HaSoTec
Schrittmotorsteuerung
SM-40 PCI
SM-41 PCI

Programmierung
und
Informationen

Version 3.07 D

Inhalt

1 Generelle Hinweise zur Programmierung
1.1 High Level Programmierung unter Windows XP/ 2000/ NT 7-3
1.2 High Level Programmierung unter Windows XP/ 2000/ NT 7-4
1.3 Low Level Programmierung unter Windows XP/ 2000/ NT 7-4
1.4. Low Level Programmierung unter Windows Me/ 98/ 95/ 3.x 7-6
1.5. Low Level Programmierung unter Dos 7-6

2 32- Bit- Programmierung auf prozeduraler Ebene
2.1. Programmierung unter Microsoft Visual C++ .NET, 6.0, 5.0, 4.2, 4.1, 4.0 und 2.0 ohne OCX Control 7-8
2.2. Programmierung unter Microsoft Visual Basic .NET, 6.0, 5.0 und 4.x 7-10

3 Low level Programmierung 7-11
3.1 Aufbau eines Funktionsaufrufs 7-11
3.1.1 Tabellarische Übersicht 7-12
3.1.2 Beschreibung der Treiberfunktionen 7-12
3.2 Hinweise zum Aufrufen von Treiberfunktionen 7-18
3.2.1 Microsoft C++ 1.0-1.52 7-18
3.2.2 Microsoft C/C++ 7.0 7-18
3.2.3 Microsoft C PDS 7-19
3.2.4 Microsoft Quick C 2.5 7-19
3.2.5 Microsoft Quick C für Windows 7-19
3.2.6 Borland C++ 3.1, 4.0, 4.5 7-20
3.3 Microsoft Quick Basic 7-20
3.4 Microsoft Visual Basic 7-21
3.5 Microsoft Macro-Assembler 6.0 7-21
3.6 Microsoft Macro-Assembler 5.1 7-21
3.7 Borland Turboassembler 7-21
3.8 Turbo Pascal für DOS 7-21
3.9 Turbo Pascal für Windows 7-21
4 HaSoTec - Lizenzvertrag 7-22

5 Erweiterte Rechte 7-24

1. Generelle Hinweise zur Programmierung

Dieses Kapitel beschreibt die Programmierung auf prozeduraler Ebene (High- Level- Programmierung) und die Programmierung durch direkte Gerätetreiberaufrufe (Low-Level Programmierung).

Dieses Kapitel beschreibt im Abschnitt 5 die vollständige Low-Level- Schnittstelle der Schrittmotor API 9909. Mit dieser Schnittstelle ist die Schrittmotorkarte vollständig systemübergreifend programmierbar. API steht für Application Programmers Interface und soll als Schnittstelle zur Schrittmotorsteuerung SM-4x dienen. Die Bezeichnung SM-4x gilt für alle Schrittmotorkarten, z.B. SM-40, SM-41. Ist die Funktionalität bei bestimmten Karten nicht vorhanden, so wird darauf gesondert hingewiesen.
Die als fertige Applikationen mitgelieferten Programme und sämtliche Bibliotheken, DLLs oder OCX-Controls benutzen diese Schnittstelle. Unter Dos, Windows 3.0, Windows 3.1, Windows 3.11, Windows 95, Windows 98 / Me erfolgen die Low Level Aufrufe über das Interrupt 60H mit der Übergabe von Parametern in den Registern ax, bx, cx und dx. Unter Windows NT 3.51, Windows NT 4.0, Windows NT 5.0, Windows 2000 und Windows XP erfolgt der Aufruf durch einen Device Driver Einsprung mit den genau gleichen Parametern, nur dass ax, bx, cx, und dx in diesem Fall Namen für Variablen eines Funktionsaufrufs sind.

Die beschriebenen Quellcodebeispiele sind komplette Applikationen die sich leicht erweitern lassen. Wenn Sie einen der angegebenen Compiler benutzen, dann steht sofort eine Applikation bereit, mit der ohne eine Zeile Programmtext zu schreiben, die ersten Bewegungen ausführbar sind. Alle Beispiele enthalten auch die *.EXE Datei, so dass man sich die Beispiele schon vor der Installation eines Compilers anschauen kann.

1.1 High Level Programmierung unter Windows XP/ 2000/ NT

Unter Windows XP, 2000 und NT gibt es OCX Controls und DLLs, mit denen die Bewegungsabläufe des SM-40 direkt ansprechbar sind. Die Kommunikation zum SM-40 erfolgt über den Treiber SM40DRV.SYS.
DLLs, Bibliotheken und Objektdateien gibt es jeweils für Windows XP/ 2000/ NT und Windows Me/ 98/ 95. Die für Windows Me/ 98/ 95 ausgelegten Komponenten sind unter Windows XP/ 2000/ NT nicht verwendbar und führen beim ersten Zugriff zu einer entsprechenden Fehlermeldung.
Unter Windows XP/ 2000/ NT werden nur 32-Bit-Programme unterstützt.

1.2 High Level Programmierung unter Windows Me/ 98/ 95/ 3.x

Unter Windows Me/ 98/ 95/ und 3.x gibt es OCX Controls und DLLs, mit denen die Bewegungsabläufe des SM-40 kompakt ansprechbar sind. Die Kommunikation zum SM-40 erfolgt über den Treiber SM40DRV.EXE, der über die Datei Autoexec.bat beim Systemstart aufgerufen wird.

1.3 Low Level Programmierung unter Windows XP/ 2000/ NT

Unter Windows XP, Windows 2000 und Windows NT erfolgt der Aufruf durch einen Device Driver Einsprung mit gleichen Parametern wie bei einem Interrupt 60h- Aufruf, nur dass bx, cx, und dx in diesem Fall Namen für Variablen eines Funktionsaufrufs sind. Ein Low Level Aufruf erfolgt durch die Funktion:

```
IoctlResult = DeviceIoControl ( hdev,                  // Handle to device
                (ULONG)FKT020,          // IO Control code LowLevel
                &freg,                                 // Buffer to driver.
                sizeof (FREG),,         // Length of buffer in bytes.
                &freg,                                 // Buffer from driver.
                sizeof (FREG),          // Length of buffer in bytes.
                &ReturnedLength,        // Bytes placed in DataBuffer
                NULL
                  );
```

hdev lässt sich durch die Funktion

```
hdev = CreateFile     ("\\\\.\\Sm40Dev",
                      GENERIC_READ,FILE_SHARE_READ, NULL,
                                   OPEN_EXISTING, 0, NULL);
```

erhalten. Diese Handle wird beim Schließen des Programms mit Close (hdev) zurückzugeben.

FREG ist eine Datenstruktur, die folgenden Aufbau hat:

```
typedef struct
        {
        USHORT fnr;     //bx
        USHORT cx;
        USHORT dx;
        PUCHAR reserved;
        ULONG reserved2;
        } FREG;
typedef FREG * PFREG;
```

Die Variablen fnr (bx), cx und dx werden in Zusammenhang mit der Erläuterung der verfügbaren Funktionen im Abschnitt 3 behandelt.

Ab Softwareversion 1.5 ist ein C++ Quellcodebeispiel installierbar.

1.4 Low Level Programmierung unter Windows Me/ 98/ 95/ 3.x/ Dos

1.5 Low Level Programmierung unter Dos

Unter Dos, Windows 3.0, Windows 3.1, Windows 3.11, Windows 95, Windows 98 und Windows Me erfolgen die Low Level Aufrufe über das Interrupt 60H mit der Übergabe von Parametern in den Registern ax, bx, cx und dx.
Im Abschnitt 3 wird neben der detaillierten Beschreibung der Einzelfunktionen auch der compilerspezifische Aufruf für Interrupt 60H behandelt.
Der Treiber SM40DRV.EXE belegt nach dem Aufruf das Softwareinterrupt 60H.
Die Grundstruktur für einen Funktionsaufruf sieht in einem 80x86 Assembler folgendermaßen aus:

```
mov     ax, 9909h
mov     bx, funktion
mov     cx,parameter1
mov     dx,parameter2
int     60h
```

Dieses Programmfragment lässt sich in vielen Hochsprachen auch durch andere Kommandos schreiben. Zum globalen Verständnis sollen zunächst dennoch die Assemblerkommandos erklärt werden.
Jeder Prozessor eines Industriestandard PCs verfügt neben anderen Registern über die 4 Grundregister AX, BX, CX und DX. Ein solches Register kann mit 16-bit-Zahlen operieren. Mit dem Kommando:

```
mov     ax,9909h
```

wird das Register AX mit der hexadezimalen Zahl 9909 geladen.

Durch die Zahl 9909h wird SM40DRV angesprochen. Wenn andere Treiber mit den Kennungen durch das Register AX in gleicher Weise umgehen, dann können weitere Treiber gleichzeitig auf das Interrupt 60h installiert werden.
Durch das Register bx wird die gewünschte Funktion gewählt.
Einer Funktion können bis zu zwei Parameter übergeben werden. Diese werden vor dem Interruptaufruf in die Register CX und DX geladen. Wenn eine Funktion keine Parameter erfordert, dann werden die Werte der Register ignoriert und können nach dem Aufruf verändert worden sein. Das Kommando

```
int     60h
```

bewirkt schließlich den Aufruf der gewünschten Funktion.
In Abhängigkeit von der Funktion werden bis zu 2 Parameter in den Registern CX und DX zurückgegeben.

Ab Softwareversion 1.4 ist ein C++ Quellcodebeispiel installierbar.

32- Bit- Programmierung auf prozeduraler Ebene mit MS-Windows 9x/ Me und MS-Windows XP/ 2000/ NT

Als prozedurale Ebene wird die Nutzung von Bibliotheksfunktionen der zur Karte gelieferten Bibliotheken bezeichnet. Die Nutzung dieser Bibliotheksfunktionen erfordert keine Kenntnisse der Treiberfunktionen von SM40DRV.EXE oder SM40DRV.SYS. Die in den Bibliotheken enthaltenen Funktionen führen in der Regel häufig benötigte komplexe Abläufe aus. Die Bibliotheksfunktionen benutzen sowohl Funktionen des Windows-API als auch des Treibers SM40DRV. Die Nutzung von Bibliothekfunktionen schließt die Anwendung von low level Funktionen an anderer Stelle im Anwenderprogramm nicht aus.
Einige Beispiele enthalten die resource srcript Dateien für die Bibliothek. Die Anordnung der Dialogboxelemente und das Entfernen nicht gebrauchter Elemente ist damit auf einfache Weise möglich.

2.1 Programmierung unter Microsoft Visual C++ 6.0, 5.0, 4.2, 4.1, 4.0 und 2.x ohne OCX Control

Dieses Beispiel ist in der Schrittmotorsoftware ab Version 2.0 enthalten.

Die Bibliothek SM40.LIB entspricht der Installierten Betriebssystemversion und muss bei Wechsel zwischen Windows Me, 98, 95, 3.x und Windows XP/ 2000/ NT ausgetauscht werden.
Die DLL enthält folgende Funktionen:

int FAR PASCAL SM40DLG (*hwnd, msg, wpar, lpar*)

HWND *hwnd*	-	Handle auf das Applikationsfenster
unsigned *msg*	-	Message Parameter der Dialogboxprozedur
WORD *wpar*	-	Word parameter der Dialogboxprozedur
LONG *lpar*	-	Long parameter der Dialogboxprozedur
Rückgabewert	-	0=erfolgreich

MOTORDLG ist eine Dialogboxfunktion für sämtliche Einstellungen aller 4 Motoren.

int FAR PASCAL SM40MOVE (hwnd, *xsteps, ysteps, zsteps, wsteps*)

HWND *hwnd*	Handle auf das Applikationsfenster
LONG xsteps	Schrittanzahl für x,
LONG ysteps	y,
LONG zsteps	z und
LONG wsteps	w- Motor mit Vorzeichen

Rückgabewert - enthält Fehlercode oder 0

SM40MOVE übergibt den Bewegungsvektor und kehrt noch während die Bewegung ausgeführt wird in das Programm zurück.

int FAR PASCAL SM40STATUS (void)

Der Rückgabewert dieser Funktion ist 0, wenn SM-40 für das nächste Kommando bereit ist.

int FAR PASCAL SM40STOP (void)

Diese Funktion bricht alle Bewegungen des SM-40 ab und bewirkt eine vollständige Neuinitialisierung. Der Rückgabewert ist 0, wenn das Rücksetzen erfolgreich war.

3. Low level Programmierung

Als low level Programmierung werden im Zusammenhang mit dem SM-40 solche Programmabschnitte bezeichnet, die direkte Aufrufe des Treibers SM40DRV benutzen.

3.1. Aufbau eines Funktionsaufrufs

Ein Low Level Funktionsaufruf übergibt den API-Code 9909h, die Funktionsnummer (bx), den ersten Parameter cx und den zweiten Parameter dx. Nach dem Aufruf wird ein erster Parameter cx und ein zweiter Parameter dx zurückgegeben. Entsprechende Pointer können für Eingabe- und Ausgabeparameter auf die selben Variablen cx und dx zeigen. Die Ausgabeparameter überschreiben dann einfach die Eingabeparameter, die man vor dem nächsten Aufruf dann neu setzen muss. Bei WinMe/98/95/3.x oder Dos und direktem Aufruf von SM40DRV.EXE werden die Prozessorregister cx und dx ebenfalls überschrieben. SM40DRV kann mit bis zu zwei Karten des Typs SM-40 arbeiten. Die Funktionen 0 bis 4 können übersprungen werden. Die Ausschrift von SM40DRV.EXE zeigt die Basisadressen von bis zu zwei Karten an und ein großes " H" zeigt an, dass die entsprechende Karte erfolgreich mit Funktion 4 initialisiert wurde, während ein kleines " h" eine Fehlermeldung anzeigt.

5.1.1.1 Tabellarische Übersicht

In der Spalte Umgebung wird die Verwendbarkeit für verschiedene Entwicklungsumgebungen gezeigt:

W - Windows 3.x/ 9x/ Me
N - Windows XP/ 2000 und NT
D - DOS

ax=9909h		Eingabe		Ausgabe		SM-41
bx Funktionsname	Umg.	cx	dx	cx	dx	erweitert
00 SmInit	WND	-	-	9909	vers	
01 SmCards	WND	-	-	count	inits	
02 SmSwitchCard	WND	card	-	-	-	
03 SmResetCard	WND	-	-	errcode	-	ja
04 SmInitCard	WND	-	-	errcode	-	
05 SmAccel	WND	accel	motor			
06 SmDecay	WND	decay	motor			o.F.
07 SmSpeed	WND	speed	motor	-	-	
08 SmSteps	WND	steps	motor	-	-	
09 SmSteps32	WND	steps32	motor	-	-	
10 SmMoveAndWait	WND	-	-	status	-	
11 SmMove	WND	-	-	errcode	-	
12 SmStatus	WND	-	-	status	-	
13 SmHoldPower	WND	power	-	-	-	ja
14 SmStepTab	WND	typ	-	-	-	ja
15 SmGetAccel	WND	-	motor	accel	-	
16 SmGetDecay	WND	-	motor	decay	-	
17 SmGetSpeed	WND	-	motor	speed	-	
18 SmGetSteps	WND	-	motor	steps	-	
19 SmGetSteps32	WND	-	motor	steps32	-	
20 SmGoMin	WND	-	motor	status	-	
21 SmGoMax	WND	-	motor	status	-	
22 SmSwitches	WND	-	-	status	-	
23 SmReleaseMin	WND	-	motor	status	-	
24 SmReleaseMax	WND	-	motor	status	-	
25 SmSwitchesDirect	WND	-	-	status	-	
26 SmPosition	WND	-	motor	posL	posH	
27 SmSetAccLength	WND	acclength	motor	-	-	
28 SmGetAccLength	WND	-	motor	acclen	-	

5.1.2. Beschreibung der Treiberfunktionen

Funktion 0
Funktionsname: SmInit

Eingabe:	ax	Treiberkennung (MASM: 9909H Basic &H9909 Pascal: $9909 C: 0x9909)
Rückgabe:	dx	Treiberversion * 100h

Diese Funktion sollte jedes Programm sofort nach Programmstart rufen.

Funktion 1
Funktionsname: SmCards

Eingabe:		keine
Rückgabe:	cx	Anzahl der SM Karten
	dx	Anzahl der initialisierten Karten

Diese Funktion kann verwendet werden, um festzustellen, ob Karten im System vorhanden sind. Ist cx ungleich dx, dann wird angezeigt, dass Karten vor der Benutzung initialisiert werden müssen (Funktionen 2,3,4). Werden unter Windows manuelle Einstellungen bei der Vergabe von Resourcen verwendet, dann ist diese Funktion zum Programmstart aufzurufen, damit nicht mit den ursprünglich vom plug & play Bios vergebenen Resource weitergearbeitet wird.

Funktion 2
Funktionsname: SmSwitchCard

Eingabe:	cx	card
Rückgabe:		keine

Sind im System mehr als eine Karte vorhanden, dann kann zwischen den Karten mit dieser Funktion umgeschaltet werden. Dabei beginnt die Numerierung mit Null. Ist nur eine Karte im System vorhanden, dann muss die Funktion nicht verwendet werden.

Funktion 3 **ab SM-41**
Funktionsname: SmResetCard

Eingabe:	keine
Rückgabe:	keine

Bewirkt den sofortigen Abbruch aller laufenden Funktionen.

Funktion 4
Funktionsname: SmInitCard

Eingabe:		keine
Rückgabe:	cx	errcode

Nach erfolgreichem Reset muss die entsprechende Karte mit dieser Funktion initialisiert werden oder es wird mit SmDownload (ab Version 2.0) ein alternatives Programm für den SM-40 Prozessor geladen.

Funktion 5
Funktionsname: SmAccel

Eingabe:	cx - Beschleunigung dx - Motor
Rückgabe:	keine

Für jeden Motor (dx= 0...3) kann eine Beschleunigungsphase festgelegt werden. Die Treibervoreinstellung ist 100 Schritte. Damit lassen sich Schrittmotoren auf längeren Wegen auf höhere Geschrindigkeiten beschleunigen.

Funktion 6 **ab Version 2.0 ohne Funktion**
Funktionsname: SmDecay

Eingabe:	cx - Bremsweg dx - Motor
Rückgabe:	keine

Falls hohe Massen schnell bewegt werden, kann es erforderlich werden, dass am Ende der Bewegung ein langsames Abklingen der Geschwindigkeit erfogen soll.

Funktion 7
Funktionsname: SmSpeed

Eingabe:	cx - Geschwindigkeit dx - Motor
Rückgabe:	keine

Für jeden Motor (dx= 0...3) kann eine maximale Geschwindigkeit festgelegt werden. Die Treibervoreinstellung ist 2000 für SM-40 und 80 für SM-41. Der Wert 1 ist die schnellste Einstellung, der Wert 65535 die langsamste.

Funktion 8
Funktionsname: SmSteps

Eingabe: cx - Schrittanzahl mit Vorzeichen
dx - Motor
Rückgabe: keine

Für jeden Motor (dx= 0...3) kann die Anzahl der Schritte für das nächste SmMoveAndWait oder SmMove Kommando festgelegt werden. Die Treibervoreinstellung ist 0. Der Wert kann im Bereich -32768 bis +32767 liegen, das Vorzeichen gibt die Richtung der Bewegung an.

Funktion 9
Funktionsname: SmSteps32

Eingabe: ecx - Schrittanzahl mit Vorzeichen
dx - Motor
Rückgabe: keine

Für jeden Motor (dx= 0...3) kann die Anzahl der Schritte für das nächste SmMoveAndWait oder SmMove Kommando festgelegt werden. Die Treibervoreinstellung ist 0. Der Wert von ecx (32-Bit-Register) kann im Bereich von -2147483648 bis +2147483647 liegen, das Vorzeichen gibt die Richtung der Bewegung an.

Funktion 10
Funktionsname: SmMoveAndWait

Eingabe: keine
Rückgabe: cx - status

Die zuvor definierte Bewegung wird jetzt ausgeführt. Dabei wird im Treiber gewartet, bis die Bewegung abgeschlossen wurde. Das sollte nur für sehr kurze Bewegungen genutzt werden, weil Rechenzeit während der Bewegung verloren geht. Besser ist die Verwendung der Funktion 11 im Zusammenhang mit timer-gesteuerten Abfragen der Funktion 12, um den Vollzug der Bewegung festzustellen. Ein zu lange Bewegung wird mit der Fehlermeldung status

ungleich 0 abgebrochen.

Funktion 11
Funktionsname: SmMove

Eingabe: keine
Rückgabe: cx - errcode

Die zuvor definierte Bewegung wird mit sofortiger Rückkehr aus dieser Funktion ausgeführt. Falls die Bewegung nicht auslösbar ist, zeigt das errcode mit einem Wert ungleich 0 an.

Funktion 12
Funktionsname: SmStatus

Eingabe: keine
Rückgabe: cx status

Die Statusabfrage zeigt generell an, ob die Schrittmotorplatine irgendwelche Aktionen ausführt. Ein status- Wert ungleich Null zeigt eine laufende Aktion, z.B. eine Bewegung an. Erst mit status=0 dürfen weitere Kommandos gesendet werden.

Funktion 13
Funktionsname: SmHoldPower

Eingabe: cx mode
Rückgabe: keine

In vielen Fällen ist es akzeptabel, die Schrittmotoren nach erfolgter Bewegung stromlos zu schalten. Die Treibervoreinstellung mode=0 bedeutet automatische Stromabschaltung nach Bewegungsende. Der Wert mode=2 bedeutet, dass die Motoren unter vollem Strom auch im Ruhezustzand betrieben werden. Optional (SM-40 Option, schwarzes Relais) kann für Motoren geringer Leistung auch ein reduzierter Haltestrom eingeschaltet werden. Für SM-41 sind dazu externe Komponenten erforderlich.

Ab SM-41 ist der Wert mode=3 neu und hat die Bedeutung, dass ein Motor schon dann stromlos geschaltet wird, wenn andere Achsen noch Bewegungen ausführen. Für mode=0 schaltet SM-41 die Motoren erst dann stromlos, wenn alle Bewegungen abgeschlossen sind.

Ab SM-41 kann der Wert auch für einzelne Motoren gesetzt werden. Die Übergabe von mode+0x100 (100H) adressiert den X-Motor, ein offset von 0x200 adressiert den Y-Motor, sowie 0x300 den Z-Motor und 0x400 den W-Motor.

Funktion 14
Funktionsname: SmStepTab

Eingabe: cx typ (0=Halbschritte 1und 2= Vollschritte)
Rückgabe: keine

Zwischen Halbschritt- und Vollschrittbetrieb schaltet diese Funktion um. Bei der Umschaltung kann es einmalig zu Abweichungen von +/- 3 Schritten kommen. Halbschrittbetrieb bedeutet, dass zwischen 2 Schritten zwei Schrittmotorphasen gleichzeitig aktiv sind, was dann Sinn macht, wenn der Schrittmotor eine Zwischenstellung zwischen zwei Spulen einnimmt. Nicht jeder Schrittmotor ist dazu geeignet. Das hängt von konstruktiven Details und der Anordnung der Magnetfelder ab. In einigen Fällen erhöht sich der Stromverbrauch um ein Vielfaches, weil die Phasen des Motors gegeneinander arbeiten.
Vollschrittbetrieb mit cx=1 bedeutet, dass mit jedem Schritt jeweils nur eine der vier Motorphasen stromführend ist. Vollschrittbetrieb mit cx=2 bedeutet, dass immer zwei Motophasen stromführend sind. Damit wird bei doppeltem Stromverbrauch in der Regel ein höheres Drehmoment des Motors erreicht.

Ab SM-41 kann typ auch die Werte 3 und 4 annehmen, dabei bedeutet 3 die Umschaltung von unipolar in den Bipolarbetrieb mit Halbschritten und 4 die Umschaltung in den Bipolarbetrieb mit Vollschritten. Die Werte 0...4 bedeuten die Umschaltung aller 4 Achsen, wie beim SM-40.
Ab SM-41 kann die Betriebsart jeder einzelnen Achse festgelegt werden. Die Übergabe von typ+0x100H adressiert den X-Motor, ein offset von 0x200 adressiert den Y-Motor, sowie 0x300 den Z-Motor und 0x400 den W-Motor.

Funktion 15
Funktionsname: SmGetAccel

Eingabe: dx Motor
Rückgabe: cx Beschleunigung

Die Werte der Treibervoreinstellung oder die zuletzt gesetzten Werte der Funktion 5 lassen sich zurücklesen.

Funktion 16
Funktionsname: SmGetDecay

Eingabe: dx Motor
Rückgabe: cx Bremsintensität

Die Werte der Treibervoreinstellung oder die zuletzt gesetzten Werte der Funktion 6 lassen sich zurücklesen.

Funktion 17
Funktionsname: SmGetSpeed

Eingabe: dx Motor
Rückgabe: cx Geschwindigkeit

Die Werte der Treibervoreinstellung oder die zuletzt gesetzten Werte der Funktion 7 lassen sich zurücklesen.

Funktion 18
Funktionsname: SmGetSteps

Eingabe: dx Motor
Rückgabe: cx Schrittzahl mit Vorzeichen

Die Werte der Treibervoreinstellung oder die zuletzt gesetzten Werte der Funktion 8 lassen sich zurücklesen.

Funktion 19
Funktionsname: SmGetSteps32

Eingabe: dx Motor
Rückgabe: ecx 32 -bit- Schrittzahl mit Vorzeichen

Die Werte der Treibervoreinstellung oder die zuletzt gesetzten Werte der Funktion 9 lassen sich zurücklesen.

Funktion 20
Funktionsname: SmGoMin

Eingabe: cx -

dx - Motor
Rückgabe: cx - status

Für den Fall, dass Motorachsen mit Endlagenschaltern ausgestattet sind, kann mit dieser Funktion eine Bewegung ausgelöst werden, die durch einen Endlagenschalter abgebrochen wird. Es werden beide Endlagenschalter der zugehörigen Achse abgefragt.

Funktion 21
Funktionsname: SmGoMax

Eingabe: cx -
dx - Motor
Rückgabe: cx - status

Diese Funktion ist bis auf die Bewegungsrichtung mit Funktion 20 identisch.

Funktion 22
Funktionsname: SmSwitches

Eingabe: keine
Rückgabe: cx - Kontakte

Das Datenword in CL hat folgenden Aufbau:

D0 - Xmin
D1 - Xmax
D2 - Ymin
D3 - Ymax

D4 - Zmin
D5 - Zmax
D6 - Wmin
D7 - Wmax

Funktion 23
Funktionsname: SmReleaseMin

Eingabe: cx -
dx - Motor
Rückgabe: cx - status

Für den Fall, dass Motorachsen mit Endlagenschaltern ausgestattet sind, kann mit dieser Funktion eine entgegengesetzte Bewegung ausgelöst werden, die durch einen Endlagenschalter mit Funktion 20 abgebrochen wurde. Es werden beide Endlagenschalter der zugehörigen Achse abgefragt. Die Bewegung erfolgt solange, wie mindestens einer der Endlagenschalter geschlossen bleibt. Die Bewegung erfolgt mit einem Achtel der voreingestellten Geschwindigkeit des selektierten Motors.

Funktion 24
Funktionsname: SmReleaseMax

Eingabe: cx -
dx - Motor
Rückgabe: cx - status

Diese Funktion ist bis auf die Bewegungsrichtung mit Funktion 23 identisch. Diese Funktion wird normalerweise nach Funktion 22 aufgerufen, um geschlossene Endlagenschalter wieder frei zu geben.

Funktion 25 **nur SM-41**
Funktionsname: SmSwitchesDirect

Eingabe: keine
Rückgabe: cl - Kontakte

Da der höherwertige Teil des 16-bit-cx Wertes nicht unbedingt Null sein muß, ist eventuell zuerst die Operation cx=cx & 0xff auszuführen.
Das Datenword in CL hat den in Funktion 22 beschriebenen Aufbau.

Im Gegensatz zu Funktion 22 werden die Endlagenschalter nicht über die CPU des SM-4x abgefragt sondern direkt über die PC-seitig lesbaren Register des SM-41 übermittelt. Damit werden kürzere Reaktionszeiten möglich. Damit können die Kontakte auch abgefragt werden, wenn die SM-4x CPU noch nicht im status “ ready” ist.

Funktion 26 **nur SM-41 ab V3.04**
Funktionsname: SmPosition

Eingabe: dx - Motor
Rückgabe: cx - Position L
dx- Position H

Nach einem Abbruch oder einem Stop durch Endlagenschalter ist je Achse die Anzahl der restlichen Schritte abfragbar.

Funktion 27 **nur SM-41 ab V3.07**
Funktionsname: SmSetAccLength

Eingabe: cl - acclength (Byte)
dx - Motor
Rückgabe:

Der Wert acclength setzt die Länge der Beschleunigungsphase. Voreinstellung acclength=32. Die Werte zwischen 16, 32, 64 und 128 sind gültig.

Es gilt ferner:
Anfangsgeschwindigkeit eines Motors = SmGetSpeed+acclength*SmGetAccel
Endgeschwindigkeit eines Motors=SmGetSpeed+acclength*SmGetDecay

Funktion 28 **nur SM-41 ab V3.07**
Funktionsname: SmSetAccLength

Eingabe: dx - Motor
Rückgabe: cl - acclength (Byte)

Der Wert acclength enthält als Rückgabe die Länge der Beschleunigungsphase aus Funktion 27 oder die Voreinstellung 32.

Hinweise zum Aufrufen von Treiberfunktionen

3.2.1. Microsoft Visual C++ 1.0... 1.52
3.2.2. Microsoft C/C++ 7.0

Ein Weg zum Ansprechen der Treiberfunktionen ist die Nutzung des Inline-Assemblers.
Durch den Inline-Assembler ist die Aktivierung des DPMI-Interfaces durch die Prozedur DrvInit möglich. Diese Funktion wird vor allen weiteren Funktionen, die sich auf den Treiber beziehen aufgerufen.

```
void DrvInit()
{
installed = 0;
_asm        {
            mov     ax, 9909h       ;Treiberkennung
            mov     bx,0            ; initialisiere SM40DRV
            int     60h
                    cmp  cx, 9909h  ;Kennung in bx = erfolgreich
            jnz     short nodpmi
            mov     installed, -1
nodpmi:
                    }
}
```

Die Variable installed ist eine globale C-Variable des types int, die vor dem Aufruf z.B. den Wert 0 erhalten hat. Der Wert installed = -1 kann nachfolgenden Programmteilen zeigen, dass der Treiber bereits erfolgreich aktiviert wurde.
Für DOS Programme würde eine Initialisierung mit:

```
_asm        {
                    mov  bx,0
                    mov  ax,9909h
                    int  60h
                    }
```

genügen.
Auch das folgende C6.0 Beispiel kann bei diesem Compiler zum Einsatz kommen.

3.2.3. Microsoft C PDS/6.0
3.2.4. Microsoft Quick C 2.5
3.2.5. Microsoft Quick C für Windows

```
#include <dos.h>

union REGS inreg,outreg;

int     DrvInit ()
{
inreg.x.ax = 0x9909;            /* Kennung */
inreg.x.bx = 0;                 /* Funktion 0 */
int86   (0x60,  &inreg,  & outreg);
if (outreg.x.cx+outreg.x.bx != 0)
        return 1;
        else
        return 0;
}
```

3.2.6. Borland C++ 3.1, 4.0, 4.5

Dieser Compiler kann sowohl den integrierten Inline-Assembler als auch die im vorherigen Abschnitt beschriebenen C-Funktionen ausführen.
Der Inline-Assembler weist einige Unterschiede zum Microsoft C-Compiler auf. Assembler - Kommentare müssen die Form eines C-Kommentars haben, Labels dürfen nur außerhalb der Assembler Segmente plaziert werden.
Das Beispiel sieht dann folgendermaßen aussehen:

```
void DrvInit()
{
installed=0;
   asm  {
                        mov   ax, 9909h       /*Treiberkennung */
                        mov   bx,0            /*initialisiere SM40DRV */
                        int   60h
                        cmp   bx, 9909h       /*Kennung = bx:erfolgreich */
                jnz     short nodpmi
                        }
  installed=-1;
  nodpmi:                                     /*Label im C-Segment */
}
```


3.3. Microsoft Quick Basic

Das Lesen von sequentiellen Daten und die Treiberaufrufe sind in Quick Basic durch die Funktion INT86 und INT86X möglich.

```
DIM INREG%(7), OUTREG(7)

AX% = 0                 'Indexdefinition für die benötigten
BX% = 1                 'Register
CX% = 2
DX% = 3

INREG%(AX%) = &H9909    'Kennung
INREG%(BX%) = 0         'Funktion 0
CALL INT86 (&H60, VARPTR ( INREG%(0)), VARPTR
        ( OUTREG%(0)))
```

3.4. Microsoft Visual Basic

Es ist besser diese Sprache von der low level Programmierung auszuklammern. Die erforderlichen Funktionen kann man besser in einer anderen Sprache als DLL implementieren und Visual Basic dann verfügbar machen. Das mitgelieferte OCX Conrol enthält den dafür geeigneten Audruf SM40DRV.

3.5. Microsoft Macro-Assembler 6.0
3.6. Microsoft Macro-Assembler 5.1
3.7. Borland Turboassembler

Mit der Einbindung von cmacros.inc durch:

```
        INCLUDE     CMACROS.INC
```

lassen sich Routinen für alle Hochsprachen unter DOS und Windows für jedes Speichermodell erstellen, ohne dass Prozeduren geändert werden müßten.

Beispielsweise würde eine Funktion zur Aktivierung des DPMI Interfaces und der Treiberinitialisierung unter MS-Windows so aussehen können:

```
;.....................................................................
; drvini
```

```
; Aufruf von C/C++:
;       void   drvinit  (int far *  lpinstalled)
;.....................................................................
cProc   drvinit, < PUBLIC,FAR,PASCAL>,<ds>
                parmD     lpinstalled

cBegin
        ldsdi,lpinstalled
        pusha
        mov     ds:[di], word ptr 0
        mov     ax, 9909h ;Treiberkennung
        mov     bx,0              ; initialisiere SM40DRV
        int     60h
        ldsdi,lpinstalled
        cmp     cx, 9909h ;Kennung in cx = erfolgreich
        jnznodpmi
        mov     ds:[di], word ptr 1
nodpmi:
        popa
cEnd
```

3.8. Turbo Pascal für DOS
3.9. Turbo Pascal für Windows

Auch in diesen Sprachen kann von einem Inline-Assembler gebrauch gemacht werden. Wenn die Kommentare durch die in Pascal gültige Syntax ersetzt lässt sich das Beispiel aus Abschnitt 3.2.6. übernehmen. Assemblerkommandos beginnen mit *asm* und enden mit *end;*

4. HaSoTec - Lizenzvertrag

Nachstehend sind die Vertragsbedingungen für die Benutzung von HaSoTec-Hardware und HaSoTec-Software durch Sie, den Anwender, aufgeführt. Durch Öffnen der Verpackung von Datenträgern oder von mit Datenträgern gelieferten Computerplatinen, sowie mit der Erteilung von Softwareentwicklungs- und Installationsaufträgen erklären Sie sich mit diesen Vertragsbedingungen einverstanden. Sofern Sie mit den Bedingungen nicht einverstanden sind, geben Sie bitte das Produkt in ungeöffneter Verpackung und alle anderen Teile des erworbenen Produktes einschließlich allen schriftlichen Materials unverzüglich dort zurück, wo Sie das Produkt erworben haben. Sie erhalten dann den vollen Kaufpreis erstattet. Softwareentwicklungs- und Installationsaufträge sind vom Rückgaberecht ausgeschlossen.

1. Vertragsgegenstand

Gegenstand des Vertrages sind die auf beiliegenden Datenträgern aufgezeichneten Computerprogramme, Computerplatinen, sowie die Bedienhandbücher und sonstiges mitgeliefertes Material. Die Computerprogramme werden im folgenden auch als "Software", die von HaSoTec entwickelten und gelieferten Computerplatinen auch als "Hardware" bezeichnet.

2. Nutzungsrecht

HaSoTec gewährt Ihnen für die Dauer dieses Vertrages das einfache, nicht ausschließliche und persönliche Recht, die Software auf einem einzelnen Computer an einem einzelnen Bildschirmarbeitsplatz zu verwenden. Als Lizenznehmer dürfen Sie die Software in körperlicher Form, gespeichert auf einem Datenträger oder über ein lokales Datennetz von einem Computer auf einen anderen überspielen. Dabei muss sichergestellt sein, dass die Software zu irgendeinem Zeitpunkt immer nur auf einem einzelnen Computer genutzt wird und die unter 4. aufgeführten Beschränkungen eingehalten werden.

3. Erweiterte Lizenzeinräumung

Sofern HaSoTec für Teile der Software entsprechende Nutzungsrechte einräumt, können Sie diese Teile ändern und in von Ihnen erstellte Programme einbinden. Eine Weitergabe ist nur in kompilierter Form als Bestandteil Ihres Programmes möglich. Dazu ist der Copyright-Vermerk von HaSoTec in Ihr Programm mit aufzunehmen. HaSoTec ist bezüglich aller Ansprüche und Kosten, die auf den Gebrauch und der Verteilung dieses Programms zurückzuführen sind, freizustellen.

4. Urheberrecht

Die Software ist Eigentum von HaSoTec oder dessen Lieferanten. Sie erhalten mit dem Erwerb nur Eigentum an den körperlichen Datenträgern. Von der Software darf ausschließlich für Sicherungs- und Archivierungszwecke eine Kopie angefertigt werden.
HaSoTec behält sich alle Veröffentlichungs-, Vervielfältigungs-, Bearbeitungs- und Verwertungsrechte an der Software, sowie Änderungen an der zukünftig unter

gleichem Produktnamen gelieferten Hardware vor.
Es ist ohne schriftliche Einwilligung von HaSoTec untersagt:

- Die Software abzuändern, zu übersetzen, zu entkompilieren oder zu entassemblieren,
- der Hardware Schaltungsideen zu entnehmen oder in der Hardware enthaltene Firmware abzuändern, zu übersetzen, zu entkompilieren oder zu entassemblieren,
- das zum Produkt gehörende schriftliche Material zu kopieren,
- die Hardware oder Software zu vermieten oder zu verleasen.

Eine dauerhafte Übertragung von Hardware oder Software ist nur dann zulässig, wenn Sie keine Kopien der Software zurückbehalten und der Empfänger sich mit den Bestimmungen dieses Vertrages einverstanden erklärt.

5. Gewährleistung

HaSoTec gewährleistet, dass die Software für einen Zeitraum von 6 Monaten ab Empfangsdatum im wesentlichen gemäß den begleitenden Bedienhandbüchern arbeitet. Der Kunde muss sicherstellen, dass beanstandete Mängel, nicht auf von Standards abweichende Computerhardware zurückzuführen ist. Die Gewährleistung wird von HaSoTec als Hersteller des Produktes übernommen und ersetzt oder beschränkt nicht etwaige gesetzliche Gewährleistungs- oder Haftungsansprüche, die Sie gegenüber dem Verkäufer haben, von dem Sie Ihre Produktkopie erworben haben.
Der Gewährleistungsanspruch besteht nach Wahl von HaSoTec in der Rückerstattung des Kaufpreises oder in Ersatzlieferung. Dazu ist das gelieferte Material mit einer Kopie Ihrer Quittung vom Kauf an HaSoTec oder an den Händler, von dem es bezogen wurde, zurückzugeben. Zu diesem Zweck wird von HaSoTec eine RMA-Nummer vergeben, die deutlich sichtbar am Paket anzubringen ist. Wird der Mangel nicht innerhalb angemessener Frist behoben, so kann der Käufer nach seiner Wahl verlangen, dass der Erwerbspreis herabgesetzt oder der Kauf rückgängig gemacht wird.
HaSoTec gewährleistet eine 6 monatige Garantie, die bis auf die Versandkosten kostenlos ist. Die Garantiebedingungen sind in der jeweiligen zum Produkt gelieferten Anwenderdokumentation enthalten.
HaSoTec gewährleistet nicht, dass die Hardware oder Software den speziellen Anforderungen des Käufers oder Nutzers genügt oder mit anderen vom Kunden gewählten Programmen zusammenarbeitet.
Jegliche Form von Gewährleistung kann erst mit der vollständigen Bezahlung des Produktes in Anspruch genommen werden.

6. Haftung

Mit Ausnahme von vorsätzlich oder durch grobe Fahrlässigkeit verursachte Schäden haften weder HaSoTec noch deren Lieferanten für irgendeinen Schaden, der auf die Verwendung der Software oder Hardware zurückzuführen ist. Dies gilt uneingeschränkt auch für entgangenen Geschäftsgewinn, Betriebsunterbrechungen, entgangene Geschäftsinformationen oder Datenverlust sowie für anderen finanziellen Verlust. Auf jeden Fall ist die Haftung von HaSoTec auf den Betrag beschränkt, den der Käufer für das Produkt bezahlt hat. Ansprüche, die auf unabdingbaren Bestimmungen des Produkthaftungsgesetzes beruhen, bleiben von dieser Haftungsbeschränkung unberührt.

7. Dauer des Vertrages

Der Vertrag läuft auf unbestimmte Zeit. Ihr Nutzungsrecht erlischt automatisch, wenn Sie eine der Bedingungen des Vertrages verletzen. In diesem Fall sind die Originaldatenträger, die Originalplatinen und alle Kopien der Software und Hardware einschließlich etwaiger abgeänderter Exemplare sowie das schriftliche Material zu vernichten.

5 Erweiterte Rechte

Alle zu den Updates und zur Platine gelieferten Quellcodes und Bibliotheken können zusammen mit Ihrem Softwareprodukt in compilierter Form solange lizenzfrei weitergegeben werden, wie sichergestellt ist, dass ausschließlich Original HaSoTec Framegrabber mit dieser Software zum Einsatz kommen.

Eine weitere Verbreitung ist auszuschließen und wird strafrechtlich verfolgt.