | |
| Byte 1 | MOTOR | | | | | | | |
| Byte 2 | IS_HARDSTOP | | | | | | | |

**Description**
Stoppt den MOTOR. Mit dem zusätzlichen Parameter IS_HARDSTOP kann angegeben werden, ob der Motor auf der Stelle stoppt (IS_HARDSTOP = TRUE) oder mit der programmierten Verzögerungsrampe abbremst (IS_HARDSTOP = FALSE).

## 2.9 GetAbsPos (MOTOR)

Command code: **0x??**

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Byte 0 | | | | | | | | |
| Byte 1 | MOTOR | | | | | | | |

**Description**
Fragt die aktuelle absolute Position des MOTORs ab. Die absolute Position wird in der Antwort mit 3 Bytes beschrieben, wobei das höchste Byte als erstes geschickt wird.

## 2.10 SetPin (PIN_NR,IS_HIGH)

Command code: **0x??**

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Byte 0 | | | | | | | | |
| Byte 1 | PIN_NR | | | | | | | |
| Byte 1 | IS_HIGH | | | | | | | |

**Description**
Dieser Befehlt setzt einen IO-Pin (PIN_NR) des $\mu$C auf den entsprechenden Pegel (IS_HIGH). Der Pin wird auf Vcc gesetzt, wenn der Parameter IS_HIGH = TRUE ist, andernfalls auf 0V. Der IO-Pin muss als Output definiert sein. Wenn der gewählte Pin nicht gesetzt werden kann, enthält die Antwort einen Errorcode.

## 2.11 GetPin (PIN_NR)

Command code: **0x??**

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Byte 0 | | | | | | | | |
| Byte 1 | PIN_NR | | | | | | | |

**Description**
Mit diesem Befehl kann der Wert eines IO-Pins (PIN_NR) des $\mu$C abgefragt werden. In der Antwort ist der Zustand des Pins direkt enthalten (Low = FALSE, High = 0x01).
Wenn der gewählte Pin nicht gelesen werden kann, enthält die Antwort einen Errorcode.

## 2.12 ConfigPin (PIN_NR, IS_OUTPUT)

Command code: **0x??**

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Byte 0 | | | | | | | | |
| Byte 1 | PIN_NR | | | | | | | |
| Byte | IS_OUTPUT | | | | | | | |

**Description**
Der gewählte IO-Pin (PIN_NR) des $\mu$Cs wird anhand des zweiten Parameters (IS_OUTPUT) konfiguriert. Der Pin wird zum Output, wenn IS_OUPUT = TRUE ist, andernfalls zum Input.

## 2.13 SaveHome (MOTOR)

Command code: **0x??**

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Byte 0 | | | | | | | | |
| Byte 1 | MOTOR | | | | | | | |

**Description**
Speichert die Home-Position für den gewählten MOTOR.

## 2.14 GoHome (MOTOR)

Command code: **0x??**

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Byte 0 | | | | | | | | |
| Byte 1 | MOTOR | | | | | | | |

**Description**
Der entsprechende MOTOR fährt auf die Home-Position

## 2.15 SaveWayPoint (MOTOR)

Command code: **0x??**

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Byte 0 | | | | | | | | |
| Byte 1 | MOTOR | | | | | | | |

**Description**
Speichert einen Wegpunkt für den gewählten MOTOR. In der Antwort ist die Nummer des gespeicherten Wegpunktes enthalten (max. 255).

## 2.16 MoveToWayPoint (MOTOR, WAY_POINT, SPEED, ACC, DEC)

Command code: **0x??**

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Byte 0 | | | | | | | | |
| Byte 1 | MOTOR | | | | | | | |
| Byte 2 | WAY_POINT | | | | | | | |
| Byte 3 | SPEED (High Byte) | | | | | | | |
| Byte 4 | SPEED (Middle Byte) | | | | | | | |
| Byte 5 | SPEED (Low Byte) | | | | | | | |
| Byte 6 | ACC (High Byte) | | | | | | | |
| Byte 7 | ACC (Low Byte) | | | | | | | |
| Byte 8 | DEC (High Byte) | | | | | | | |
| Byte 9 | DEC (Low Byte) | | | | | | | |

**Description**
Der MOTOR fährt zum angegebenen Wegpunkt (WAY_POINT). Dabei kann die Geschwindigkeit ((SPEED in $\frac{Steps}{s}$)), sowie eine Beschleunigungsrampe (ACC in $\frac{Steps}{s^2}$) und Verzögerungsrampe (DEC in $\frac{Steps}{s^2}$) vorgegeben werden.
Falls Standardwerte für Geschwindigkeit, Beschleunigung und Verzögerung verwendet werden sollen, können die jeweiligen Parameter auf 0x00 gesetzt werden.

# 3 Errorcodes