HaSoTec Framegrabber HC-34

Interface für digitale Kameras

Programmierung und Informationen

Version 4.25D

Inhalt

1 Generelle Hinweise zur Programmierung 7-5

2 32- Bit- Programmierung mit MS-Windows 9x/Me und MS-Windows NT/2000 7-7

2.1 Programmierung unter Microsoft Visual C++ 2.0, 4.0, 4.1, 4.2, 5.0, 6.0 7-7

2.2 Programmierung unter Borland C++ 5.01 und C- Builder 7-13

2.3 Programmierung unter Visual Basic 4.0, 5.0, 6.0 7-14

3 Programmierung auf prozeduraler Ebene mit MS-Windows 3.x und MS-Windows 9x 7-15

3.1 Programmierung in C 7-16

3.1.1 Microsoft Visual C++ 1.0-1.52 7-16

3.1.2 Microsoft C/C++ 7.0 7-17

3.1.3 Borland C++ 3.1, C++ 4.0, C++ 4.5 7-17

3.2 Programmierung in C++ 7-18

3.2.1 Microsoft Visual C++ 1.0 ... 1.51 7-18

3.2.2 Microsoft C/C++ 7.0 7-18

3.3 Programmierung in Pascal 7-19

3.3.1 Borland Turbo Pascal für Windows 7.0 7-19

3.3.2 Borland Delphi 1.0

. 7-20
3.4 Programmierung in Basic unter DOS
. 7-21
3.4.1 Microsoft Quick Basic 4.5
. 7-121
3.5 Programmierung in Pascal unter DOS
. 7-22
3.5.1 Borland Pascal 7.0
. 7-22

4 Low level Programmierung
. 7-23
4.1 Aufbau eines Funktionsaufrufs
. 7-23
4.1.1 Tabellarische Übersicht
. 7-24
4.1.2 Beschreibung der Treiberfunktionen
. 7-27
4.2 Hinweise zum Aufrufen von Treiberfunktionen 7-38
4.2.2 Microsoft C/C++ 7.0 7-39
4.2.3 Microsoft C PDS/6.0
. 7-39
4.2.4 Microsoft Quick C 2.5
. 7-39
4.2.5 Microsoft Quick C für Windows
. 7-39
4.2.6 Borland C++ 3.1, 4.0, 4.5 7-41

. 7-41
4.3 Microsoft Quick Basic
. 7-42
4.4 Microsoft Visual Basic
. 7-44
4.5 Microsoft Macro-Assembler 6.0
. 7-44
4.6 Microsoft Macro-Assembler 5.1

. 7-44
4.7 Borland Turboassembler
. 7-44
4.8 Turbo Pascal für DOS
. 7-45
4.9 Turbo Pascal für Windows
. 7-45

5 Updates
. 7-47

6 HaSoTec - Lizenzvertrag
. 7-48

7 Erweiterte Rechte
. 7-50

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 Unterkapitel 4.1 die vollständige Low- Level- Schnittstelle für den HC-34. Damit ist die Framegrabberkarte vollständig systemübergreifend programmierbar. Die als fertige Applikationen mitgelieferten Programme und sämtliche Bibliotheken, DLLs oder OCX-Controls benutzen genau diese Schnittstelle. Unter Dos, Windows 3.0, Windows 3.1, Windows 3.11, Windows 95, Windows 98 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 OS/2 ab Version 2.0 erfolgt der Aufruf durch einen Device Driver Einsprung mit genau den gleichen Parametern, nur daß ax, bx, cx, und dx in diesem Fall Namen für Variablen eines Funktionsaufrufs sind.

Zu beachten ist unter MS-Windows 3x-9x der zum Einsatz kommende Bildschirmtreiber. Es ist vorteilhaft mit einer Auflösung von 256 (8 Bit/pixel) Farben zu arbeiten. Damit können Grauwertbilder mit Nutzung von Palettenfunktionen in guter Qualität dargestellt werden. 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 Bilder in allen Standards mit und ohne Averaging digitalisiert sowie sämtliche Einstellungen über Dialogboxen vorgenommen werden können. Übrigens enthalten alle Beispiele auch die *.EXE Datei, so daß man sich die Beispiele auch vor der Installation eines Compilers anschauen kann. Wir glauben daß die Quellcodebeispiele für Windows einen schnellen Einstieg in die Windows-

Programmierung ermöglichen. Mit nur wenigen Elementen der Windows API (GlobalAlloc, GlobalLock, GlobalUnlock, GlobalFree, SetDIBBitsToDevice, BITMAPINFOHEADER, Aufbau einer DIB) finden Nutzer die bisher andere Platformen genutzt haben, einen schnellen Einstieg in die Windows-Programmierung.

2 32- Bit- Programmierung mit MS-Windows 9x/Me und MS-Windows NT/2000

Als prozedurale Ebene wird die Nutzung von Bibliotheksfunktionen der zur Karte gelieferten Bibliotheken bezeichnet. Die Nutzung dieser Bibliotheksfunktionen erfordert keine Kenntnisse der Treiberfunktionen von HC34DRV.EXE. 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 HC34DRV.EXE . Die Nutzung von Bibliothekfunktionen schließt die Anwendung von low level Funktionen an anderer Stelle im Anwenderprogramm nicht aus.
Die Organisation von Bildspeichern ist unter MS-Windows durch Device Independend Bitmaps (DIB) standardisiert. Der mit dem Einhalten solcher Standards verbundene Mehraufwand an Programmierarbeit lohnt sich, wenn man bedenkt, daß damit das Zusammenarbeiten mit beliebigen Grafikkarten erreicht wird. Einige Beispiele enthalten auch die resource script Dateien der Bibliothek. Die Anordnung der Dialogboxelemente und das Entfernen nicht gebrauchter Elemente ist damit auf einfache Weise möglich.

2.1 Programmierung unter Microsoft Visual C++ 2.0, 4.0, 4.1, 4.2, 5.0, 6.0

2.1.1. High Level Programmierung unter Windows NT/2000

Unter Windows NT und Windows 2000 gibt es die Besonderheit, dass ein direkter Zugriff auf die Hardwareresourcen vom Nutzerprogramm nicht möglich ist. Beim HC-34 werden die Bilddaten aus I/O Ports sequentiell gelesen. Aus diesem Grund sind die High Level Funktionen fast vollständig im Devicedriver HC34DRV.SYS enthalten. Wird auf High Level Ebene ein OCX-Control, eine Lib- Bibliothek, eine Objektdatei oder eine DLL angesprochen, dann muss sichergestellt sein, das das Control,

bzw. die Bibliothek auch für Windows NT/2000 ausgelegt ist. DLLs, Bibliotheken und Objektdateien gibt es jeweils für Windows NT/2000 und Windows 98/95. Die für Windows 95/98 ausgelegten Komponenten sind unter Windows NT/2000 nicht verwendbar und führen beim ersten I/O Port Befehl zu einer entsprechenden Fehlermeldung.
Unter Windows NT/2000 werden nur 32-Bit-Programme unterstützt. 16-Bit-Programmierbeispiele sind unter Windows NT/2000 nicht bereitstellbar, weil die Compiler diese Plattform nicht unterstützen.

2.1.2. Low Level Programmierung unter Windows NT/2000

Unter Windows NT 3.51, Windows NT 4.0, Windows NT 5.0, Windows 2000 und OS/2 ab Version 2.0 erfolgt der Aufruf durch einen Device Driver Einsprung mit den genau gleichen Parametern, 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    ("\\\\.\\Hc34Dev",
                     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 Unterkapitel 4 behandelt.

Außer den regulären Low Level Aufrufen des Gerätetreibers werden weitere Funktionen für das Auslesen des Bildspeichers der Karte benötigt. Diese Funktionen fasst die folgende Tabelle zusammen:

Windows NT/2000 Beispiel

Im Unterverzeichnis Win9x-NT\HC34NT befindet sich ein 32-bit-Programmierbeispiel für MS-Windows NT. Die Bildaufnahmefunktionen werden direkt vom Gerätetreiber aufgerufen. Die Funktion DevIOCtl wird dazu vom Betriebssystem bereitgestellt. Die symbolischen Funktionsnummern bewirken die Auswahl der gewünschten Bildaufnahmefunktion. In der Datei fgioctl.h sind die Funktionsnummern definiert. Diese Datei soll vom Nutzer nicht geändert werden. In künftigen Treiberversionen kann über diese Datei die Kompatibilität des Nutzerprogramms zum Gerätetreiber sichergestellt werden, selbst dann, wenn sich Funktionszuordnungen ändern. Deshalb sollen im

Nutzerprogramm ausschließlich die symbolischen Namen IMG 002 ,IMG003 usw. verwendet werden.

API Funktion	Grab-ben	Aus-lesen	Daten	X-Auf-lösung	Y-Auf-lösung	Kopf-stehend	Aver-aging	WinNT FNR:
Grau		bits						DevIoCtl
HC34IMGBIGX8	grbflg	8 bit	Grey8	max	max	nein	nein	IMG002
HC34IMGWXHX8	grbflg	8 bit	Grey8	W	H	nein	nein	IMG003
Grau mit Averaging								
HA34IMGBIGX8	ja	8 bit	8,16	max	max	nein	ja	IMG012
HA34IMGWXHX8	ja	8 bit	8,16	W	H	nein	ja	IMG013
Grau in DIB								
HC34DIBBIGX8	grbflg	8 bit	Grey8	max	max	ja	nein	IMG022
HC34DIBWXHX8	grbflg	8 bit	Grey8	W	H	ja	nein	IMG023
Grau 16								
HC34IMGBIGX16	grbflg	16 bit	Grey16	max	max	nein	nein	IMG032
HC34IMGWXHX16	grbflg	16 bit	Grey16	W	H	nein	nein	IMG033
Grau mit Averaging								
HA34IMGBIGX16	ja	16 bit	8,16	max	max	nein	ja	IMG042
HA34IMGWXHX16	ja	16 bit	8,16	W	H	nein	ja	IMG043
HC34IMGXXX	nein	32 bit	32 bit	dwords	-	nein	nein	FUN008
HC34DIBXXX	nein	32 bit	32 bit	W	H	nein	nein	FUN009

Die Parameterübergabe aller Funktionen erfolgt in Form eines Pointers auf die Datenstruktur DirectXParams. Darin muß der pointer für die Ablage der Bilddaten angegeben werden.
Das Window-NT Beispiel wurde mit MS Visual C++ 4.1B übersetzt.
Allen Funktionen wird ein pointer auf eine Datenstruktur DIRECTXPARAMS übergeben:

```
typedef struct
    {
    PBYTE ptr;          // Adresse TopLeft
    ULONG dx;           // Breite in Pixeln
    ULONG zcount;       // Anzahl der Zeilen
    ULONG zoffset;      // Zeilenoffset in Bytes
    ULONG zlen;         // Zeilenlänge in Bytes
    ULONG av            // average anzahl
```

```
ULONG basis;          // basisadresse für direct x
ULONG reserved[8]
} DIRECTXPARAMS;
```

Für die Funktionen 1-99 muss lediglich ein Pointer ptr (Adresse TopLeft des Bildes) gesetzt werden.

Pos.	API Funktion	Grabben	Aus-lesen	Daten	X-Auf-löösung	Y-Auf-löösung	zoom	WinNT FNR:
100	DDRSW08	nein	8	8	dx	zcount	1	FKT090
101	DDRSW15	nein	8	555	dx	zcount	1	FKT091
102	DDRSW16	nein	8	565	dx	zcount	1	FKT092
103	DDRSW24	nein	8	888	dx	zcount	1	FKT093
104	DDRSW32	nein	8	0888	dx	zcount	1	FKT094
105	DDRCO08	nein	16	8	dx	zcount	1	FKT095
106	DDRCO15	nein	16	555	dx	zcount	1	FKT096
107	DDRCO16	nein	16	565	dx	zcount	1	FKT097
108	DDRCO24	nein	16	888	dx	zcount	1	FKT098
109	DDRCO32	nein	16	0888	dx	zcount	1	FKT099
110	DDR2SW08	nein	8	8	dx	zcount	2	FKT100
111	DDR2SW15	nein	8	555	dx	zcount	2	FKT101
112	DDR2SW16	nein	8	565	dx	zcount	2	FKT102
113	DDR2SW24	nein	8	888	dx	zcount	2	FKT103
114	DDR2SW32	nein	8	0888	dx	zcount	2	FKT104
115	DDR2CO08	nein	16	8	dx	zcount	2	FKT105
116	DDR2CO15	nein	16	555	dx	zcount	2	FKT106
117	DDR2CO16	nein	16	565	dx	zcount	2	FKT107
118	DDR2CO24	nein	16	888	dx	zcount	2	FKT108
119	DDR2CO32	nein	16	0888	dx	zcount	2	FKT109
120	DDR4SW08	nein	8	8	dx	zcount	4	FKT110
121	DDR4SW15	nein	8	555	dx	zcount	4	FKT111
122	DDR4SW16	nein	8	565	dx	zcount	4	FKT112
123	DDR4SW24	nein	8	888	dx	zcount	4	FKT113
124	DDR4SW32	nein	8	0888	dx	zcount	4	FKT114
125	DDR4CO08	nein	16	8	dx	zcount	4	FKT115
126	DDR4CO15	nein	16	555	dx	zcount	4	FKT116
127	DDR4CO16	nein	16	565	dx	zcount	4	FKT117
128	DDR4CO24	nein	16	888	dx	zcount	4	FKT118
129	DDR4CO32	nein	16	0888	dx	zcount	4	FKT119

Für Funktionen ab 100 oder deren Namen mit DDR (Direct DRaw) beginnt, wird für interlaced Mode die Funktion zweimal gerufen und zoffset zu zlen addiert, um immer eine Zeile frei zu lassen. Zwischen ungeradem und geradem Halbbild müssen imode*zlen Bytes blind gelesen werden. Für zweifach und vierfach vergrößerte Darstellungen muß zoffset um zlen bzw. 3*zlen erhöht sein. Auch DIBs können mit den DirectX Funktionen beschrieben werden, zoffset kann auch negativ sein.

Windows 95, 98,Me

Im Windows 9x Beispiel werden die Bildaufnahmefunktionen durch die Datei HC34IM32.OBJ realisiert.
Das Windows 95/98 Beispiel enthält folgende Dateien:

```
HC34      OPT        48.640  15.10.98  10:40
HC34      DSW           561  14.10.98  11:46
HC34      ICO           766  01.07.93   2:00
HC34      MAK         2.097  13.10.98  16:44
HC34IM32  OBJ         7.011  15.10.98   9:56
HC34      DSP         3.950  15.10.98  10:16
RESOURCE  H             436  13.10.98   9:53
HC34      APS         5.956  15.10.98  10:13
HC34      NCB       107.520  15.10.98  10:40
HC34      C          20.446  15.10.98  10:39
HC34      H           6.506  13.10.98  13:36
HC34      EXE       106.496  15.10.98  10:39
HC34      DEF           194  14.10.98  16:16
HC34      RC          6.144  15.10.98  10:38
HC34-9X   EXE       106.496  15.10.98  10:39
```

Das Beispiel ist funktionell gleich zur Windows NT Version.
Die Datei HC34-32.EXE ist mit Visual C++ 5.0 übersetzt worden.

Windows NT/2000 und 9x/Me

Um das Beispiel einfach zu halten erfolgte die RS-232 Einbindung der Kamera mit festen Voreinstellungen. Bei der Änderung der Bildgröße oder der Bittiefe wird nicht auf die Fertigmeldung der Kamera gewartet.

Die Darstellung für Bildformate mit 16 Bit erfolgt in diesem Beispiel durch eine DIB mit 16 Bit/ Pixel. Leider gibt es keine Funktion der Windows API, um diese Pixel als Grauwerte darzustellen. Bei der Darstellung werden diese Pixel von momentan als 5+5+5 Bit RGB Werte interpretiert Die fünfte der umschaltbaren Look-Up- Tables definiert 32 Grauwerte im RGB Format.

Averaging ist mit den Faktoren 2, 4, 8, 16, 32, 64, 128 und 256 möglich. Zur Einstellung der Ausschnittgröße und des Averaging Faktors enthält das Beispiel einen Dialog.

2.2 Programmierung unter Borland C++ 5.01 und C- Builder

Das Windows NT Beispiel kann mit C-Builder verwendet werden. Unter Windows 95 bzw. 98 ist das Beispiel nicht lauffähig, weil dieser Compiler im Objekt-Code bzw. Library-Code unzureichend zum Microsoft Standard kompatibel ist. Durch Implementierung eigener Bildausleseroutinen in Kombination mit den Low-Level Funktionen läßt sich dieser Compiler auch unter Windows 9x nutzen.

2.3 Programmierung unter Visual Basic 4.0, 5.0, 6.0

Die Unterstützung durch geeignete Bibliotheken und DLLs erfolgt ab HC-34 Softwareversion 5.10.

3 Programmierung auf prozeduraler Ebene mit MS-Windows 3.x und MS-Windows 9x

3.1 Programmierung in C

Diese Sprache ist für die MS - Windows Programmierung am Besten geeignet, weil die Programmierwerkzeuge einen im Vergleich zu anderen Sprachen besseren Entwicklungsstand haben. Die Anforderungen an den Aufbau von Bibliotheken sind leider zwischen verschiedenen Herstellern unterschiedlich.
16-Bit- Programmierung wird nicht für Windows NT unterstützt.

3.1.1 Microsoft Visual C++ 1.0-1.52
3.1.2 Microsoft C/C++ 7.0

In Verzeichnis HC34-16 sind folgende Dateien enthalten:

```
HC34     APS         3.367  13.10.98  13:17
HC34IMGA OBJ         4.409  14.10.98  11:29
HC34     C          22.897  14.10.98  11:24
HC34     DEF           277  13.10.98   9:53
HC34     VCW           194  14.10.98  11:31
HC34     H           6.506  13.10.98  13:36
HC34     ICO           766  01.07.93   2:00
HC34     PDB         3.052  14.10.98  11:24
HC34     RC          5.674  14.10.98  10:17
HC34     RES         2.265  14.10.98  10:17
HC34     WSP           170  14.10.98  11:31
HC34-16  EXE        45.680  14.10.98  11:29
HC34     MAK         2.097  13.10.98  16:44
HC34     OBJ        16.891  14.10.98  11:24
HC34     BSC        76.265  14.10.98  11:24
RESOURCE H             436  13.10.98   9:53
```

Die Datei HC34IMGA.OBJ enthält folgende Funktionen zum Grabben verschiedener Bildformate:

API Funktion	Grab-ben	Aus-lesen bits	Daten	X-Auf-lösung	Y-Auf-lösung	Kopf-stehend	Aver-aging
Grau							
HC34IMGBIGX8	grbflg	8 bit	Grey8	max	max	nein	nein
HC34IMGWXHX8	grbflg	8 bit	Grey8	W	H	nein	nein
Grau mit Averaging							
HA34IMGBIGX8	ja	8 bit	8,16	max	max	nein	ja
HA34IMGWXHX8	ja	8 bit	8,16	W	H	nein	ja
Grau in DIB							
HC34DIBBIGX8	grbflg	8 bit	Grey8	max	max	ja	nein
HC34DIBWXHX8	grbflg	8 bit	Grey8	W	H	ja	nein
Grau 16							
HC34IMGBIGX16	grbflg	16 bit	Grey16	max	max	nein	nein
HC34IMGWXHX16	grbflg	16 bit	Grey16	W	H	nein	nein
Grau mit Averaging							
HA34IMGBIGX16	ja	16 bit	8,16	max	max	nein	ja
HA34IMGWXHX16	ja	16 bit	8,16	W	H	nein	ja

3.1.3 Borland C++ 3.1, C++ 4.0, C++ 4.5

Die Unterstützung durch geeignete Bibliotheken und DLLs erfolgt ab HC-34 Softwareversion 5.10.

3.2 Programmierung in C++

3.2.1 Microsoft Visual C++ 1.0 ... 1.51
3.2.2 Microsoft C/C++ 7.0

Die objektorientierte C++ Unterstützung durch geeignete Bibliotheken und DLLs erfolgt ab HC-34 Softwareversion 5.10.

3.3 Programmierung in Pascal

3.3.1 Borland Turbo Pascal für Windows 7.0

Die Unterstützung durch geeignete Bibliotheken und DLLs erfolgt ab HC-34 Softwareversion 5.10.

3.3.2 Borland Delphi 1.0 und 16-bit 2.x-3.x

Die Unterstützung durch geeignete Bibliotheken und DLLs erfolgt ab HC-34 Softwareversion 5.10.

3.4 Programmierung in Basic unter DOS

3.4.1 Microsoft Quick Basic 4.5

Die Unterstützung durch geeignete Bibliotheken und DLLs erfolgt ab HC-34 Softwareversion 5.10.

3.5 Programmierung in Pascal unter DOS

3.5.1 Borland Pascal 7.0

Die Unterstützung durch geeignete Bibliotheken und DLLs erfolgt ab HC-34 Softwareversion 5.10.

4 Low level Programmierung

Als low level Programmierung werden im Zusammenhang mit dem HC-34 solche Programmabschnitte bezeichnet, die durch Aufruf des DOS Interrupts 60H die Funktionen des Treibers HC34DRV.EXE benutzen. Diese Art der Programmierung sollte nur dann zur Anwendung kommen, wenn Abläufe oder Funktionen benötigt werden, die in den Bibliotheken nicht enthalten sind.

4.1 Aufbau eines Funktionsaufrufs

Der Treiber HC34DRV.EXE belegt nach dem Aufruf das Softwareinterrupt 60H .
Die Grundstruktur für einen Funktionsaufruf sieht in einem 80x86 Assembler folgendermaßen aus:

```
mov     ax, 9905h
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,9905h
```

wird das Register AX mit der hexadezimalen Zahl 9809 geladen. Durch die Zahl 9809h wird HC34DRV angesprochen. Wenn andere Treiber mit dem im Microsoft SDK propagierten Kennungen durch das Register AX in gleicher Weise umgehen, dann können weitere Treiber gleichzeitig auf das Interrupt 60h installiert werden. Leider lassen Netzwerktreiber in der Regel keine Installation mehrerer Treiber auf Interrupt 60H zu.
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.

4.1.1 Tabellarische Übersicht

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

D - DOS
W - MS-Windows
O - OS/2, Windows NT

ax=9905h			Eingabe		Ausgabe	
bx	**Funktionsname**	**Umg.**	**cx**	**dx**	**cx**	**dx**
00	DrvInit	DWO	-	-	9905h	vers
09	DrvSetBase	DWO	basis	1-kein NT download	-	-
18	DrvSetWS	DWO	timing	-	-	-
41	DrvGetBasis	DWO	-	-	basis	prod-id
42	DrvGetWaits	DWO	-	-	waits	-
50	FlmSetSize	DWO	fxsize	fysize	-	-
51	FlmSetTopLeft	DWO	xpos	ypos	-	-
52	FlmSetStatus	DWO	xstatus	ystatus	-	-
53	FlmFirstFrame	DWO	skip54	-	status	-
54	FlmNextFrame	DWO	-	-	status	-
55	FlmIniNextFrame	DWO	-	-	-	-
56	FlmWaitNextFrame	DWO	-	-	status	-
57	FlmGetStatus	DWO	maske	-	status	-
58	FlmReadBlind	DWO	count	basisoffs	-	-
59	FlmIniStatus	DWO	-	-	-	basis
60	FlmAcq	DWO	-	-	-	basis
61	FlmWait	DWO	-	-	-	basis
62	FlmReRead	DWO	-	-	-	basis
63	RealModeRead	D	CDwords	auf dx:di	-	-
64	ReadDword	DWO	-	-	loword	hiword
65	FlmReadSize	DWO	0, 1		0: xsize 1:frames	0:ysize 1:over-flows
66	SetTrigger	DWO	0,1,2	0,1,2,10	status	
67	SetLutAddress	DWO	anzahl	adresse		
68	SetLutData	DWO	daten	daten		
69	SetLut	DWO	lutnr			
70	AutograbOn	DWO				
71	AutograbOff	DWO				
72	GetServiceStatus	DWO			w300	w30c
73	SetServiceStatus	DWO	w300	w30c		

Die Funktionen 80 bis 119 sind reserviert, weil diese durch beim HC-34 nicht relevante Funktionen der FG-30 serie belegt werden.

4.1.2 Beschreibung der Treiberfunktionen

Funktion 0 **ab Version 2.32**
Funktionsname: DrvInit

Eingabe:	ax	Treiberkennung (MASM: 9905H Basic &H9905 Pascal: $9905 C: 0x9905)
Rückgabe:	dx	Treiberversion * 100h

Diese Funktion sollte jedes Programm sofort nach Programmstart rufen.

Funktion 9
Funktionsname: DrvSetBasis

Eingabe:	cx	Basisadresse des HC-34
	dx	dx=1 kein download im NT-Treiber
Rückgabe:		keine

Gibt die Basisadresse vor, auf der ein HC-34 angesprochen werden soll. Wenn mehrere Karten im System laufen, kann mit dieser Funktion zwischen den Karten umgeschaltet werden.
Im NT Treiber erfolgt durch diese Funktion das Download der Kamerakopfkonfiguration. Sollte zwischen mehreren Basisadressen häufig umgeschaltet werden, so läßt sich mit dx=1 das erneute Download zur Zeiteinsparung abschalten. Im ersten Aufruf nach Systemstart erfolgt immer ein download. Jeder weitere Aufruf wertet den Inhalt des dx Registers aus.

Funktion 18
Funktionsname: DrvSetWS

Eingabe:	cx	Timingkorrektur 0 bis 255
Rückgabe:	keine	

Funktion 41
Funktionsname: DrvGetBasis

Eingabe:		keine
Rückgabe:	cx	Basisadresse

Diese Funktion gibt als Wert die Basisadresse des HC-34 zurück. Die aktuelle

Basisadresse muß vor Aufruf dieser Funktion gesetzt worden sein oder beträgt 360H.

Funktion 42
Funktionsname: DrvGetWaits

Eingabe:		keine
Rückgabe:	cx	Wartetakte (0 oder 1)

Diese Funktion gibt als Wert die im Treiber gesetzten Wartetakte zurück.

Funktion 50
Funktionsname: DrvFlmSetSize

Eingabe:	cx	x - Filmbildgröße
	dx	y - Filmbildgröße
Rückgabe:	cx	x - Filmbildgröße
	dx	y - Filmbildgröße

Setzen der Bildmaße für Bilder und Filmsequenzen variabler Größe. Wirkt nur auf die Funktionen 53 und 54.

Die folgende Tabelle zeigt einige Beispiele für Bilder und Bildausschnitte. Die Beispiele entsprechen den Bildformaten, die im HC34CLIP (Kapitel 3 und 8) gegrabbt werden können.

Funktion/Register	50cx	50dx	51cx	51dx	52cx	52dx	57	53	Pixel	sind
Big Grau 8	0	0	0	0	88E0h	1	ja	ja	4	YYYY
W X H Grau 8	W	H	0	0	88E0h	1	ja	ja	4	YYYY
Big Grau 16	0	0	0	0	8862h	1	ja	ja	2	Y16Y16
W X H Grau 16	W	H	0	0	8862h	1	ja	ja	2	Y16Y16

Für eine Filmbildgröße von x=0 und y=0 ersetzt der Treiber die Werte durch die maximal mögliche Auflösung. Diese Auflösung ist abhängig vom verwendeten Kamerakopf. Weil zu jedem Kamerakopf ein Treiber gehört, enthält der Treiber die Information über die maximale mögliche Auflösung. Die Abfrage der maximal möglichen Auflösung kann durch Aufruf dieser Funktion mit cx=0 und dx=0 erfolgen. Der Treiber gibt die Auflösung mit cx=xmax und dx=ymax zurück.

Funktion 51
Funktionsname: DrvFlmSetTopLeft

Eingabe:	cx	x - Koordinate
	dx	y - Koordinate
Rückgabe:		keine

Diese Werte sind zur Zeit immer 0

Funktion 52
Funktionsname: DrvFlmSetStatus

Eingabe:	cx	x - Statuswort
	dx	y - Statuswort
Rückgabe:		keine

Setzt Digitalisierungsstatus für die Funktionen 53 und 54.
Das x-Statuswort hat folgende Werte für die mit dieser Funktion möglichen Betriebsarten, die in der Tabelle der Funktion 50 gezeigt sind:

Grau 8 Bit:	88E0h, 88E1h
Grau 16 Bit	8862h, 8863h

Das ystatus Wort zeigt die Anzahl der vollständigen Bilder, die zur Datenübernahme in den Speicher übernommen werden sollen.
Die Pipeline hat eine Größe von 256KByte x 36 Für einen Bildausschnitt von 400x300 Pixeln mit 16 BitPixel könnten also 4 aufeinanderfolgende Bilder in der Pipeline zwischengespeichert werden.

Funktion 53
Funktionsname: DrvFlmFirstFrame

Eingabe:	cx	1=Digitalisierung überspringen
Rückgabe:	cx	0= erfolgreich
	dx	Basisadresse (ab Version 4.0)

Digitalisierung eines ersten Bildes nach der Spezifikation der Funktionen 50-52. Die Funktion eignet sich zum Digitalisieren im 8-bit und 16-bit Grauwert Modus. Die Bilddaten werden sequentiell aus folgenden Adressen gelesen:

Grauwerte 8 bit:	Basisadresse	4 Pixel je Wort
Grauwerte 16 bit:	Basisadresse	2 Pixel je Wort

Ab Treiber Version 4.20 bewirkt der Aufruf mit cx=1, daß die nachfolgende Digitalisierung übersprungen wird. Funktion 53 mit cx<>1 entspricht also Funktion 53 mit cx=1 + Funktion 54.

Funktion 54
Funktionsname: DrvFlmNextFrame

Eingabe:	keine	
Rückgabe:	cx	0= erfolgreich
	dx	Basisadresse (ab 4.0)

Digitalisierung weiterer Bilder im gleichen Format des durch Funktion 53 zuvor digitalisierten Bildes. Die Laufzeit dieser Funktion ist kürzer, weil zur Wiederholung der Digitalisierung nur ein Teil der Initialisierungen erforderlich ist.

Funktion 55
Funktionsname: DrvFlmIniNextFrame

Eingabe:	keine
Rückgabe:	keine

Digitalisierung weiterer Bilder im gleichen Format des durch Funktion 53 zuvor digitalisierten Bildes. Die Funktion entspricht Funktion 54, mit dem Unterschied, daß nicht auf das zu digitalisierende Bild gewartet wird.

Funktion 56
Funktionsname: DrvFlmWaitNextFrame

Eingabe:	keine	
Rückgabe:	cx	0= erfolgreich

Diese Funktion kann beispielsweise nach Funktion 55 gerufen werden und wartet bis das zu digitalisierende Bild begonnen hat.

Funktion 57
Funktionsname: DrvFlmGetStatus

Eingabe:	keine	
Rückgabe:	cx	status

Der rückgegebene Statuswert enthält in den einzelnen bits zusätzliche Informationen. Durch die AND- Verknüpfung mit einer Bitmaske sind folgende Informationen abfragbar:

Bitmaske	Statusinformation
2000H	TTL- Triggersignal (D-SUB 15 pin 10)
1000H	Optokoppler Triggersignal
400H	RDY=0 zeigt, daß das erste digitalisierte Halbbild abrufbar ist.

Diese Funktion kehrt unverzüglich zurück und ist für eine schnelle, nicht wartende Statusabfrage geeignet.

Funktion 58
Funktionsname: DrvFlmBlindRead

Eingabe:	cx	Anzahl
	dx	offset zur Basisadresse
Rückgabe:		keine

Diese Funktion dient zum Blindlesen von cx Worten von Basisadresse+dx.

Funktion 59
Funktionsname: DrvFlmIniStatus

Eingabe:	cx	
	dx	Basisadresse
Rückgabe:		keine

Diese Funktion initialisiert den Grabvorgang ohne das Grabben auszulösen. Sie entspricht Funktion 53 ohne Grabben, 59+54 entspricht 53. Weiterhin entspricht 59+55+56 ebenfalls 53. Oder 59+60+61+62 entspricht Funktion 53.

Funktion 60
Funktionsname: DrvFlmAcq

Eingabe:	cx	
	dx	Basisadresse
Rückgabe:		keine

Diese Funktion dient zum Starten des Grabvorgangs.

Funktion 61
Funktionsname: DrvFlmWait

Eingabe:	cx	
	dx	Basisadresse
Rückgabe:		keine

Diese Funktion wartet bis der Grabvorgang beginnt.

Funktion 62
Funktionsname: DrvFlmReRead

Eingabe: keine
Rückgabe: dx Basisadresse

Diese Funktion initialisiert das (ggf. wiederholte) Auslesen des Bildspeichers.

Wodurch unterscheiden sich die Funktionen 53,54,55,56,60, 61 und 62?

60+61+62 entspricht 55+56 und entspricht 54.
Funktion 53 dient der Bilderfassung mit den Einstellungen der Funktionen 50,51,52. Funktion 54 wiederholt die Bilderfassung der Funktion 53. Manchmal soll die Zeit von der Anforderung bis zur Bildübernahme noch für andere Prozesse genutzt werden. Dann kann die Funktion 54 durch 55+56 ersetzt werden, indem zwischen Funktion 55 und 56 noch andere Prozeduren aufgerufen werden. Für den Fall, daß ein Bild mehrfach gelesen werden soll, sind die Funktionen 60,61 und 62 gedacht. Zwischen 60 und 61 können wie zwischen 55 und 56 Prozeduren zur Ausnutzung der Wartezeit gerufen werden, Funktion 62 kann dann mehrfach ohne Wartezeit gerufen werden und setzt den pipeline-pointer zum wiederholten Lesen auf den Bildanfang zurück.

Funktion 63 **nur HC-3x**
Funktionsname: DrvRealModeRead

Eingabe: keine
Rückgabe: cx Anzahl DWORDS
dx:di Schreibpointer

Diese Funktion hat nur für Programme, die unter DOS laufen sollen Bedeutung. Auf eine Segment (dx): Offset (di) Adresse werden fortlaufend Bilddaten geschrieben.

Funktion 64 **nur HC-3x**
Funktionsname: DrvReadDword

Eingabe: keine
Rückgabe: cx - Lower 16 bit
dx - Higher 16 bit

Diese Funktion ist eine Hilfsfunktion für Compiler, die keine 32-Bit-Port-

Funktionen haben.

Funktion 65 **nur HC-3x**
Funktionsname: DrvFlmReadSize

Eingabe: cx=0 cx=1
Rückgabe: cx - x - Größe cx - gegrabbte Bildanzahl
dx - y - Größe dx - Overflow Zähler

Gibt für cx=0 die Bildgröße des zuletzt von der Kamera akquirierten Bildes zurück. Diese Funktion kann zur Bestimmung der Größe des auszulesenden Bildes benutzt werden. Die Angabe der Bildgröße über die Funktion 50 könnte dann ignoriert werden. Wenn die auszulesenden Bildgröße von den der Funktion 50 gesendeten Werte bestimmt wird, dann kann die mit dieser Funktion bestimmte Größe zur Kontrolle benutzt werden. Wenn die Bildgrößendaten sich unterscheiden, dann wurde über die RS232 Schnittstelle der Kamera eine andere Bildgröße mitgeteilt oder die Kamera hat die Bildgröße begrenzt. Es ist zu prüfen, ob in diesem Fall eine Fehlermeldung auszulösen ist.
Ein Aufruf mit cx=1 bewirkt, daß in cx die Anzahl der gegrabbten Bilder und in dx die Anzahl der von der Kamera mitgeteilten Overflows zurückgegeben wird.

Funktion 66 **nur HC-3x**
Funktionsname: DrvSetTrigger

Es werden zwei Triggerarten unterschieden.
1. *Software Triggermode* bedeutet dabei, dass per Software ein Push- Signal an den Kamerakopf geschickt wird, um zeitnah zu diesem Signal ein Bild zu erhalten. Es ist dem Programmierer überlassen, ob zum Auslösen dieses Signals noch ein Triggereingangssignal im Polling abgefragt wird. Ob solche Bits von Printerports oder von den Eingängen der Karte (beispielsweise Funktion 66 mit cx=0, dx=10) benutzt werden, ist dabei offen.
Als Funktionsfolge wird erwartet, dass Basisadresse und Bildgeometrie über die Funktionen 9, 50, 51, 52 festgelegt werden, dass der Software Triggermode mit Funktion 66 (cx=0, dx=0) eingeschaltet wird, dass ggf. Triggereingänge abgefragt werden (z.B. Funktion 66 cx=0, dx=10) und ein Triggersignal über Funktion 66 (cx=0, dx=1) ausgelöst wird und der Triggermode über die Clear- Funktion 66 (cx=0, dx=2) wieder verlassen wird. Die Bilddaten können mit Funktion 56 ausgelesen werden.

2. *Hardware Triggermode* bedeutet, dass ein Triggersignal am

Optokopplereingang des HC-34 das Grabben eines Bildes so auslöst, dass die Kamera im Moment der Bildaufnahme ein per Hardware generiertes Triggersignal erhält und die Grabberkarte per Hardware gesteuert ein Bild in den Bildspeicher des HC-34 übernimmt, dass mit Funktion 56 ausgelesen werden kann.
Als Funktionsfolge wird erwartet, dass Basisadresse und Bildgeometrie über die Funktionen 9, 50, 51, 52 festgelegt werden, dass der Hardware Triggermode mit Funktion 66 (cx=1, dx=0) eingeschaltet wird, dass ein Signal am Eingang des Optokopplers mit seiner positiven Flanke die Triggerung auslöst. Dieser Zeitpunkt ist durch periodische Abfrage per Software mit Funktion 66 cx=0, dx=10 erfassbar, indem das Bit D2 ausgewertet wird. Der Triggermode wird über die Clear- Funktion 66 (cx=2, dx=2) wieder verlassen, auch dann, wenn keine Triggersignale aufgetreten sind. Die Bilddaten können mit Funktion 56 ausgelesen werden.

Eingabe: cx: 0=Software 1=Hardware 2=Hard- und Software
dx: 0=set 1=push 2=clear, 10=status
Rückgabe: cx: keine bzw. status

Beim Aufruf mit dx=0 wird der Triggermodus gesetzt.
Das Triggern kann per Software für cx=0 oder cx=2 erfolgen oder per Hardware für cx=1 und cx=2.
Die Push- Funktion ist ein per Software gesendeter Triggerimpuls, dabei ist dx=1 und cx=0 oder cx=2.
Mit dx=2 wird die Triggerfunktion beendet, auch dann, wenn kein Triggerereignis stattgefunden hat.
dx=10 bewirkt die Rückgabe eines statuswortes mit folgendem Aufbau:

D15				D2	D1	D0
			. . . . .	RDY	TTL	OPTO

dabei ist OPTO das Zustandssignal für Triggerimpulse über den Optokopplereingang, TTL das Zustandssignal für Triggerimpulse über den TTL Eingang und RDY ein statusflag, das anzeigt, wann ein Bild in den Speicher übernommen wurde.

Funktion 67 **nur HC-3x**
Funktionsname: DrvSetLutAddress

Eingabe: cx Anzahl zu setzender Worte
dx LUT Adresse [0...15] [0...3]

Rückgabe: keine

Vorbereitung der LUT Initialisierung für Funktion 68. Derzeitig werden 1024 Worte zur Konvertierung der 10 Datenbits der Kamera verwendet. Derzeitig sind je nach Speicherausstattung 16 bzw. 4 umschaltbare Look up tables erlaubt.

Funktion 68 **nur HC-3x**
Funktionsname: DrvSetLutData

Eingabe: cx Datenwort 2n
dx Datenwort 2n+1
Rückgabe: keine

Diese Funktion wird n-fach zur Datenübergabe gerufen, bis entweder 2n oder 2n+1 der in Funktion 67 geforderten Anzahl der Datenworte entspricht.
Die folgende Tabelle zeigt die genaue Anordnung der Bits im Datenwort:

8bpp							7	6	5	4	3	2	1	0		
16 bpp	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Funktion 69 **nur HC-3x**
Funktionsname: DrvSetLut

Eingabe: cx Lut Nummer
Rückgabe: keine

Diese Funktion bewirkt die sofortige Umschaltung auf die gewählte Look up table.

Funktion 70 **nur HC-3x**
Funktionsname: DrvAutoGrabOn

Eingabe: keine
Rückgabe: keine

Diese Funktion bewirkt, daß bei Triggerfunktionen, nachdem ein Bild in den

HC-34 Speicher übernommen wurde sofort der nächste Triggervorgang automatisch ausgelöst wird.

Funktion 71 **nur HC-3x**
Funktionsname: DrvAutoGrabOff

Eingabe: keine
Rückgabe: keine

Diese Funktion schaltet die Wirkung von Funktion 70 aus.

Funktion 72 **nur HC-3x**
Funktionsname: GetServiceStatus

Eingabe: kein
Rückgabe: cx: statusregister w300
dx: statusregister w30c

Diese Funktion ist eine Servicefunktion. Die Werte für w300 und w30c werden zurückgegeben.

Funktion 73 **nur HC-3x**
Funktionsname: SetServiceStatus

Eingabe: cx: statusregister w300
dx: statusregister w30c
Rückgabe: keine

Diese Funktion ist eine Servicefunktion. Die Werte für w300 und w30c werden gesetzt.

Funktionen 80 bis 119 sind zur Wahrung der Kompatibilität zur FG-30 API reserviert.

4.2 Hinweise zum Aufrufen von Treiberfunktionen

4.2.1 Microsoft Visual C++ 1.0... 1.52
4.2.2 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, 0200h   ;real mode interrupt vector holen
            mov     al, 60h     ;gewünschter interrupt
            int     31h         ;DPMI Aufruf
            or      cx, dx      ;Fehler ?
            jz      short nodpmi
            mov     ax, 9905h   ;Treiberkennung
            mov     bx,0        ; initialisiere HC34DRV
            int     60h
            cmp     cx, 9905h   ;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,9905h
            int     60h
            }
```

genügen.

Bilddaten sind sequentielle Datenströme, die beim HC-34 aus einer 32-Bit-Portadresse lesbar sind

```
void ReadBuffer (pbuffer, maxbuffer, basis)
DWORD  far * pbuffer;
int maxbuffer, basis;
{
for (i=0;i<maxbuffer;i++)   *pbuffer++ = inp_dword (basis);
}

DWORD inp_dword (basis)
int basis;
{
int hi;
int lo;
_asm            {
                mov     ax,9905h
                mov     bx,64
                int     60h
                mov     hi,dx
                mov     lo,cx
                }
return ((DWORD)hi<<16+(DWORD)lo);
}
```

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

4.2.3 Microsoft C PDS/6.0
4.2.4 Microsoft Quick C 2.5
4.2.5 Microsoft Quick C für Windows

```
#include <dos.h>

union REGS inreg,outreg;

int             DrvInit ()
{
inreg.x.ax = 0x9905;        /* Kennung */
inreg.x.bx = 0;             /* Funktion 0 */
int86 (0x60,  &inreg,  & outreg);
if (outreg.x.cx+outreg.x.bx != 0)
 return 1;
 else
 return 0;
}
```

Sequentielle Daten lesen:

```
void ReadBuffer (pbuffer, maxbuffer, basis)
int  far * pbuffer;
int maxbuffer, basis;
{
for (i=0;i<maxbuffer;i++)
                {
                inreg.x.ax=0x9905;
                inreg.x.bx=64;
                int86 (0x60, &inreg, &outreg;
                *pbuffer++ = outreg.x.cx;
                *pbuffer++ = outreg.x.dx;
                }
}
```

4.2.6 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, 0200h   /*real mode interrupt vector holen */
                mov     bl, 60h     /*gewünschter interrupt*/
                int     31h         /*DPMI Aufruf */
                or      cx, dx      /*Fehler ? */
                jz      short nodpmi
                mov     ax, 9905h   /*Treiberkennung */
                mov     bx,0        /*initialisiere HC34DRV */
                int     60h
                cmp     bx, 9905h   /*Kennung = bx:erfolgreich */
                jnz     short nodpmi
                }
 installed=-1;
 nodpmi:                            /*Label im C-Segment */
}
```


4.3 Microsoft Quick Basic

Das Lesen von sequentiellen Daten und die Treiberaufrufe sind in Quick Basic durch die Funktion INT86 und INT86X möglich.
Der Quick Basic Befehl INP (port%) kann nur 8-bit-Daten der Portadressen von 0...255 lesen.

Ein Puffer mit 2048 Worten (16-bit) könnte so gelesen werden:

```
DIM Buffer% (2048)
DIM INARY%(7), OUTARY% (7);
AXREG%=0
BXREG%=1
CXREG%=2
DXREG%=3

INARY%(AXREG%)=&H9905
INARY%(BXREG%)=64

FOR  i%=0 TO 1023
  CALL INT86 (&H60, VARPTR (INARY%(0)), VARPTR(OUTARY%(0)))
  Buffer% (i%*2)=OUTARY%(CXREG%)
  Buffer% (i%*2+1)=OUTARY%(DXREG%)
NEXT i%
```

Die Datei auf der mitgelieferten Diskette QBAS45.EXE enthält außerdem folgende Dateien:

INPW.ASM	Quellcode der Quick Library
INPW.OBJ	Objectcode der Quick Library, damit Sie keinen Assembler benötigen
INPW.QLB	Quick Library

Darin enthalten ist eine Funktion zum direkten 32-Bit Port lesen.
Leider gibt es innerhalb von Quick Basic 4.5 unterschiedliche Formate für die Quick Libraries. Die deutsche und englische Version von Quick Basic unterscheiden sich sogar recht erheblich. Um sicher zu gehen, daß eine für Ihre Version lauffähige Quick Library vorliegt, führen Sie folgende Schritte aus:

1. Kopieren Sie INPW.OBJ in das Verzeichnis, in dem sich der von Quick Basic mitgelieferte Linker LINK.EXE befindet
2. Das Linken einer neuen Quick Library erfolgt mit:

```
link        /QU  inpw, , ,  lib\bqlb45.lib
```

3. Starten Sie nun Quick Basic mit:

```
qb   /l inpw.qlb
```

Für Quick Basic würde die Initialisierung von HC34DRV so aussehen:

Ab Version 3.0

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

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

INREG%(AX%) = &H9905            'Kennung
INREG%(BX%) = 0                 'Funktion 0
CALL INT86 (&H60, VARPTR ( INREG%(0)), VARPTR
 ( OUTREG%(0))))
```

Ab Version 4.5

Laden Sie Quick Basic zusammen mit der zum Quick Basic gelieferten Quick Library QB.LIB. Sie können dann die Funktion CALL INTERRUPT verwenden.
Das folgende Beispiel verwendet die Funktion CALL INT86OLD:

```
$INCLUDE: 'QB.BI'

DIM INREG%(7), OUTREG%(7)
CONST  AX=0, BX=1, CX=2, DX=3

INREG%(AX)= &H9905              'Kennung
INREG%(BX)= 0
CALL INT86OLD (&H60, INREG%(), OUTREG%())
```

Die Initialisierung des DPMI Interfaces könnte auch so erfolgen:

```
TYPE  RegType
 ax   AS INTEGER
 bx   AS INTEGER
 cx   AS INTEGER
 dx   AS INTEGER
 bp   AS INTEGER
 si   AS INTEGER
 diI   AS INTEGER
flags INTEGER
 ds AS INTEGER
 es AS INTEGER
END TYPE


RegType  rgi, rgo

installed% = 0
rgi.ax = &H200              'real mode interrupt vector holen
rgi.bx = &H60               'der gewünschte int-handle
CALL   INTERRUPT   (&H31, rgi, rgo)                 'int 31h
IF   rgo.cx + rgo.dx = 0  THEN  GOTO nodpmi     'Fehler?
rgi.ax = &H9905                                     'Kennung
rgi.bx = 0                                          'fkt 0:  init
CALL   INTERRUPT (&H60, rgi, rgo)
IF  rgo.bx <> &H9905   THEN  GOTO  nodpmi installed% = 1
.... weiter nach mit erfolgreicher Aktivierung des Treibers
nodpmi:
.....  weiter nach fehlerhafter Aktivierung des Treibers
```

4.4 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 Visual Basic Beispiel WINMSVB zeigt das Zusammenspiel mit DLLs.

4.5 Microsoft Macro-Assembler 6.0
4.6 Microsoft Macro-Assembler 5.1
4.7 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
        lds     di,lpinstalled
        pusha
        mov     ds:[di], word ptr 0
        push    ds
        push    di
        mov     ax, 0200h ;real mode interrupt vector ;holen
        mov     bl, 60h   ;gewünschter interrupt
                                                    ;handler
        int     31h                 ; DPMI Aufruf
        or       cx, dx   ;Fehler ?
        jz      nodpmi
        mov     ax, 9905h ;Treiberkennung
        mov     bx,0                ; initialisiere HC34DRV
        int     60h
        pop     di
        pop     ds
        cmp     cx, 9905h ;Kennung in cx = erfolgreich
        jnz     nodpmi
        mov     ds:[di], word ptr 1
nodpmi:
        popa
cEnd
```


4.8 Turbo Pascal für DOS
4.9 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.

Sequentielle Daten können so gelesen werden:

```
procedure  ReadBuffer8Bit (xres,yres,basis : integer;);
var             buffer: array[1..xres,1..yres] of integer;
 x,y   : integer;


begin
for   y:=1   to   yres+1  do
 begin
 for  x:=1 to xres+1  do
                 begin
                 puffer  [x,y] := portw [basis];
                 end
 end
.
.
end;
```

Beachten Sie bitte, daß die Variable xres für 8-bit Daten nur den halben Wert der X-Auflösung haben darf. Es werden 16 - bit - Worte mit jeweils 2 Grauwertpixel gelesen.

5 Updates

Die Vergabe von Versionsnummern für die HC-34 Treiber erfolgt in Anlehnung an die gesamte FG-30 Serie. Globale Treibereigenschaften werden ab einer bestimmten Versionsnummer für alle Produkte der FG-30 Serie wirksam.

Version 4.10	ist die erste Softwareversion für den Framegrabber HC-34
Version 4.12	enthält Korrekturen der Firmware, die positive Auswirkungen auf die korrekte Digitalisierung der Randbereiche haben. Korrekturen erfolgten auch in der Firmware der auf der Platine befindlichen programmierbaren Logigbausteine. Ab 01.10.1998 wird diese Version mit umschaltbaren Basisadressen ausgeliefert.
Version 4.13	Einführung von Funktion 66
Version 4.14	Einführung von Funktion 67, 68
Version 4.15	Geändertes download
Version 4.16	Geändertes download für Triggerung
Version 4.17	Geändertes download für Triggerung von Filmsequenzen
Version 4.18	Geändertes download für Kamera Mode 4
Version 4.19	Geändertes download für Filsequenzen
Version 4.20	Windows NT Treiber, Windows NT Programme und Beispiele. Ergänzungen zur Übereinstimmung mit Windows NT Treiber Funktion 59
Version 4.21	Funktion 9 download abschaltbar im NT- Treiber
Version 4.22	Weitere Kameraköpfe
Version 4.23	Separate Datensätze für 2x HC34 und 2 Kameras
Version 4.24	synchrones und asynchrones Triggern mit >=2 Kameras
Version 4.25	Weitere Linekameraserie der Optologic GmbH

Firmwareänderungen:
Alle Systeme sind zu allen späteren Softwarevarianten vollständig kompatibel. Am 18.05.1999 gab es nur eine ausgelieferte Firmwareversion. Im Fall von Hardwareproblemen ist über die HaSoTec Hotline die Verfügbarkeit neuer Firmwareversionen abfragbar. Auf Wunsch kann dann Ihr vorhandener HC-34 mit beigelegten Rückporto (Deutschland 10,00 DM, Europa 35,00 DM oder a.A.) zur Firmwareaktualisierung an uns eingeschickt werden.

6 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. 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.

7 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.