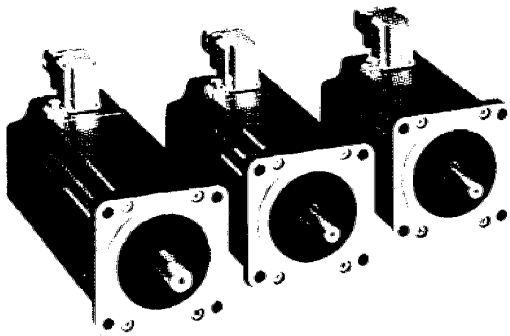


**Schrittmotoren ohne  
Zusatz-Hardware am  
Personal Computer  
mit Ihren eigenen  
Anwendungsprogrammen**



***X4Step***

**Funktionenbibliothek  
zur Verwendung in  
C-Programmen**

**Version 1.0  
November 1995**

Vervielfältigungen der Diskette und des Handbuchs von **X4Step**, sowie Veränderungen am Handbuch und den einzelnen Dateien (außer den Beispielen) sind nicht gestattet. Alle Rechte an den Programmen und am Handbuch, insbesondere das Urheberrecht, liegen bei den Autoren.

Programme, in denen **X4Step** verwendet wird, müssen mit einem Hinweis auf diese Verwendung versehen werden. Bei Vermarktung und Benutzung dieser Programme bestehen keine weitergehende Ansprüche unsererseits.

Die vorliegende Funktionenbibliothek wurde sehr aufwendig getestet. Eine Garantie für fehlerfreie Funktion in allen Anwendungen und Kombinationen kann dennoch nicht gegeben werden. Die Autoren sichern zu, daß **X4Step** im Sinne der Beschreibung und Benutzungsanleitung grundsätzlich für den vorgesehenen Zweck geeignet ist. Eine Haftung für alle bei der Benutzung von **X4Step** entstehenden Schäden wird nicht übernommen. Jede Haftung für Folgeschäden oder Schäden aus entgangenem Gewinn, Betriebsunterbrechung, Verlust von Informationen usw. ist ausgeschlossen.

Da sich Fehler, trotz aller Bemühungen, nie ganz vermeiden lassen, sind wir für jeden Hinweis dankbar.

Burkhard Lewetz  
Hardware-Software  
Brückenstrasse 7  
D-88074 Meckenbeuren  
Tel. (07542) 21886  
FAX: (07542) 3889  
eMail Lewetz@T-Online.de

November 1995

MS-DOS, MS-Windows sind eingetragene Warenzeichen der Microsoft Corporation.  
IBM ist ein eingetragenes Warenzeichen der International Business Machines Corporation.  
Turbo C und Borland C sind ein eingetragenes Warenzeichen von Borland International, INC.  
Andere namentlich genannten Produkte sind Warenzeichen oder eingetragene Warenzeichen ihrer jeweiligen Firmen.

1.000

# Inhalt

<b>1.</b>	<b>Überblick .....</b>	<b>4</b>
<b>2.</b>	<b>Funktionsweise .....</b>	<b>6</b>
<b>3.</b>	<b><i>X4Step</i> und eigene Applikationen .....</b>	<b>10</b>
3.1.	Installation .....	10
3.2.	Verwendung mit Turbo C und Borland C .....	11
3.3.	Verwendung mit Microsoft C .....	11
3.4.	Beispielprogramm .....	12
<b>4.</b>	<b>Referenz .....</b>	<b>15</b>
4.1.	Mögliche Fehlercodes als Rückgabewerte der Funktionen .....	15
4.2.	Beschreibung der Funktionen .....	16
	X4Init .....	16
	X4ReInit .....	17
	X4Load .....	18
	X4Start .....	20
	X4Brake .....	21
	X4Stop .....	22
	X4GetState .....	23
	X4GetSteps .....	24
	X4MovePos .....	25
	X4MoveNeg .....	25
	X4BrakeCodes .....	26
	X4LastBrakeCode .....	28
	X4ClearTimer .....	29
	X4GetTimer .....	29
	X4GetIn .....	30
	X4SetOut .....	31
<b>5.</b>	<b>Anhang .....</b>	<b>32</b>
5.1.	Steckerbelegung .....	32
5.2.	Signal-Timing .....	33
5.3.	Einschränkungen und Grenzen von <i>X4Step</i> .....	33

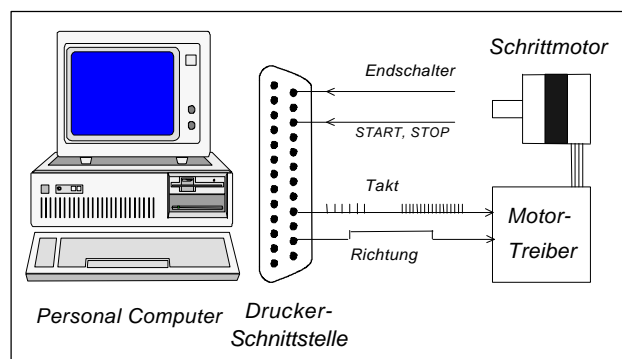
---

# 1. Überblick

---

## *4 Schrittmotoren ohne Zusatz- Hardware*

Die Funktionenbibliothek **X4Step** wurde entwickelt, um die Ansteuerung von bis zu 4 Schrittmotoren in beliebigen Anwendungsprogrammen zu ermöglichen. Es ist weder ein externer Prozessor noch sonstige intelligente Zusatz-Hardware notwendig. Die Motorendstufen werden direkt über die parallele Druckerschnittstelle des PCs mit Takt- und Richtungssignalen bedient.



## *Realisierung unterschiedlicher Anlagen*

Mit 4 Schrittmotorachsen kann man beliebige CNC-Mechaniken steuern, z.B. Roboter, Fräs- und Bohrtische, Stift- und Schneidplotter, Dispensieranlagen, Walzenvorschübe und sonstige Werkzeug- oder Sondermaschinen.

## *Highlights*

### **Die Besonderheiten von X4Step sind :**

- Ansteuerung von 1-4 Schrittmotoren direkt an der parallelen Schnittstelle des PCs
- verwendet LPT1 bis LPT4
- läuft auf allen IBM kompatiblen PCs ab 286
- Motoren in allen Leistungsklassen steuerbar
- Schrittausgabe im Hintergrund (quasi Multitasking-Betrieb)
- Schrittfrequenzen von 10 Hz bis 20 kHz
- Rampenzeiten von 0 bis 1000 ms
- Wege von 1 bis 16 Mio Schritten pro Positioniervorgang oder Endlosfahrt
- während einer Positionierung kann man bereits die nächsten Wege nachladen
- aktuell zurückgelegte Schritte jederzeit auslesbar
- alle Achsen fahren linear interpoliert zueinander

- Zusatzsignale AKTIV und BOOST und zwei weitere frei belegbare Ausgänge
- Einlesen von 5 Eingängen als externe Signale, z.B. START, STOP oder Endschalter
- 1 ms Timer jederzeit verfügbar
- Positionierung mit definierbaren Tasten abbrechbar
- Stop der Fahrt mit definierter Bremsrampe oder als NOT-Stop ohne Rampe
- Unterstützung von Turbo C, Borland C, Power C und Microsoft C

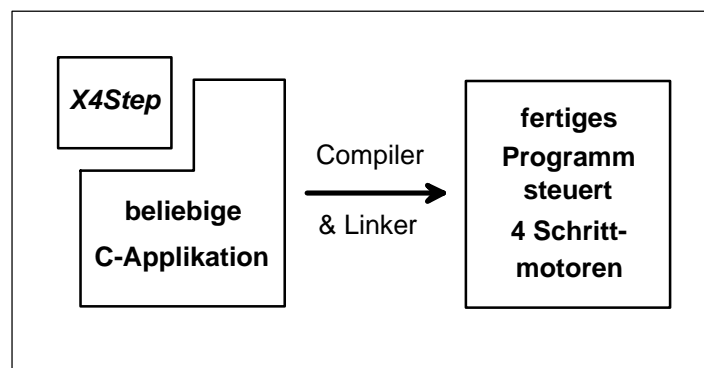
---

## 2. Funktionsweise

---

*Einbinden von X4Step*

**X4Step** ist eine Funktionenbibliothek, die Sie in Ihre eigenen C-Applikationen einbinden können. Die Ansteuerung von bis zu 4 Schrittmotoren wird dadurch zum Kinderspiel.



*X4Step starten und beenden mit speziellen Funktionen*

Um **X4Step** verwenden zu können, müssen Sie lediglich mit der Initialisierungsfunktion **X4Init** die Steuerfunktionen aktivieren.

Nach der Initialisierung, bei der die internen Daten angelegt, die benötigten Interrupts belegt und die Schnittstelle vorbereitet wird, können alle weiteren Funktionen verwendet und die Schrittmotoren gemäß Ihrer Applikation bewegt werden.

Am Ende des Programms, d.h. vor dem Rücksprung zum Betriebssystem müssen die bei der Initialisierung erledigten Arbeiten unbedingt wieder rückgängig gemacht werden. Dies erfolgt mit dem Aufruf der Funktion **X4ReInit**.



---

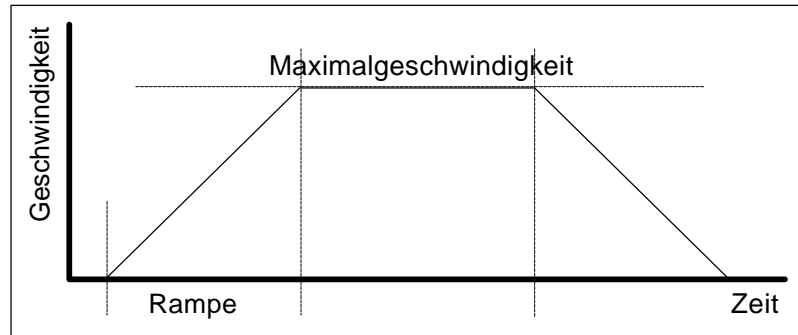
**Wenn die Reinitialisierung vergessen wird, kommt es unter Umständen zu einem Rechnerabsturz.**

---

*Beispiel*

```
main()
{
    X4Init(LPT1);
    // andere Funktionen
    ....
    X4ReInit();
}
```

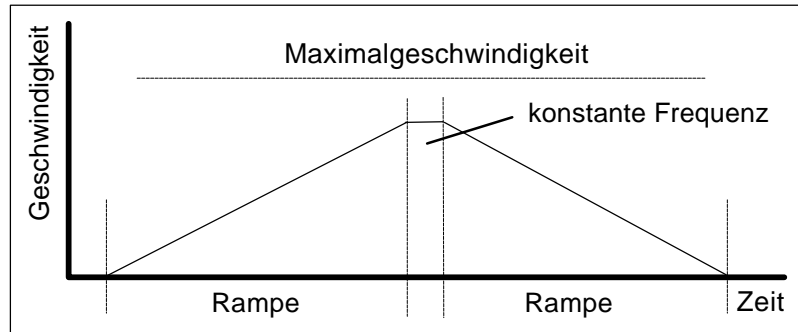
Der Fahrvorgang der einzelnen Achsen erfolgt mit einem speziellen Algorithmus. Je nach Parameter berechnet **X4Step** die Rampen und Geschwindigkeiten für jede Bewegung. Die Rampen sind in Millisekunden und die Zielfrequenz in Hertz definierbar.



*Verlauf der Positionierung*

Die Rampen werden immer symmetrisch ausgeführt, d.h. die Beschleunigungsrampe ist genauso lang wie die Bremsrampe.

Wenn die zu fahrenden Wege zu kurz sind, um die Maximalgeschwindigkeit zu erreichen, werden trotzdem nach der Beschleunigung einige Schritte mit konstanter Frequenz gefahren. Dies verhindert ein Abreisen der Motoren im Übergang von der Beschleunigungs- zur Bremsrampe.



*interpolierte Fahrt aller Achsen*

Beim Ansteuern von mehreren Achsen wird die Achse mit dem längsten Weg immer zur Führungsachse, die mit der definierten Frequenz fährt. Alle anderen Achsen fahren interpoliert zur Führungsachse mit maximal der gleichen, meist aber einer kleineren Frequenz.

Die interpolierte Fahrt aller Achsen hat den Vorteil, daß alle Motoren gleichzeitig losfahren und wieder gleichzeitig stehenbleiben, unabhängig von der Länge des Verfahrwegs.

Die Fahrt mehrerer Achsen muß mit einem Befehl vorbereitet werden (**X4Load**). Hierbei erfolgt die Berechnung der Rampen und Schrittfrequenzen. Die eigentliche Positionierung beginnt dann mit dem Startbefehl (**X4Start**).

Sofort nach dem Start können bereits die nächsten Wege nachgeladen werden. Auf diese Weise ist die Totzeit zwischen den Positionierungen auf ein Minimum reduziert.

#### *Beispiel*

```
...
X4Load(1000,800,600,400,1000,50);    Achsen laden
X4Start();                            starten
X4Load(-1000,-800,-600,-400,1000,50); nachladen
while (X4GetState()) ;                warten
X4Start();                            starten
...
```

#### *Abbruch einer Fahrt*

Eine laufende Positionierung kann nur durch den Aufruf der beiden Abbruchfunktionen beendet werden. Die Funktion **X4Brake** beendet die Fahrt mit der definierten Rampe, d.h. die Motoren werden kontrolliert in den Stillstand gebracht. Mit der Funktion **X4Stop** bricht die Fahrt sofort und ohne Rampe ab. Die Motoren können dabei wegen ihrer Massenträgheit noch einige weitere unkontrollierte Schritte ausführen.



---

**Nach einem NOT-Stop sollte die Maschine unbedingt referenzgefahren werden, da eventuell unkontrollierte Schritte von der Software nicht erfaßt wurden.**

---

**X4Step** bietet aber auch noch die Möglichkeit, die Funktion **X4Brake** mit bestimmten Tasten oder Tastencodes zu verknüpfen. Die Positionierung ist somit mit bestimmten Tastatureingaben unterbrechbar ohne daß dies in der übergeordneten Applikation verwaltet werden muß.

#### *Bremsen bei bestimmten Tastencodes*

Die Funktion **X4BrakeCodes** dient zur Definition dieser Tasten. Es sind bis zu 20 Tastencodes möglich. Die PC-Tastatur sendet beim Niederdrücken einer Taste einen sogenannten Makecode und beim Loslassen den entsprechenden Breakcode. Beide Code-Arten sind zur Definition verwendbar. Es ist somit leicht möglich, eine Positionierung beim Niederdrücken einer bestimmten Taste zu beginnen und beim Loslassen wieder zu beenden. In Beispiel 7 ist diese Art der Tastenbehandlung realisiert.

Eine Aufstellung der Tastencodes mit den Tastaturlayouts befindet sich im Referenzteil dieses Handbuchs. Die Scancodes einzelner Tasten kann man auch mit dem beigefügten Freeware-Programm STATUS.EXE ermitteln.

*X4Step verwaltet Tastatureingaben selbständig*

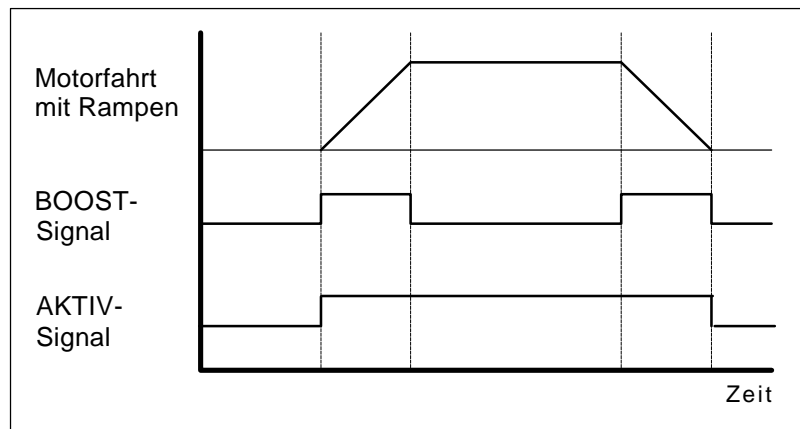
**X4Step** verwaltet die Tastatureingaben während einer laufenden Positionierung selbst. Dies ist zur Gewährleistung einer gleichmäßigen Ansteuerung der Motoren und Einhaltung aller zeitlichen Vorgaben absolut notwendig.

Es bedeutet allerdings auch, daß während laufender Motoren keine Tastatureingaben an die übergeordnete Applikation möglich sind. Wenn die Motoren stehen erfolgen die Tastatureingaben wieder in gewohnter Weise über den geladenen Treiber oder die BIOS-Funktion.

*eigener 1ms Timer*

Zur genauen Zeitmessung kann der PC-Timer mit seiner Auflösung von etwa 55 ms kaum verwendet werden. **X4Step** bietet deshalb einen Timer mit einem 1 ms Raster, der jederzeit ausgelesen werden kann.

Neben den 8 Datenleitungen besitzt die parallele Schnittstelle noch 4 weitere Ausgänge und 5 Eingänge. Zwei dieser Ausgänge verwendet **X4Step** für ein AKTIV- und BOOST-Signal. Das AKTIV-Signal ist immer dann HIGH, wenn einer oder mehrere Motoren laufen. Das BOOST-Signal wird während den Rampen HIGH und kann zur kurzzeitigen Stromerhöhung an den Motorendstufen verwendet werden.



*weitere Ausgänge zur freien Verwendung*

Die beiden anderen Ausgänge können je nach Bedarf von der Applikation verwendet werden. Das Setzen und Rücksetzen erfolgt mit der Funktion **X4SetOut**.

Die fünf Eingänge kann man mit der Funktion **X4GetIn** abfragen und z.B. als START- oder STOP-Signal auswerten.

---

## 3. *X4Step* und eigene Applikationen

---

### 3.1 Installation

*Inhalt der  
Diskette*

Die Diskette mit *X4Step* beinhaltet mehrere Dateien. Die Bibliothek selbst liegt im Objektformat für verschiedene Compiler vor. Der letzte Buchstabe im Dateinamen lautet S oder L und definiert das Speichermodell SMALL oder LARGE.

#### Die Dateien im Einzelnen

<b>X4STEP.H</b>	Header-Datei mit den Funktionsprototypen und Definitionen
<b>X4SBC30x.OBJ</b>	<i>X4Step</i> für Borland C++ 3.0
<b>X4SBC45.x.OBJ</b>	<i>X4Step</i> für Borland C++ 4.5
<b>X4STCP1x.OBJ</b>	<i>X4Step</i> für Turbo C++ 1.0
<b>X4SMSC5x.OBJ</b>	<i>X4Step</i> für Microsoft C 5.1
<b>X4SMSC6x.OBJ</b>	<i>X4Step</i> für Microsoft C 6.0
<b>X4SMSC8x.OBJ</b>	<i>X4Step</i> für Microsoft C 8.0
<b>X4SPC20x.MIX</b>	<i>X4Step</i> für Power C 2.0
<b>STATUS.EXE</b>	Hilfsprogramm zur Ermittlung der Tastaturcodes
<b>BEISP1..10.C</b>	Beispiele dieses Handbuchs als C-Quelle und
<b>BEISP1..10.EXE</b>	ausführbares Programm
<b>TEST.C</b>	Testprogramm mit allen Funktionen als
<b>TEST.EXE</b>	C-Quelle und ausführbar
<b>README</b>	Letzte Hinweise und Änderungen, die im Handbuch nicht mehr berücksichtigt wurden

Die gewünschte Bibliothek von *X4Step* sollte ins Verzeichnis der anderen Compiler-Bibliotheken und die Header-Datei zu den anderen Header-Dateien des Compilers kopiert werden.

Die Header-Datei **X4STEP.H** muß in jedem Modul per **#include** eingebunden werden, in dem Sie Funktionen von *X4Step* verwenden.

*weitere Compiler  
und Compiler-  
Versionen*

Sollte die für Ihre Compiler-Version passende OBJ-Datei fehlen, so versuchen Sie bitte das Modul der nächstniedrigeren Version. Wenn Sie z.B. mit Microsoft C Version 7.0 arbeiten, können die Module X4SMSC6x.OBJ verwendet werden. Die Objektformate sind meist über mehrere Compiler-Versionen kompatibel.

*X4Step* für Compiler anderer Hersteller ist eventuell auf Anfrage erhältlich.

## 3.2 Verwendung mit Turbo C und Borland C

Bei den C-Compilern von Borland gibt es zwei Möglichkeiten der Projektbearbeitung. Die integrierte Arbeitsumgebung IDE verwaltet komplette Projekte, deren einzelne Module in einer Projektdatei definiert werden müssen. In diese Projektdatei muß man auch *X4Step* aufnehmen und zwar die zum Compiler und zum verwendeten Speichermodell gehörende OBJ-Datei.

Bei Verwendung der Kommandozeilenversion müssen Sie lediglich die passende Objektdatei von *X4Step* für den Linkaufruf angeben. Der Aufrufschalter -o definiert die folgende Datei als Objektdatei, die direkt an den Linker weitergereicht wird.

*Beispiel*

<b>bcc -ox4sbc30s testprog.c</b>	für Borland C++
<b>tcc -ox4stcp1s testprog.c</b>	für Turbo C++
<b>tlink textprog x4stcp1s</b>	nur Linkeraufruf

## 3.3 Verwendung mit Microsoft C

Microsoft C Compiler erlauben in der Aufrufzeile neben \*.C auch \*.OBJ Dateien. Bei ihnen findet keine Übersetzung statt aber der anschließende Linklauf bindet sie ins lauffähige Zielprogramm ein.

*Beispiel*

<b>cl testprog.c x4smc6s.obj</b>	Compiler und Linker
<b>link testprog.obj x4smc6s.obj</b>	nur Linkeraufruf

## 3.4 Beispielprogramm

Bestandteil dieses Pakets ist neben allen in diesem Handbuch beschriebenen Beispielen auch ein Testprogramm im C-Quellcode und als ausführbare EXE-Datei, in dem alle Funktionen von *X4Step* anschaulich verwendet werden.

*Testprogramm  
mit allen  
Funktionen*

**Der Aufruf erfolgt einfach mit**

TEST  oder

TEST 2  für LPT2

Sofort nach dem Start zeigt das Programm alle möglichen Kommandos an. Diese Kurzanleitung ist auch jederzeit mit dem Befehl **h** abrufbar.

Testprogramm für Funktionensammlung X4Step

Mögliche Kommandos :

```
-----  
x->    - zeigt selektierte Achse  
Q      - beendet das Testprogramm  
Sx     - selektiert Achse 1 bis 4  
Dx     - lädt Anzahl Schritte für selektierte Achse  
Fx     - lädt Frequenz in Hz für selektierte Achse  
Rx     - lädt Rampenzeit in ms für selektierte Achse  
G      - startet alle geladenen Achsen  
I      - zeigt alle aktuell geladenen Achsen  
M+     - fährt selektierte Achse endlos bis ESC  
M-     - fährt selektierte Achse endlos bis ESC in  
        Gegenrichtung  
C      - ermöglicht Fahren der selektierten Achse mit  
        den Cursor-Tasten  
Ax,y,z - führt Frequenztest für selektierte Achse aus  
        fährt von Frequenz x bis Frequenz y in  
        Abständen z  
T      - fährt selektierte Achse ständig vor und zurück  
E      - liest zyklisch die 5 Eingänge und zeigt deren  
        Pegel an  
H, ?   - zeigt diese Hilfe an  
  
1->
```

Das Testprogramm dient hauptsächlich als Beispiel für die Funktionalität von *X4Step*. Es wurde kein Wert auf Bedienungsfreundlichkeit gelegt. Die Ein- und Ausgaben sind ausschließlich mit Standard-C-Funktionen realisiert.

*mögliche  
Kommandos*

**Das Testprogramm läßt sich mit folgenden Kommandos bedienen :**

**Q** beendet das Programm und kehrt zu MS-DOS zurück

**H, ?** zeigt den Hilfetext an

- Sx** wählt eine Achse 1..4 als aktuelle Achse aus. Nur die aktuelle Achse kann mit Werten geladen werden.
- Dx** lädt die aktuelle Achse mit einem Verfahrensweg. Es sind Wege von -16777216 bis +16777216 Schritte möglich.
- Fx** lädt die aktuelle Achse mit einer Frequenz. Es sind Werte von 10 bis 20000 Hertz möglich.
- Rx** lädt die aktuelle Achse mit einer Rampenzeit. Es sind Werte von 0 bis 1000 ms möglich. Die Zeit gilt sowohl für die Beschleunigungs- als auch die Bremsrampe.
- G** startet die Positionierung aller Achsen mit geladenen Werten. Alle Achsen fahren interpoliert zueinander, d.h. sie fahren gleichzeitig los und sind gleichzeitig fertig.
- I** zeigt die geladenen Werte (Weg, Frequenz und Rampenzeit) aller Achsen zur Übersicht an
- M+** startet die aktuelle Achse zur Endlosfahrt in positive Richtung. Es werden die geladene Rampenzeit und Frequenz verwendet. Die Fahrt ist mit der Taste `[ESC]` abbrechbar.
- M-** dgl. nur negative Richtung
- C** ermöglicht die Fahrt der aktuellen Achse mit den Cursortasten `[LEFT]` und `[RIGHT]`. Es werden die geladene Rampenzeit und Frequenz verwendet. Die Fahrt bricht beim Loslassen der Taste ab.
- A<sub>x,y,z</sub>** fährt die aktuelle Achse über ein Frequenzspektrum. Es wird der aktuelle Weg mit allen Frequenzen von *x* bis *y* im Frequenzabstand *z* verfahren. Damit sind leicht Resonanzstellen der Motoren ermittelbar.
- T** fährt die aktuelle Achse mit den eingestellten Werten ständig vor und zurück. Der Abbruch erfolgt mit der Taste `[ESC]`.
- E** liest ständig die 5 Eingänge ein und zeigt den Pegel an.

*Bedienungs-  
beispiele*

**Achse 1 um 1000 Schritte mit  
Frequenz 2000 Hz und 50 ms  
Rampenzeit verfahren.**

s1  
d1000  
f2000  
r50  
g

**Achsen 1, 2 und 3 interpoliert  
verfahren.**

```
s1  
d1000  
f10000  
r100  
s2  
d500  
s3  
d100  
g
```

**Frequenzspektrum von 200 Hz  
bis 500 Hz mit Abstand  
20 Hz mit Achse 2 abfahren.**

```
s2  
d1000  
r0  
a200,500,20
```

**Abwechselnde Vor-/Rückfahrt  
mit Achse 1 um 2000 Schritte  
mit 5000 Hz und  
100 ms Rampenzeit**

```
s1  
d2000  
f5000  
r100  
t Abbruch 
```

**Endlosfahrt mit Achse 3 bei  
10000 Hz und 500 ms Rampen-  
zeit**

```
s3  
f10000  
r500  
m+ Abbruch 
```

---

## 4. Referenz

---

### 4.1 Mögliche Fehlercodes als Rückgabewerte

<code>#define X4NoErr</code>	0	kein Fehler aufgetreten, Funktion korrekt ausgeführt
<code>#define X4ErrNoSteps</code>	1	Weg muß mindestens aus einem Schritt bestehen
<code>#define X4ErrFreqTooHigh</code>	2	Frequenz ist größer 20 kHz, es sind Werte von 10 Hz bis 20 kHz möglich
<code>#define X4ErrFreqTooLow</code>	3	Frequenz ist kleiner 10 Hz
<code>#define X4ErrRampTooHigh</code>	4	Rampenzeit größer 1000 ms
<code>#define X4ErrRampFreq</code>	5	Rampenzeit gegenüber Frequenz zu groß, die Rampenzeit in ms darf nicht größer sein als die Frequenz in Hz
<code>#define X4ErrLPTNotFound</code>	6	gewählte Schnittstelle bei der Initialisierung nicht vorhanden
<code>#define X4ErrNotActive</code>	7	im Moment läuft keine Achse, die angeforderte Funktion ist deshalb sinnlos
<code>#define X4ErrParamFail</code>	8	ein Parameter ist nicht im gültigen Bereich, z.B. LPT-Schnittstelle 0 oder größer 4
<code>#define X4ErrInitNotDone</code>	9	die Initialisierung wurde bisher noch nicht durchgeführt
<code>#define X4ErrIsActive</code>	10	im Moment laufen eine oder mehrere Achsen, die angeforderte Funktion ist deshalb sinnlos
<code>#define X4ErrNotLoaded</code>	11	ein Start-Befehl kann erst erfolgen, wenn eine oder mehrere Achsen mit Wegen geladen wurden
<code>#define X4ErrIsLoaded</code>	12	die Achsen wurden bereits mit Wegen geladen, ein weiteres Laden ist erst nachdem nächsten Start möglich
<code>#define X4ErrStepsTooHigh</code>	13	es sind max. 16777216 Schritte pro Weg möglich

## 4.2 Beschreibung der einzelnen Funktionen

### X4Init

**Aufruf :** `#include "x4step.h"`  
`int X4Init(int lpt);`

**Beschreibung :** Die Initialisierungsfunktion erledigt alle notwendigen Arbeiten für den nachfolgenden Gebrauch der Positionierfunktionen. Mit ihr muß die Bibliothek *X4Step* gestartet werden. Man muß sie unbedingt vor der Benutzung anderer Funktionen aufrufen.

Am Programmende müssen alle Einstellungen mit der Funktion **X4ReInit** zurückgenommen werden, anderenfalls kann es zu einem Rechnerabsturz kommen.

**Parameter :** `int lpt` parallele Schnittstelle 1 bis 4, über die die Motoren gesteuert werden sollen

**Rückgabewert :** `X4ErrParamFail`  
`X4ErrLPTNotFound`  
`X4NoErr`

**Beispiel :**

```
#include "x4step.h"

main()
{
    X4Init(2);           // X4Step starten und
    X4MovePos(1,1000,50); // LPT2 verwenden
                        // Achse 1 endlos fahren mit
                        // 1000 Hz und 50 ms Rampe
    while (X4GetState())
        ;               // warten, bis mit ESC
                        // abgebrochen wird
    X4ReInit();         // X4Step beenden
}
```

# X4ReInit

**Aufruf :** `#include "x4step.h"`  
`int X4ReInit(void);`

**Beschreibung :** Alle durch **X4Init** erledigten Einstellung nimmt **X4ReInit** wieder zurück. Vor dem Programmende muß unbedingt diese Funktion aufgerufen werden, anderenfalls kann es zu Rechnerabstürzen kommen.

**Parameter :** keine

**Rückgabewert :** X4ErrInitNotDone  
X4NoErr

**Beispiel :**

```
#include "x4step.h"

main()
{
    // X4Step starten und
    X4Init(2);           // LPT2 verwenden
    X4MovePos(1,1000,50);
                        // Achsel endlos fahren mit
                        // 1000 Hz und 50 ms Rampe
    while (X4GetState() // warten, bis mit ESC
           ;           // abgebrochen wird
           X4ReInit(); // X4Step beenden
}
```

# X4Load

**Aufruf :**

```
#include "x4step.h"
int X4Load(
long weg1,
long weg2,
long weg3,
long weg4,
int freq,
int ramp);
```

**Beschreibung :** Mit der Funktion **X4Load** kann man Wege für eine oder mehrere Achsen laden. Die Achse mit dem längsten Weg wird zur Führungsachse und fährt mit der definierten Frequenz und Rampenzeit. Alle anderen Achsen fahren ihren Weg linear interpoliert zur Führungsachse, d.h. sie fahren mit einer niedrigeren oder maximal der gleichen Geschwindigkeit so, daß sie exakt mit der Führungssache wieder zum Stillstand kommen.

Die Fahrt geladener Wege kann mit der Funktion **X4Start** gestartet werden. Sofort nach dem Start der Achsen ist das Nachladen der nächsten Wege möglich. Die Funktionen von **X4Step** können während der Abarbeitung eines Wegs schon die nächsten Wege aufbereiten. Auf diese Weise wird die Totzeit zwischen einzelnen Positionierungen minimiert.

**Achtung :** Die Frequenz in Hz muß gleich oder größer sein als die gewählte Rampenzeit in ms.

**Parameter :**

long weg1..4	zu fahrende Wege in Schritten für jede Achse. Die Wege können -16777216 bis +16777216 Schritte lang sein, wobei mindestens eine Achse mit einem Weg ungleich 0 geladen werden muß
--------------	---

int freq	Geschwindigkeit der Führungsachse in Hz (10 bis 20000)
----------	--

int ramp	Rampenzeit der Führungsachse in ms (0 bis 1000)
----------	---

**Rückgabewert :**

X4ErrIsLoaded	X4ErrNoErr
X4ErrFreqTooHigh	X4ErrFreqTooLow
X4ErrRampTooHigh	X4ErrRampFreq
X4ErrNoSteps	X4ErrStepsTooHigh
X4ErrInitNotDone	

**Beispiel :**

```
#include "x4step.h"

main()
{
    // X4Step starten und
    X4Init(2); // LPT2 verwenden
    X4Load(1000,0,0,0,1000,50);
    // Achse 1 laden
    X4Start(); // Positionierung starten
    X4Load(-1000,0,0,0,1000,50);
    // Achse nachladen
    while (X4GetState())
        // warten, bis Positionierung
        ; // beendet
    X4Start(); // zweiten Weg starten
    while (X4GetState())
        // warten, bis Positionierung
        ; // beendet
    X4ReInit(); // X4Step beenden
}
```

# X4Start

**Aufruf :** `#include "x4step.h"`  
`int X4Start(void);`

**Beschreibung :** Die Funktion **X4Start** startet die Positionierung der Achsen, sofern sie vorher mit Wegen geladen wurden. Sofort nach dem Start ist ein Nachladen der nächsten Wege möglich. Das Ende der Positionierungen kann mit der Funktion **X4GetState** abgefragt werden.

**Parameter :** keine

**Rückgabewert :** X4ErrNotLoaded  
X4ErrIsActive  
X4NoErr

**Beispiel :**

```
#include "x4step.h"

main()
{
    X4Init(2);           // X4Step starten und
                        // LPT2 verwenden
    X4Load(1000,0,0,0,1000,50); // Achse 1 laden
    X4Start();          // Positionierung starten
    while (X4GetState())
        ;               // warten, bis Positionierung
                        // beendet
    X4ReInit();         // X4Step beenden
}
```

# X4Brake

**Aufruf :** `#include "x4step.h"`  
`int X4Brake(void);`

**Beschreibung :** Die Funktion **X4Brake** stoppt eine laufende Positionierung mit der definierten Rampe ab, d.h. die aktuelle Fahrt aller Motoren wird in die Bremsrampe übergeleitet. Die insgesamt verfahrenen Schritte kann man mit der Funktion **X4GetSteps** abfragen.

**Parameter :** keine

**Rückgabewert :** X4ErrNotActive  
X4NoErr

**Beispiel :**

```
#include "x4step.h"

long i;

main()
{
    // X4Step starten und
    X4Init(2);           // LPT2 verwenden
    X4MovePos(1,1000,50);
                        // Achse 1 endlos fahren mit
                        // 1000 Hz und 50 ms Rampe
    for (i=0;i<30000;+i) // warten
        ;
    X4Brake();          // Fahrt abbrechen
    while (X4GetState())
        // warten bis Achse steht
        ;
    i = X4GetSteps(1);
    X4ReInit();        // gefahrene Schritte lesen
                        // X4Step beenden
}
```

# X4Stop

**Aufruf :** `#include "x4step.h"`  
`int X4Stop(void);`

**Beschreibung :** Die Funktion **X4Stop** stoppt die laufende Positionierung ohne Rampe ab. Die Motoren können wegen ihrer Massenträgheit noch einige weitere Schritte machen, die die Software nicht erkennen kann. Diese Funktion sollte nur im Notfall oder als NOT-Stop verwendet werden.

**Parameter :** keine

**Rückgabewert :** X4NoErr  
X4ErrNotActive

**Beispiel :**

```
#include "x4step.h"

main()
{
    // X4Step starten und
    X4Init(2); // LPT2 verwenden
    X4MovePos(1,1000,50);
    // Achse 1 endlos fahren mit
    // 1000 Hz und 50 ms Rampe
    for (i=0;i<30000;++i) // warten
        ;
    X4Stop(); // Fahrt abbrechen
    X4ReInit(); // X4Step beenden
}
```

# X4GetState

**Aufruf :** `#include "x4step.h"`  
`int X4GetState(void);`

**Beschreibung :** Die Funktion **X4GetState** liefert den Achszustand zurück. Mit ihr kann abgefragt werden, ob gerade eine oder mehrere Achsen laufen oder alle Achsen stehen. Eigentlich zeigt **X4GetState** nur den Zustand des AKTIV-Signals an.

**Parameter :** keine

**Rückgabewert :** 0 alle Achsen stehen  
>0 eine oder mehrere Achsen laufen

**Beispiel :**

```
#include "x4step.h"

main()
{
    // X4Step starten und
    X4Init(2);           // LPT2 verwenden
    X4MovePos(1,1000,50);
                        // Achse 1 endlos fahren mit
                        // 1000 Hz und 50 ms Rampe
    while (X4GetState())
        // warten, bis mit ESC
        ;               // abgebrochen wird
    X4ReInit();         // X4Step beenden
}
```

# X4GetSteps

**Aufruf :** `#include "x4step.h"`  
`long X4GetSteps(int achse);`

**Beschreibung :** **X4GetSteps** liefert die aktuell zurückgelegten Schritte seit dem letzten Start für eine bestimmten Achse. Der interne Schrittzähler jeder Achse kann bis zum nächsten Start ausgelesen werden, also auch im Stillstand nach einer Positionierung. Mit jedem Startbefehl wird er automatisch gelöscht.

**Parameter :** `int achse` Achse 1 bis 4, deren Schrittzähler gelesen werden soll

**Rückgabewert :** `long` seit dem letzten Start zurückgelegte Schritte

**Beispiel :**

```
#include <stdio.h>
#include "x4step.h"

main()
{
    X4Init(2);           // X4Step starten und
                        // LPT2 verwenden
    X4MovePos(1,1000,50);
                        // Achse 1 endlos fahren mit
                        // 1000 Hz und 50 ms Rampe
    while (X4GetState()) // warten, bis Abbruch
        printf("\n%d",X4GetSteps(1));
                        // Schritte lesen und ausgeben
    X4ReInit();         // X4Step beenden
}
```

# X4MovePos

# X4MoveNeg

**Aufruf :**

```
#include "x4step.h"
int X4MovePos(
int achse,
int freq,
int ramp);

int X4MoveNeg(
int achse,
int freq,
int ramp);
```

**Beschreibung :** Die Funktionen **X4MovePos** und **X4MoveNeg** fahren eine Achse endlos mit gewählter Frequenz und Rampenzeit. Die Positionierung kann nur durch die Funktionen **X4Brake** oder **X4Stop** oder durch einen definierten Tastendruck unterbrochen werden. Die aktuell zurückgelegten Schritte sind mit der Funktion **X4GetSteps** lesbar.

**Achtung :** Die Frequenz in Hz muß gleich oder größer sein als die gewählte Rampenzeit in ms.

**Parameter :**

int achse	Achse, die endlos verfahren wird
int freq	Schrittfrequenz, mit der verfahren wird
int ramp	Rampenzeit bis zur Sollfrequenz

**Rückgabewert :**

X4ErrIsLoaded	X4ErrNoErr
X4ErrFreqTooHigh	X4ErrFreqTooLow
X4ErrRampTooHigh	X4ErrRampFreq
X4ErrInitNotDone	X4ErrIsActive

**Beispiel :**

```
#include "x4step.h"

main()
{
    // X4Step starten und
    X4Init(2); // LPT2 verwenden
    X4MovePos(1,1000,50);
    // Achse 1 endlos fahren mit
    // 1000 Hz und 50 ms Rampe
    while (X4GetState()) // warten, bis mit ESC
        ; // abgebrochen wird
    X4ReInit(); // X4Step beenden
}
```

# X4BrakeCodes

**Aufruf :** `#include "x4step.h"`  
`int X4BrakeCodes(char *codes);`

**Beschreibung :** Mit der Funktion **X4BrakeCodes** können alle Tastencodes definiert werden, mit denen eine laufende Positionierung unterbrochen wird. **X4Step** verwaltet die Tastatur selbständig und kann somit alle Tastencodes (Make- und Breakcodes) erkennen. Leider bedeutet dies auch, daß normale Tastatureingaben während laufender Positionierung nicht möglich sind.

Die Tastatur sendet beim Niederdrücken einer Taste einen bestimmten Makecode und beim Loslassen den zur Taste gehörenden Breakcode. Der Breakcode einer Taste entspricht dem Makecode +128.

Neben den definierten Tastencodes führt immer die Taste ESC (Makecode 1) zum Abbruch einer Positionierung. Sie muß nicht definiert werden, sondern ist aus Sicherheitsgründen standardmäßig aktiv.

Die Makecodes der Tasten sind auf folgenden Schemen dargestellt und können auch mit dem beiliegendem Programm STATUS.EXE ermittelt werden.

59	60	41	2	3	4	5	6	7	8	9	10	11	12	13	43	14	1	69	70	84
61	62	15	16	17	18	19	20	21	22	23	24	25	26	27	71	72	73	55		
63	64	29	30	31	32	33	34	35	36	37	38	39	40	28	75	76	77	74		
65	66	42	44	45	46	47	48	49	50	51	52	53	54	79	80	81				
67	68	56	57										58	82	83	78				

Layout mit Make-Codes einer AT-Tastatur (Breakcodes +128)

1	59	60	61	62	63	64	65	66	67	68	87	88	70							
41	2	3	4	5	6	7	8	9	10	11	12	13	14	82	71	73	69	53	55	74
15	16	17	18	19	20	21	22	23	24	25	26	27	28	83	79	81	71	72	73	78
29	30	31	32	33	34	35	36	37	38	39	40	43				75	76	77		
42	86	44	45	46	47	48	49	50	51	52	53	54	72			79	80	81	28	
29		56	57								56		29	75	80	77	82	83		

Layout mit Make-Codes einer MF2-Tastatur (Breakcodes +128)

**Parameter :** char \*codes Feld mit Codes der Tasten, die zum Abbruch der Positionierung führen, Abschluß des Felds mit \0.

**Rückgabewert :** X4NoErr

**Beispiel :**

```
#include "x4step.h"

char codes[2];

main()
{
    X4Init(2);           // X4Step starten und
                        // LPT2 verwenden
    codes[0] = 28;      // ENTER-Taste soll abbrechen
    codes[1] = 0;      // Feld mit \0 abschließen
    X4BrakeCodes(codes);
                        // Abbruchtasten definieren
    X4MovePos(1,1000,50);
                        // Achse 1 endlos fahren mit
                        // 1000 Hz und 50 ms Rampe
    while (X4GetState())
        ;               // warten, bis mit ESC oder
                        // ENTER abgebrochen wird
    X4ReInit();         // X4Step beenden
}
```

```
#include <dos.h>
#include "x4step-h"

main()
{
    char codes[2], c;

    X4Init(2);           // X4Step starten und
                        // LPT2 verwenden
    codes[1] = 0;       // Feld mit \0 abschließen
    switch (getch())
        // auf + oder - Taste warten
    {
        case '+' :
            c = inp(0x60); // Makecode holen
            codes[0] = c+128; // Breakcode rechnen
            X4BrakeCodes(codes);
                        // und als Abbruch-
                        // taste definieren
            X4MovePos(1,1000,50);
                        // endlos fahren bis
            while (X4GetState())
                ;       // Taste losgelassen
            break;
        case '-' :
            // dgl. für negative
            c = inp(0x60); // Richtung
            codes[0] = c+128;
            X4BrakeCodes(codes);
            X4MoveNeg(1,1000,50);
            while (X4GetState())
                ;
            break;
        default : break;
    }
    X4ReInit();         // X4Step beenden
}
```

# X4LastBrakeCode

**Aufruf :** `#include "x4step.h"`  
`int X4LastBrakeCode(void);`

**Beschreibung :** Die Funktion **X4LastBrakeCode** liefert nach einem Abbruch den erkannten Tastencode zurück. Wenn mehrere Tastencodes definiert sind kann somit ermittelt werden, welche Taste zum Abbruch der Positionierung führte.

**Parameter :** keine

**Rückgabewert :** `int` Tastencode, der zum Abbruch der Positionierung führte, oder 0 wenn noch nicht abgebrochen oder kein Tastencode erkannt

**Beispiel :**

```
#include <stdio.h>
#include "x4step.h"

char codes[2];

main()
{
    X4Init(2);           // X4Step starten und
    codes[0] = 28;      // LPT2 verwenden
    codes[1] = 0;       // ENTER-Taste soll abbrechen
    X4BrakeCodes(codes); // Feld mit \0 abschließen
                       // Abbruchtasten definieren
    X4MovePos(1,1000,50); // Achse 1 endlos fahren mit
                       // 1000 Hz und 50 ms Rampe
    while (X4GetState()) // warten, bis mit ESC oder
    ;                       // ENTER abgebrochen wird
    switch (X4LastBrakeCode()) // Tastencode auswerten
    {
        case 1 :
            printf("ESC hat abgebrochen");
            break;
        case 28 :
            printf("ENTER hat abgebrochen");
            break;
        default :
            break;
    }
    X4ReInit();         // X4Step beenden
}
```

# X4ClearTimer

# X4GetTimer

**Aufruf :**

```
#include "x4step.h"
void X4ClearTimer(void);

long X4GetTimer(void);
```

**Beschreibung :** Die Funktionen **X4ClearTimer** und **X4GetTimer** steuern den 1 ms Timer, den **X4Step** zur Verfügung stellt. Bei der Initialisierung und bei jedem Aufruf von **X4ClearTimer** wird der Timer gelöscht. Die aktuell verstrichene Zeit in ms seit der letzten Löschung kann man mit der Funktion **X4GetTimer** auslesen.

**Parameter :** keine

**Rückgabewert :** long verstrichene Zeit in ms

**Beispiel :**

```
#include "x4step.h"

main()
{
    X4Init(2);           // X4Step starten und
                        // LPT2 verwenden
    X4MovePos(1,1000,50);
                        // Achse 1 endlos fahren mit
                        // 1000 Hz und 50 ms Rampe
    X4ClearTimer();     // Timer löschen
    while (X4GetTimer() < 500)
        ;               // 500 ms warten
    X4Brake();          // Fahrt abbrechen
    X4ReInit();         // X4Step beenden
}
```

# X4GetIn

**Aufruf :** `#include "x4step.h"`  
`int X4GetIn(void);`

**Beschreibung :** Die Funktion **X4GetIn** liefert die aktuellen Signalpegel der 5 Eingänge zurück. Die parallele Schnittstelle hat neben den 12 Ausgängen noch 5 TTL-Eingänge, die für externe Signale wie START, STOP oder Endschalter verwendbar sind. Eingänge sind die Pins 10, 11, 12, 13 und 15 an der 25-poligen SUB-D-Buchse.

**Parameter :** keine

**Rückgabewert :** `int` Pegel der 5 Eingänge binär codiert, 1 heißt Pegel HIGH (+5V), 0 heißt Pegel LOW (GND)

<b>Bit</b>	<i>15..5</i>	<i>4</i>	<i>3</i>	<i>2</i>	<i>1</i>	<i>0</i>
<b>Pin</b>	<i>x</i>	<i>11</i>	<i>10</i>	<i>12</i>	<i>13</i>	<i>15</i>

**Beispiel :**

```
#include "x4step.h"

main()
{
    X4Init(2);           // X4Step starten und
    X4MovePos(1,1000,50); // LPT2 verwenden
                        // Achse 1 endlos fahren mit
                        // 1000 Hz und 50 ms Rampe
    while ((X4GetIn() & 0x01) == 0)
        ;               // warten auf Pin 15 HIGH
    X4Brake();          // dann stoppen
    X4ReInit();         // X4Step beenden
}
```

# X4SetOut

**Aufruf :** `#include "x4step.h"`  
`int X4SetOut(int sig);`

**Beschreibung :** Mit der Funktion **X4SetOut** kann man die beiden Zusatzsignale setzen oder rücksetzen. Jede parallele Schnittstelle hat neben den 8 Datenausgängen und 5 Eingängen noch 4 weitere TTL-Ausgänge, die für beliebige Signale verwendet werden können. Es handelt sich hierbei um die Pins 1, 14, 16 und 17 der 25-poligen SUB-D-Buchse.

**X4Step** benutzt Pin 16 als BOOST- und Pin 17 als AKTIV-Signal. Das BOOST-Signal wird immer während der Rampenfahrt HIGH und kann bei Schrittmotor-Endstufen zur kurzzeitigen Stromerhöhung benutzt werden. Das AKTIV-Signal ist immer dann HIGH, wenn einer oder mehrere Motoren laufen.

Auf den beiden anderen Pins 1 und 14 liegen die Zusatzsignale 1 und 2, die mit der Funktion **X4SetOut** geschaltet werden. Zum An- und Ausschalten kann man die vordefinierten Konstanten SIGNAL1ON, SIGNAL1OFF, SIGNAL2ON und SIGNAL2OFF verwenden.

**Parameter :** `int sig` Werte 1 bis 4 oder vordefinierte Konstanten

```
#define SIGNAL1ON      1
#define SIGNAL1OFF    2
#define SIGNAL2ON     3
#define SIGNAL2OFF    4
```

**Rückgabewert :** X4NoErr

**Beispiel :**

```
#include "x4step.h"

main()
{
    X4Init(2);           // X4Step starten und
                        // LPT2 verwenden
    X4SetOut(SIGNAL1ON); // Pin 1 HIGH schalten
    X4MovePos(1,1000,50);
                        // Achse 1 endlos fahren mit
                        // 1000 Hz und 50 ms Rampe
    while (X4GetState()) // warten, bis mit ESC
        ;               // abgebrochen wird
    X4SetOut(SIGNAL1OFF);
                        // Pin 1 wieder LOW
    X4ReInit();        // X4Step beenden
}
```

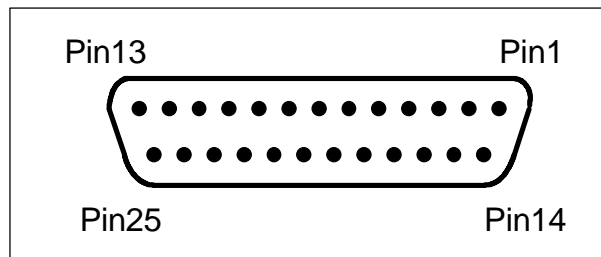
---

## 5. Anhang

---

### 5.1 Steckerbelegung

*X4Step* steuert die Schrittmotor-Endstufen über eine der parallelen Druckerschnittstellen an. Es handelt sich hierbei um 25-polige SUB-D-Buchsen, deren Belegung wie folgt aussieht.



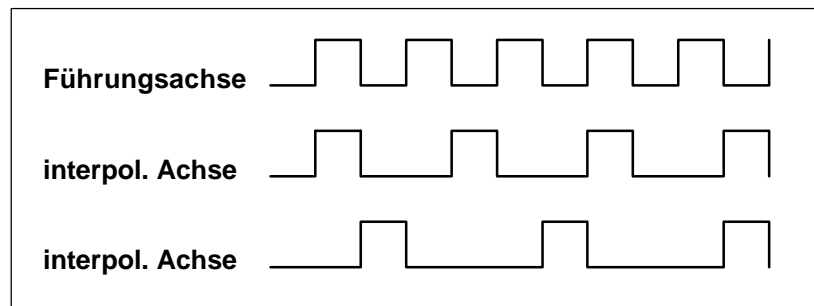
<b>Pin 2</b>	Richtung Motor 1
<b>Pin 3</b>	Takt Motor 1
<b>Pin 4</b>	Richtung Motor 2
<b>Pin 5</b>	Takt Motor 2
<b>Pin 6</b>	Richtung Motor 3
<b>Pin 7</b>	Takt Motor 3
<b>Pin 8</b>	Richtung Motor 4
<b>Pin 9</b>	Takt Motor 4
<b>Pin 1</b>	Signal 1, frei belegbar
<b>Pin 14</b>	Signal 2, frei belegbar
<b>Pin 16</b>	BOOST-Signal während Rampenfahrt
<b>Pin 17</b>	AKTIV-Signal wenn Motoren laufen
<b>Pin 10/11/ 12/13/15</b>	Eingänge, frei nutzbar im Anwendungsprogramm z.B. START, STOP oder Endschalter
<b>Pin 18-25</b>	Signalmasse (0V GND)

Mit den Signalen der parallelen Druckerschnittstelle können alle Schrittmotor-Endstufen angesteuert werden, die mit +5V TTL-Pegel an den Eingängen arbeiten. Idealerweise sollten die Eingangs- und Ausgangssignale galvanisch entkoppelt werden.

## 5.2 Signal-Timing

Die Taktsignale der Führungssachse sind symmetrisch, d.h. LOW- und HIGH-Pegel liegen immer exakt gleichlang an. Die Taktimpulse der interpolierten Achsen sind immer so lang wie die der Führungssachse. Alle Taktimpulse werden niemals kürzer als 25  $\mu$ s.

Die Richtungssignale aller Achsen werden 100  $\mu$ s vor dem ersten Schritt ausgegeben und bleiben bis zum Ende der Positionierung erhalten.



## 5.3 Einschränkungen und Grenzen von X4Step

Trotz der umfangreichen Funktionalität und leichten Verwendbarkeit besitzt **X4Step** auch Grenzen, auf die hier hingewiesen wird.

*läuft nicht unter MS-Windows*

Durch die exklusive Verwendung des PC-Timers und selbständige Verwaltung der Tastatureingaben können Applikationen mit **X4Step** nicht oder nur eingeschränkt unter MS-Windows betrieben werden.

Applikationen zur Ansteuerung von Motoren sind Echtzeitanwendungen für die MS-Windows nicht die notwendigen Eigenschaften zur Verfügung stellt.

*eigene Tastaturverwaltung*

Die selbständige Verwaltung der Tastatureingaben während laufender Positionierungen verhindert Eingaben an die übergeordnete Applikation.

Es kann lediglich mit bestimmten definierten Tastencodes die Fahrt der Motoren unterbrochen werden. Bei stehenden Achsen erfolgt die Tastatureingabe in gewohnter Weise über den installierten Treiber oder die BIOS-Funktion und ist somit auch wieder für die Applikation verfügbar.

*kein Schnittstellen-Zugriff von anderen Programmteilen*

Da **X4Step** die parallele Schnittstelle eigenständig verwaltet, darf von anderen Programmteilen kein Zugriff auf diese erfolgen, weder mit den entsprechenden BIOS-Funktionen noch mit direkten Portzugriffen.

Zur gleichmäßigen und zeitgenauen Ansteuerung der Motoren wird von **X4Step** der PC interne Timerbaustein programmiert und außerdem die Interrupts der Uhr und der Tastatur verwendet. Diese Interrupts stehen der übergeordneten Applikation oder sonstigen aktiven Programmen nicht mehr zur Verfügung. Speziell bei der Benutzung von Debuggern zur Programmentwicklung kann es deswegen zu Konflikten kommen.

*Verhältnis Frequenz zu Rampenzeit*

Bei allen Positionierung muß die Frequenz in Hz gleich oder größer sein als die gewählte Rampenzeit in ms.

**Beispiel :**

1000 Hz, 100 ms	ok
100 Hz, 90 ms	ok
10 Hz, 50 ms	Fehler

*läuft auf Rechnern ab 286*

**X4Step** läuft auf allen 100% IBM kompatiblen Rechnern unter MS-DOS ab Version 5.0. Es sind alle Prozessortypen ab 80286 verwendbar. Eventuell kann bei langsamen Rechnern die maximale Frequenz von 20 kHz nicht erreicht werden.