HaSoTec

Framegrabber FG-32/ FG-34 PCI
FG-31 ISA/ FG-35 Low Profile PCI
FG-33 CardBus (32-bit-PCMCIA)
FG-30 PCMCIA

Programmierung
und
Informationen

Version 4.87

Inhalt

1. Generelle Hinweise zur Programmierung

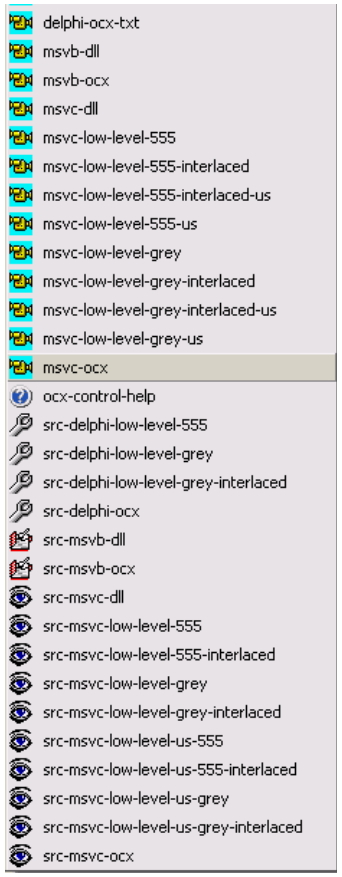
Dieses Kapitel beschreibt die Programmierung auf prozeduraler Ebene (High- Level- Programmierung) und die Programmierung durch direkte Gerätetreiberaufrufe (Low-Level Programmierung).

Für die High- Level Programmierung mit OCX Control ist auch das Kapitel 9 zu beachten. Für die Programmierung unter OS/2 ist auch das Kapitel 8 zu beachten.

Dieses Kapitel beschreibt im Unterkapitel 5.1 die vollständige Low-Level- Schnittstelle der Framegrabber API 9709. Mit dieser Schnittstelle ist die Framegrabber Platine vollständig systemübergreifend programmierbar. API steht für Application Programmers Interface und soll als Schnittstelle zu den Framegrabbern FG31 bis FG35 dienen. 1992 wurde die API 9209 für FG-30 ISA entwickelt und bis Windows Me für FG30 PCMCIA unterstützt. Die 1997 entwickelte API 9709 enthält alle Funktionen des Vorgängers, basiert jedoch statt 16-bit nun auf einer Videodatenbreite von 32-bit.
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 / 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, sowie OS/2 ab Version 2.0 oder Linux 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. Das in Kapitel 9 beschriebene OCX-Control enthält neben den Highlevel Funktion

auch die Low-Level-Schnittstelle in Form einer Funktion FG30DRV (diese Namensgebung ist unabhängig vom Framegrabbertyp der 30iger Serie), die das laufende Betriebssystem automatisch erkennt und die Aufrufe richtig weiterleitet. Ein FG30DRV Aufruf ist auch in allen OS/2 Bibliotheken zu finden.

Zu beachten ist unter MS-Windows 3.x-9x der zum Einsatz kommende Bildschirmtreiber. Es ist vorteilhaft zunächst mit einer Auflösung von 32768 oder 65536 (15 bzw. 16 bit/pixel) Farben zu arbeiten. Damit können Farb- und Grauwertbilder auch ohne Nutzung von Palettenfunktionen in guter Qualität dargestellt werden. Die beschriebenen Quellcodebeispiele sind komplette Applikationen die sich leicht erweitern lassen. Wenn Sie einen der direkt unterstützten Compiler benutzen, dann stehen sofort einfache und komplexe Programme bereit, mit der noch 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. Die Quellcodebeispiele enthalten auch die *.EXE Datei, so dass man sich die Beispiele auch vor der Installation eines Compilers bzw. Vor der Ausführung eines Compilerlaufes anschauen kann. Im Bild links sind die ausführbaren Dateien der Unterprogrammgruppe Win9x-NT als Link mit Kamerasymbolen aufgelistet. Mit den



übrigen Links im Bild links lassen sich die Projektdateien für die jeweiligen Compiler aufrufen. 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, SetDIBBitsTo-Device, BITMAPINFOHEADER, Aufbau einer DIB) finden Nutzer die bisher andere Platformen genutzt haben, einen schnellen Einstieg in die Windows- Programmierung.

1.1 High Level Programmierung unter Windows XP/ 2000/ NT

High- Level- Beispiele verwenden OCX- Conrols, DLLs, Objekt- oder LIB- Dateien. Diese Dateien enthalten relativ komplexe Funktionen, die durch einen einzigen Funktionsaufruf Bilder digitalisieren oder Dialoge öffnen. Für einige dieser Dateien ist es wichtig, die Installations- CD auch unter dem Betriebssystem des Zielsystems zu installieren, denn beispielsweise kann eine DLL für Win98 sich von einer DLL für WinXP intern unterscheiden, selbst wenn nach Außen die gleicher Funktionalität bereitgestellt wird. Das Installationsprogramm identifiziert das Betriebssystem, unter dem es gestartet wird, und installiert die dazu passenden Komponenten.
Wird auf High Level Ebene eine Lib- Bibliothek, eine Objektdatei oder eine DLL angesprochen, dann muss sichergestellt sein, dass die Bibliothek auch für Windows XP/ 2000/ NT ausgelegt ist und bei Auslieferung des Programms dann die für das Zielsystem richtige Bibliothek mitgegeben wird.
Zur Zeit sind alle OCX Controls einer Framegrabber Karte unter verschiedenen Betriebssystemen identische Dateien.
DLLs, Bibliotheken und Objektdateien hingegen 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 I/O Port Befehl zu einer entsprechenden Fehlermeldung.
Unter Windows XP/ 2000/ NT werden nur 32-Bit-Programme installiert und unterstützt.

1.2 Low Level Programmierung unter Windows XP/ 2000/ NT

Unter Windows XP/2000/NT, Linux und OS/2 erfolgt der Aufruf der Framegrabber API durch einen Device- Driver- Einsprung mit zu WinMe/9x/3.x und Dos gleichnamigen Parametern, nur dass bx, cx, und dx in diesem Fall Namen für Variablen eines Funktionsaufrufs sind. Ein Low- Level- Aufruf der Framegrabber-API 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    ("\\\\.\\Fg32Dev",
                     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 API-Funktionen im Unterkapitel 5 behandelt.
Unter Windows XP, 2000 und NT gibt es die Besonderheit, dass ein direkter Zugriff auf die Hardwareresourcen vom Nutzerprogramm aus nicht möglich ist. Die Bilddaten sind über I/O Ports sequentiell lesbar. WinMe/9x/3x und Dos Programme können den Datentransfer deshalb einfach selbst implementieren WinXP/2000/NT Programme benötigen dagegen Datentransfer-Funktionen, die im Device Driver FG32DRV.SYS realisiert sind. Diese Funktionen dienen dem Auslesen des Bildspeichers der Framegrabberkarte und sind in folgender Tabelle zusammengefasst:

Die FG-3x Highlevelfunktionen im Überblick

Pos.	API Funktion	Grabben	Auslesen	Daten	X-Auflöösung	Y-Auflöösung	Norm	Kopfstehend	Averaging	WinNT FNR:

	Grau		bits							DevIoCtl
1	FG32IMG160X120X8	grbflg	8	Grey8	160	120	US	nein	nein	IMG001
2	FG32IMG192X144X8	grbflg	8	Grey8	192	144	Eu	nein	nein	IMG002
3	FG32IMG320X240X8	grbflg	8	Grey8	320	240	US	nein	nein	IMG003
4	FG32IMG384X288X8	grbflg	8	Grey8	384	288	Eu	nein	nein	IMG004
5	FG32IMG640X480X8	grbflg	8	Grey8	640	480	US	nein	nein	IMG005
6	FG32IMG768X576X8	grbflg	8	Grey8	768	576	Eu	nein	nein	IMG006
	Grau mit Averaging									
7	FA32IMG160X120X8	ja	8, 16	Grey8	160	120	US	nein	ja	IMG011
8	FA32IMG192X144X8	ja	8, 16	Grey8	192	144	Eu	nein	ja	IMG012
9	FA32IMG320X240X8	ja	8, 16	Grey8	320	240	US	nein	ja	IMG013
10	FA32IMG384X288X8	ja	8, 16	Grey8	384	288	Eu	nein	ja	IMG014
11	FA32IMG640X480X8	ja	8, 16	Grey8	640	480	US	nein	ja	IMG015
12	FA32IMG768X576X8	ja	8, 16	Grey8	768	576	Eu	nein	ja	IMG016
	Grau in DIB									
13	FG32DIB160X120X8	grbflg	8 bit	Grey8	160	120	US	ja	nein	IMG021
14	FG32DIB192X144X8	grbflg	8 bit	Grey8	192	144	Eu	ja	nein	IMG022
15	FG32DIB320X240X8	grbflg	8 bit	Grey8	320	240	US	ja	nein	IMG023
16	FG32DIB384X288X8	grbflg	8 bit	Grey8	384	288	Eu	ja	nein	IMG024
17	FG32DIB640X480X8	grbflg	8 bit	Grey8	640	480	US	ja	nein	IMG025
18	FG32DIB768X576X8	grbflg	8 bit	Grey8	768	576	Eu	ja	nein	IMG026
	Color 24									
19	FG32IMG160X120X24	grbflg	24, 48	RGB	160	120	US	nein	nein	IMG031
20	FG32IMG192X144X24	grbflg	24, 48	RGB	192	144	Eu	nein	nein	IMG032
21	FG32IMG320X240X24	grbflg	24, 48	RGB	320	240	US	nein	nein	IMG033
22	FG32IMG384X288X24	grbflg	24, 48	RGB	384	288	Eu	nein	nein	IMG034
23	FA32IMG592X442X24US	grbflg	24, 48	RGB	592	442	US	nein	nein	IMG035
24	FA32IMG592X442X24	grbflg	24, 48	RGB	592	442	Eu	nein	nein	IMG036
25	FG32IMG640X480X24	grbflg	24, 48	RGB	640	480	US	nein	nein	IMG037
26	FG32IMG768X576X24	grbflg	24, 48	RGB	768	576	Eu	nein	nein	IMG038
	Color Averaging									
27	FA32IMG160X120X24	ja	24, 48	RGB	160	120	US	nein	ja	IMG041
28	FA32IMG192X144X24	ja	24, 48	RGB	192	144	Eu	nein	ja	IMG042
29	FA32IMG320X240X24	ja	24, 48	RGB	320	240	US	nein	ja	IMG043
30	FA32IMG384X288X24	ja	24, 48	RGB	384	288	Eu	nein	ja	IMG044
31	FA32IMG592X442X24US	ja	24, 48	RGB	592	442	US	ja	nein	IMG045
32	FA32IMG592X442X24	ja	24, 48	RGB	592	442	Eu	ja	nein	IMG046
33	FA32IMG640X480X24	ja	24, 48	RGB	640	480	US	nein	ja	IMG047
34	FA32IMG768X576X24	ja	24, 48	RGB	768	576	Eu	nein	ja	IMG048
	Color DIB									
35	FG32DIB160X120X24	grbflg	24 bit	RGB	160	120	US	ja	nein	IMG051
36	FG32DIB192X144X24	grbflg	24 bit	RGB	192	144	Eu	ja	nein	IMG052
37	FG32DIB320X240X24	grbflg	24 bit	RGB	320	240	US	ja	nein	IMG053
38	FG32DIB384X288X24	grbflg	24 bit	RGB	384	288	Eu	ja	nein	IMG054
39	FG32DIB592X442X24US	grbflg	24 bit	RGB	592	442	US	ja	nein	IMG055
40	FG32DIB592X442X24	grbflg	24 bit	RGB	592	442	Eu	ja	nein	IMG056
41	FG32DIB640X480X24	grbflg	24 bit	RGB	640	480	US	ja	nein	IMG057

42	FG32DIB768X576X24 Color DIB Online	grbflg	24 bit	RGB	768	576	Eu	ja	nein	IMG058
43	FG32DIB160X120X16IN24	ja, 15	ja	RGB	160	120	US	ja	nein	IMG061
44	FG32DIB192X144X16IN24	ja, 15	ja	RGB	192	144	Eu	ja	nein	IMG062
45	FG32DIB320X240X16IN24	ja, 15	ja	RGB	320	240	US	ja	nein	IMG063
46	FG32DIB384X288X16IN24	ja, 15	ja	RGB	384	288	Eu	ja	nein	IMG064
47	FG32DIB592X442X16IN24US	ja, 15	ja	RGB	592	442	US	ja	nein	IMG065
48	FG32DIB592X442X16IN24	ja, 15	ja	RGB	592	442	Eu	ja	nein	IMG066
49	FG32DIB640X480X16IN24	ja, 15	ja	RGB	640	480	US	ja	nein	IMG067
50	FG32DIB768X576X16IN24 Color DIB 15	ja, 15	ja	RGB	768	576	Eu	ja	nein	IMG068
51	FG32DIB160X120X16	ja, 15	ja	555	160	120	US	ja	nein	IMG071
52	FG32DIB192X144X16	ja, 15	ja	555	192	144	Eu	ja	nein	IMG072
53	FG32DIB320X240X16	ja, 15	ja	555	320	240	US	ja	nein	IMG073
54	FG32DIB384X288X16	ja, 15	ja	555	384	288	Eu	ja	nein	IMG074
55	FG32DIB592X442X16US	ja, 15	ja	555	592	442	US	ja	nein	IMG075
56	FG32DIB592X442X16	ja, 15	ja	555	592	442	Eu	ja	nein	IMG076
57	FG32DIB640X480X16	ja, 15	ja	555	640	480	US	ja	nein	IMG077
58	FG32DIB768X576X16 Color 15	ja, 15	ja	555	768	576	Eu	ja	nein	IMG078
59	FG32IMG160X120X16	grbflg	24, 48	555	160	120	US	nein	nein	IMG081
60	FG32IMG192X144X16	grbflg	24, 48	555	192	144	Eu	nein	nein	IMG082
61	FG32IMG320X240X16	grbflg	24, 48	555	320	240	US	nein	nein	IMG083
62	FG32IMG384X288X16	grbflg	24, 48	555	384	288	Eu	nein	nein	IMG084
63	FA32IMG592X442X16US	grbflg	24, 48	555	592	442	US	nein	nein	IMG085
64	FA32IMG592X442X16	grbflg	24, 48	555	592	442	Eu	nein	nein	IMG086
65	FG32IMG640X480X16	grbflg	24, 48	555	640	480	US	nein	nein	IMG087
66	FG32IMG768X576X16 Color 16	grbflg	24, 48	555	768	576	Eu	nein	nein	IMG088
67	FG32IMG160X120X16AS565	grbflg	24, 48	565	160	120	US	nein	nein	IMG091
68	FG32IMG192X144X16AS565	grbflg	24, 48	565	192	144	Eu	nein	nein	IMG092
69	FG32IMG320X240X16AS565	grbflg	24, 48	565	320	240	US	nein	nein	IMG093
70	FG32IMG384X288X16AS565	grbflg	24, 48	565	384	288	Eu	nein	nein	IMG094
71	FA32IMG592X442X16AS565US	grbflg	24, 48	565	592	442	US	nein	nein	IMG095
72	FA32IMG592X442X16AS565	grbflg	24, 48	565	592	442	Eu	nein	nein	IMG096
73	FG32IMG640X480X16AS565	grbflg	24, 48	565	640	480	US	nein	nein	IMG097
74	FG32IMG768X576X16AS565	grbflg	24, 48	565	768	576	Eu	nein	nein	IMG098
75	FG32IMGXXX	nein	32	32 bit	dwords	-	-	nein	nein	FUN009
76	FG32DIBXXX	nein	32	32 bit	xxx	yyy	-	ja	nein	FUN008

100	DDRSW08	nein	8	8	dx	zcount	zoom =	1 FKT090
101	DDRSW15	nein	8	555	dx	zcount	zoom =	1 FKT091
102	DDRSW16	nein	8	565	dx	zcount	zoom =	1 FKT092
103	DDRSW24	nein	8	888	dx	zcount	zoom =	1 FKT093
104	DDRSW32	nein	8	0888	dx	zcount	zoom =	1 FKT094
105	DDRCO08	nein	16	8	dx	zcount	zoom =	1 FKT095
106	DDRCO15	nein	16	555	dx	zcount	zoom =	1 FKT096
107	DDRCO16	nein	16	565	dx	zcount	zoom =	1 FKT097

108	DDRCO24	nein	16	888	dx	zcount	zoom =	1 FKT098
109	DDRCO32	nein	16	0888	dx	zcount	zoom =	1 FKT099
110	DDR2SW08	nein	8	8	dx	zcount	zoom =	2 FKT100
111	DDR2SW15	nein	8	555	dx	zcount	zoom =	2 FKT101
112	DDR2SW16	nein	8	565	dx	zcount	zoom =	2 FKT102
113	DDR2SW24	nein	8	888	dx	zcount	zoom =	2 FKT103
114	DDR2SW32	nein	8	0888	dx	zcount	zoom =	2 FKT104
115	DDR2CO08	nein	16	8	dx	zcount	zoom =	2 FKT105
116	DDR2CO15	nein	16	555	dx	zcount	zoom =	2 FKT106
117	DDR2CO16	nein	16	565	dx	zcount	zoom =	2 FKT107
118	DDR2CO24	nein	16	888	dx	zcount	zoom =	2 FKT108
119	DDR2CO32	nein	16	0888	dx	zcount	zoom =	2 FKT109
120	DDR4SW08	nein	8	8	dx	zcount	zoom =	4 FKT110
121	DDR4SW15	nein	8	555	dx	zcount	zoom =	4 FKT111
122	DDR4SW16	nein	8	565	dx	zcount	zoom =	4 FKT112
123	DDR4SW24	nein	8	888	dx	zcount	zoom =	4 FKT113
124	DDR4SW32	nein	8	0888	dx	zcount	zoom =	4 FKT114
125	DDR4CO08	nein	16	8	dx	zcount	zoom =	4 FKT115
126	DDR4CO15	nein	16	555	dx	zcount	zoom =	4 FKT116
127	DDR4CO16	nein	16	565	dx	zcount	zoom =	4 FKT117
128	DDR4CO24	nein	16	888	dx	zcount	zoom =	4 FKT118
129	DDR4CO32	nein	16	0888	dx	zcount	zoom =	4 FKT119

Die Spalte WinNT Fnr enthält die IoCtl Codes, die in der Datei FgIoctl.h definiert sind.

Allen Funktionen wird ein pointer auf eine einheitliche 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 und basis die korrekte Basisadresse der Framegrabber Karte enthalten.
Für Funktionen ab 100 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. Das ist in den Low- Level- Quellcodebeispielen für die interlaced mode Formate gezeigt. 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.

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

1.4 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 Unterkapitel 5 wird neben der detaillierten Beschreibung der Einzelfunktionen auch der compilerspezifische Aufruf für Interrupt 60H behandelt.

1.5 Low Level Programmierung unter OS/2

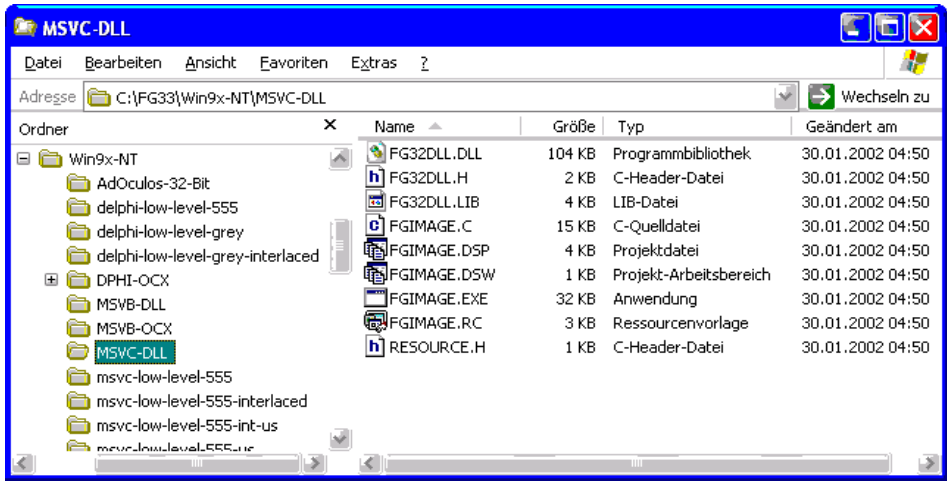
Ähnlich wie unter 1.2. beschrieben erfolgt unter OS/2 der Aufruf über die Funktion DosDevIOCtl. Die Details beschreiben Kapitel 8 und die OS/2 Programmierbeispiele für Borland C und IBM C/2.
Für dieses Betriebssystem gibt es Datentransfer- Befehle, die eine Untermenge der Transferfunktionen für WinXP/2000/NT sind. Alle vorhandenen Transfer- Funktionen werden in den Programmierbeispielen deklariert.

II.

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

2.1 Programmierung unter Microsoft Visual C++ 2.0, 4.0, 4.1, 4.2, 5.0 und 6.0 ohne OCX Control

ab Version 4.20

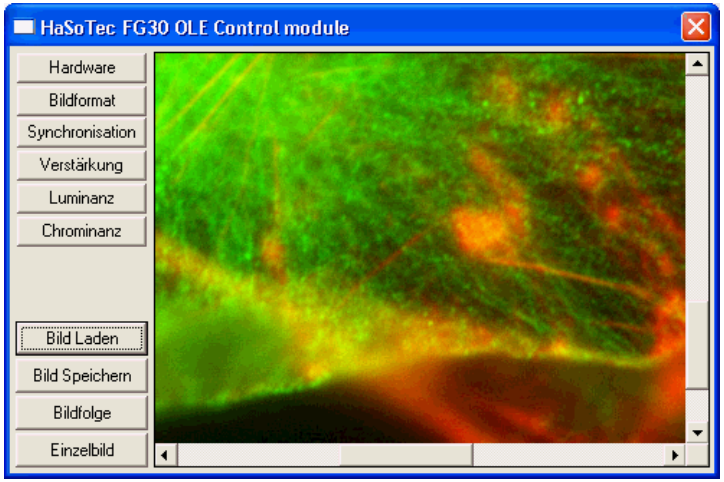


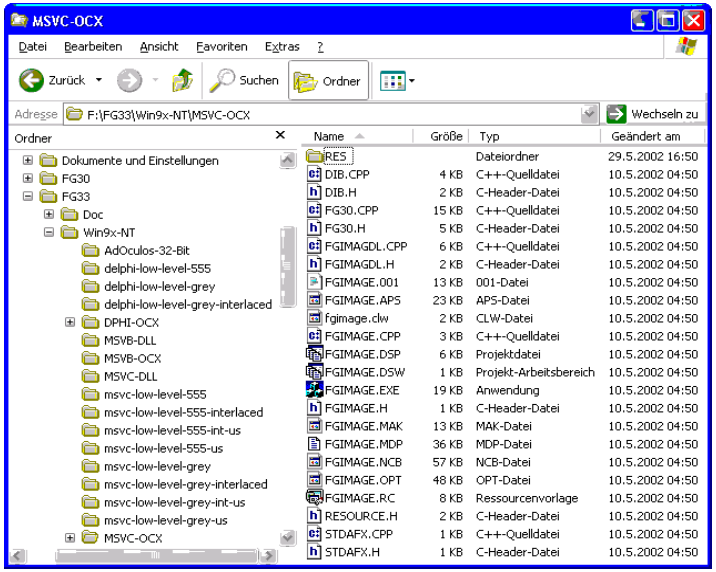
Im Unterverzeichnis ....\Win9x-NT\MSVC-DLL
befinden sich folgende Dateien:

Die Bedienung des Programmes beschreibt Kapitel 6. Die Datei FG32DLL.DLL wird zur Laufzeit benötigt. Für Farb- und Graubilder wird jeweils eine Grab-Funktion und ein Dialog bereitgestellt. Der Dialog erlaubt mit Hilfe von Unterdialogen fast sämtliche Einstellungen der Grabberkarte. Das Beispiel ist funktionell identisch mit den später detailliert beschrieben 16-Bit Bibliotheken.

2.2 Programmierung unter Microsoft Visual C++ 6.0, 5.0 und 4.2 mit OCX Control

Das OCX Control ist in moderne Compiler Umgebungen integrierbar. Die Funktionalität des Controls ist detailliert im Kapitel 9 beschrieben. In Microsoft C++ gibt es Assistenten zur Einbindung von OCX Controls. Ein so generiertes und modifiziertes Beispiel ist installierbar:



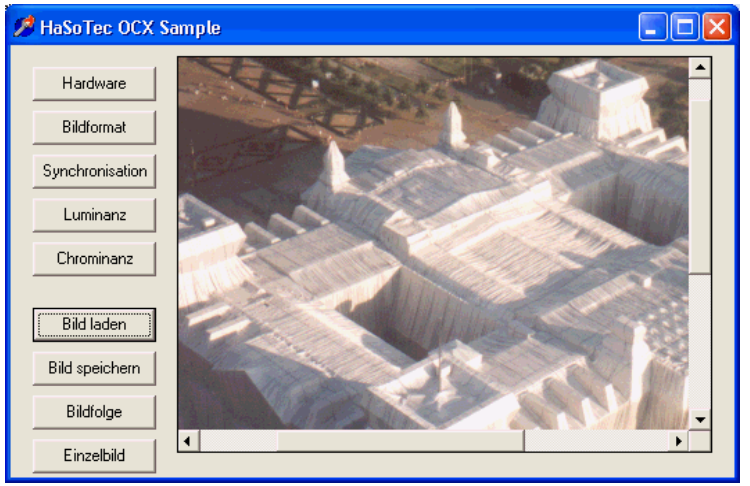


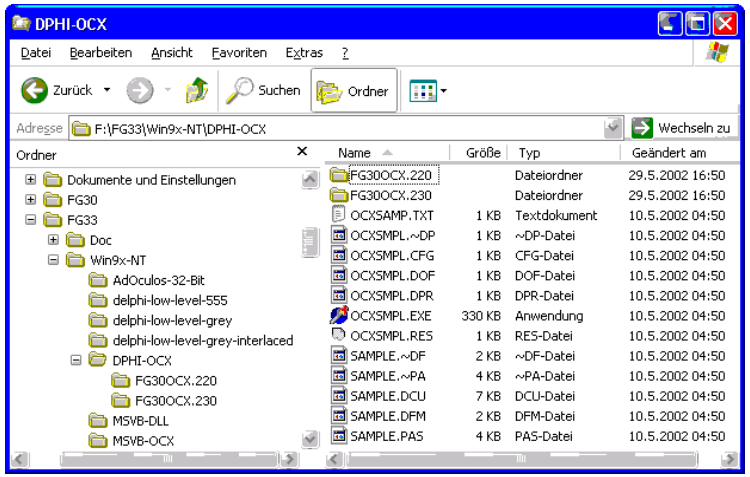
2.3 Programmierung unter Borland Delphi 6.0 5.0, 4.0, 3.00, 3.01, 3.02 und 2.0 ohne OCX Control

Die für Visual C++ und Visual Basic gelieferten Microsoft DLLs sind nicht mit Delphi kompatibel. Low- Level Beispiele sind für alle Formate vorhanden.

2.4 Programmierung unter Borland Delphi 2.0 - 6.0 mit OCX Control

Unter Delphi werden beim Import des OCX Controls automatisch Bibliotheken generiert, die die Einsprünge in das OCX Control enthalten. Älte Delphi Versionen sind mit Controls kompati-bel, die in Unterver-zeichnissen des Bei-spiels zu finden sind.





2.5 Programmierung unter Borland C-Builder mit OCX Control

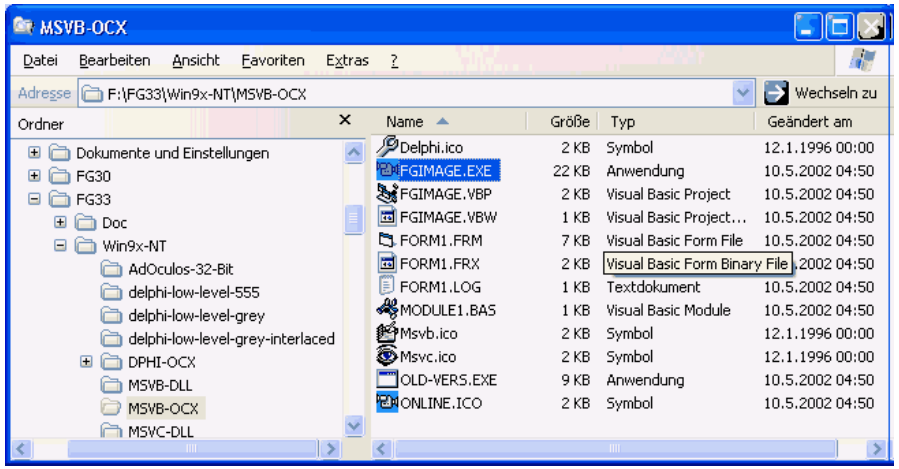
Die Einbettung der Controls ist dem Delphi Beispiel ähnlich. Ein Quellcodebeispiel liegt derzeitig nicht vor. Es ist aus dem Delphi Beispiel ableitbar.

2.6 Programmierung unter Borland C++ 5.01 ohne OCX Control

Microsoft DLLs sind mit diesem Compiler nicht kompatibel. Low-Level- Visual C Beispiele sind dagegen unverändert lauffähig, lediglich einige Kommentare des Inline Assemblers müssen der C-Syntax angepasst werden.

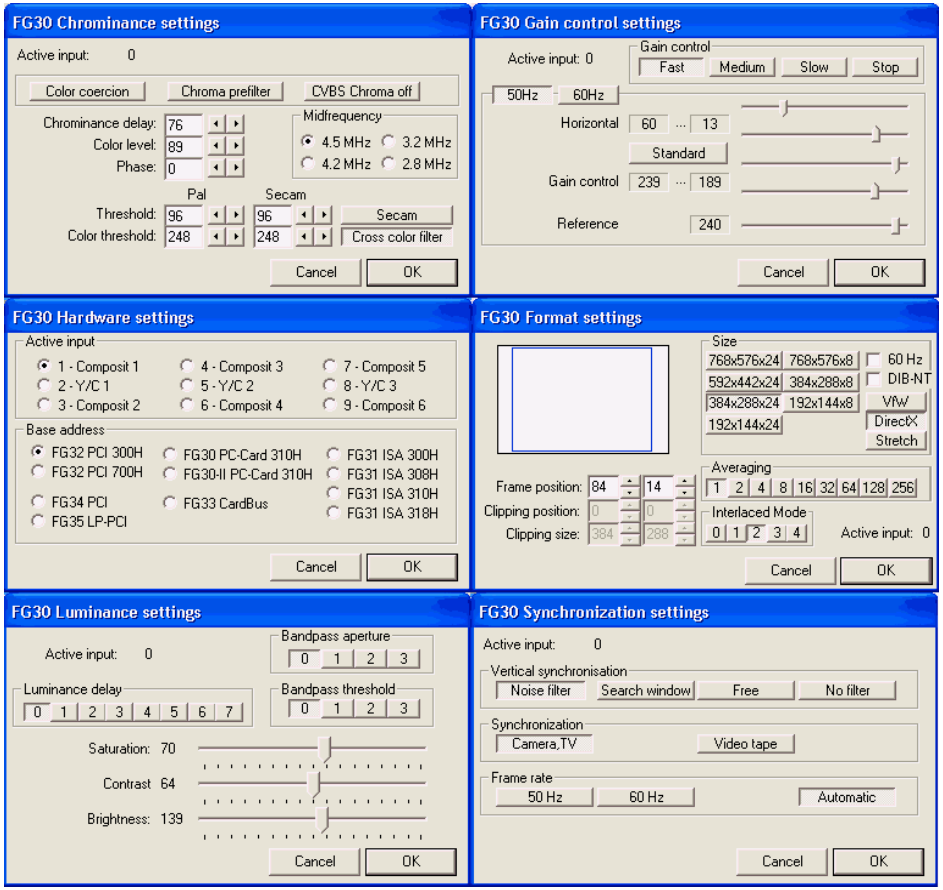
2.7 Programmierung unter Visual Basic 6.0, 5.0 und 4.0 mit OCX Control

Das Aussehen des Programmbeispiels unterscheidet sich nicht



von der C++ Version.

Die Dialoge sind für alle OCX Beispiele identisch und nachfolgend abgebildet:



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

Als prozedurale Ebene wird die Nutzung von Bibliotheksfunktionen der zur Karte gelieferten Bibliotheken bezeichnet. Die Nutzung dieser Bibliotheksfunktionen erfordert keine Kenntnisse der Treiberfunktionen von FG3xDRV.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 FG3xDRV. 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 srcript Dateien der Bibliothek. Die Anordnung der Dialogboxelemente und das Entfernen nicht gebrauchter Elemente ist damit auf einfache Weise möglich.

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.

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

In Verzeichnis WINMSC70 sind folgende Dateien enthalten:

Name	Original	Packed	Ratio	Date	Time	Attr	CRC
FG32IMG.CFG	32	23	76.7%	93-07-01	02:00:00	a--w	D04E
FGIMAGE	512	235	45.9%	93-03-15	01:01:00	a--w	7C06
FGIMAGE.C	13056	3866	29.6%	93-03-15	01:01:00	a--w	6A64
FGIMAGE.DEF	256	171	66.8%	93-03-15	01:01:00	a--w	B029
FGIMAGE.DLG	16896	3589	21.2%	93-07-01	02:00:00	a--w	4AA3
FGIMAGE.EXE	49104	21122	43.0%	93-07-01	02:00:00	a--w	875A
FGIMAGE.H	4096	1022	25.0%	93-07-01	02:00:00	a--w	9C3F
FGIMAGE.ICO	766	169	22.1%	93-03-15	01:01:00	a--w	0492
FGIMAGE.LIB	54303	23318	42.9%	93-07-01	02:00:00	a--w	3E1C
FGIMAGE.RC	512	220	43.0%	93-03-15	01:01:00	a--w	BDFF
README.TXT	363	232	63.9%	93-07-01	02:00:00	a--w	E647
11 files	139894	53967	38.6%	93-07-01	02:00:00		

Die Uhrzeit der Dateien zeigt die Versionsnummer an. Im Interesse des einfachen Übergangs auf neue Versionen sollten nur folgende Dateien durch den Anwender geändert werden:

FGIMAGE.C	-	Quellcodebeispiel
FGIMAGE.DEF	-	modul definition file
FGIMAGE.ICO	-	Icon
FGIMAGE.RC	-	Resourcen des Demobeispiels
FGIMAGE.EXE	-	Das ausführbare Ergebnis
FGIMAGE.CFG	-	Konfigurationsdatei
FGIMAGE.	-	Make / Nmake Datei für MS-C

Diese Dateien liefert Ihnen HaSoTec bei jedem Update

FGIMAGE.DLG	-	Resourcen der Library Dialoge
FGIMAGE.H	-	Datenstrukturen Indexdefinitionen
FGIMAGE.LIB	-	Bibliothek

Die Bibliothek FGIMAGE.LIB enthält alle Funktionen, die im Kapitel 6 (Ad Oculos) beschrieben sind. Das optische Aussehen und die Bedienung ist zum Ad Oculos Treiber völlig äquivalent. Die Bibliothek erfordert einen FG-32 mit HiCOLOR Option.

int FAR PASCAL LibMain (*hwnd*, *r0, r1, lpstr*)

HWND *hwnd*	-	Handle auf das Applikationsfenster
WORD *r0*	-	reserviert = 0
WORD *r1*	-	reserviert = 0
LPSTR *lpstr*	-	reserviert = NULL

LibMain initialisiert die Bibliothek und den FG-32. Dabei wird versucht die Konfigurationsdatei FG32IMG zu lesen.

Rückgabewert	-	bisher immer 0

int FAR PASCAL SNAPDLG (*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

SNAPDLG ist eine Dialogboxfunktion für die Digitalisierung von Grauwertbildern. Neben einer langsamen (über MS-Windows Funktionen) Onlinedarstellung sind alle derzeitig einstellbaren Parameter erreichbar. Diese Funktion läuft auf allen Grafikkarten,

wenngleich eine Grafikkarte mit wenigstens 256 Farben vorteilhaft für die Darstellung der Graustufen ist.

int FAR PASCAL SNAPDLGRGB (*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

SNAPDLGRGB ist eine Dialogboxfunktion für die Digitalisierung von Echtfarbbildern mit 24 bit pro Pixel. Eine bewegte Onlinedarstellung wird automatisch für ET4000 HiColor Grafikkarten aktiviert. Auf allen anderen Grafikkarten lassen sich Einzelbilder in der Dialogbox anzeigen.

int FAR PASCAL TakeFg32Image (*phin, phout, pfgi, pfgo, hwnd*)

HANDLE FAR * *phin*	Pointer auf Handlearray von Eingangsbildern (kann NULL gesetzt werden)
HANDLE FAR * *phout*	Pointer auf Handlearray von Ausgangsbildern (das array kann eindimensional sein)
FGINFO FAR * *pfgi*	Pointer auf eine FGINFO structure (kann NULL gesetzt werden)
FGINFO FAR * *pfgo*	Pointer auf eine FGINFO structure für

Ausgangsbilder

HWND *hwnd* Handle auf das Applikationsfenster

Rückgabewert - enthält Fehlercode oder 0

TakeFG32Image stellt ein eingefrorenes oder ein aktuelles Grauwertbild in Abhängigkeit vom Status der Dialogbox aus SNAPDLG bereit. Die Datenübernahme kann in lineare Datenpuffer oder geräteunabhängige Bitmaps (DIB) erfolgen.

int FAR PASCAL TakeFg32ImageRgb (*phin, phout, pfgi, pfgo, hwnd*)

HANDLE FAR * *phin*
Pointer auf Handlearray von Eingangsbildern (kann NULL gesetzt werden)

HANDLE FAR * *phout*
Pointer auf Handlearray von Ausgangsbildern (das array ist dreidimensional)

FGINFO FAR * *pfgi* Pointer auf eine FGINFO structure (kann NULL gesetzt werden)

FGINFO FAR * *pfgo*
Pointer auf eine FGINFO structure für Ausgangsbilder

HWND *hwnd* Handle des Applikationsfensters

Rückgabewert enthält Fehlercode oder 0

TakeFG32ImageRgb ist das farbige Äquivalent zur Funktion TakeFG32Image.

VOID SizeWindow (*hwnd*)

HWND *hwnd* Handle des Applikationsfensters

SizeWindow optimiert die Größe eines Fensters und positioniert ein Fenster zur Darstellung einer DIB. Bei Bedarf werden automatisch Scrollbars generiert. Die Funktion vermeidet außerdem Scrollbars für kleinere Bilder, bei denen Windows ein Menueleiste zweizeilig darstellt.
Die Bildgröße wird der globalen DIB Handle hdibCurr entnommen.

BOOL MakeDialog (*lpstr,hwnd,farproc*)

LPSTR *lpstr* Dialogname als string
HWND *hwnd* Handle des Applikationsfensters
FARPROC *farproc* far pointer auf Dialogboxprozedur

Rückgabewert true für erfolgreich

Funktion zum Aufruf von Dialogen

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

In der Datei WINBRLND.EXE sind in komprimierter Form folgende Dateien enthalten:

Name	Original	Packed	Ratio	Date	Time	Attr	CRC
FG32IMG.CFG	30	25	83.3%	93-07-01	02:00:00	a--w	C1C3
FGIMAGE.C	12935	3875	30.0%	93-03-15	01:01:00	a--w	7FE5
FGIMAGE.DEF	236	163	69.1%	93-03-15	01:01:00	a--w	D760
FGIMAGE.DLG	16863	3587	21.3%	93-07-01	02:00:00	a--w	B00F
FGIMAGE.DSK	1472	570	38.7%	93-03-15	01:01:00	a--w	6053
FGIMAGE.H	4522	1288	28.5%	93-07-01	02:00:00	a--w	138E
FGIMAGE.ICO	766	169	22.1%	93-07-01	01:01:00	a--w	0492
FGIMAGE.LIB	60416	25954	43.0%	93-07-01	02:00:00	a--w	2900
FGIMAGE.PRJ	3856	988	25.6%	93-03-15	01:01:00	a--w	DF3B
FGIMAGE.RC	935	424	45.3%	93-03-15	01:01:00	a--w	7350
README.TXT	363	232	63.9%	93-07-01	02:00:00	a--w	E647
FGIMAGE.EXE	71919	26211	36.4%	93-07-01	02:00:00	a--w	EA9B
12 files	174313	63486	36.4%	93-07-01	02:00:00		

Die Uhrzeit der Dateien zeigt die Versionsnummer an. Im Interesse des einfachen Übergangs auf neue Versionen sollten nur folgende Dateien durch den Anwender geändert werden:

FGIMAGE.C - Quellcodebeispiel
FGIMAGE.DEF - modul definition file
FGIMAGE.ICO - Icon
FGIMAGE.RC - Resourcen des Demobeispiels
FGIMAGE.EXE - Das ausführbare Ergebnis
FGIMAGE.CFG - Konfigurationsdatei
FGIMAGE.PRJ- Projektdatei
FGIMAGE.DSK - Projektdatei

Diese Dateien liefert Ihnen HaSoTec bei jedem Update

FGIMAGE.DLG - Resourcen der Library Dialoge
FGIMAGE.H - Datenstrukturen Indexdefinitionen
FGIMAGE.LIB - Bibliothek

Die Funktionen der Bibliotheken entsprechen Kapitel 3.1.2.

3.2. Programmierung in C++

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

In der Datei WINMSCPP.EXE sind in komprimierter Form folgende Dateien enthalten:

Name	Original	Packed	Ratio	Date	Time	Attr	CRC
FG32IMG.CFG	30	23	76.7%	93-07-01	02:00:00	a--w	D04E
FGIMAGE.CPP	6528	2104	32.2%	93-07-01	02:00:00	a--w	7A3E
FGIMAGE.DEF	256	155	60.5%	93-07-01	02:00:00	a--w	FA43
FGIMAGE.DLG	16896	3589	21.2%	93-07-01	02:00:00	a--w	4156
FGIMAGE.EXE	71168	32281	45.4%	93-07-01	02:00:00	a--w	997D
FGIMAGE.H	4608	1205	26.2%	93-07-01	02:00:00	a--w	2171
FGIMAGE.ICO	766	169	22.1%	93-03-15	01:01:00	a--w	0492
FGIMAGE.LIB	54303	23318	42.9%	93-07-01	02:00:00	a--w	3E1C
FGIMAGE.RC	2560	683	26.7%	93-07-01	02:00:00	a--w	795B
FGIMAGEC.C	7168	2061	28.8%	93-07-01	02:00:00	a--w	D1D8
MAKEFILE	640	289	45.2%	93-07-01	02:00:00	a--w	C6EF
11 files	164923	65877	39.9%	93-07-01	02:00:00		

Auch hier gilt die Funktionsbeschreibung des Kapitels 3.1.2. Die Dateien FGIMAGE.DLG, FGIMAGE.LIB, FGIMAGE.H sind nicht zur Änderung vorgesehen.

3.3. Programmierung in Basic

3.3.1. Microsoft Visual Basic

Sicherlich ist der Funktionsumfang von Visual Basic 1.0 nicht die

optimale Lösung für das bearbeiten von Bildern.
Die Evolution der Programmiersprache Quick Basic hat jedoch gezeigt, dass in den späten Versionen ein sinnvoll erweiterter Befehlssatz zur Verfügung stand. Mit Visual Basic 2.0, 3.0 oder Visual Basic Professional gibt es schon einige Erweiterungen. Das hier beigelegte Beispiel ist mit der Version 1.0 getestet und dürfte problemlos auch in den Folgeversionen erweiterbar sein. Das Programm ist sehr kurz, weil es in Visual Basic die Möglichkeit gibt der variable **picture** eine Bitmap zuzuweisen, wobei sämtliche Funktionen einer Paint Prozedur entfallen können.
Die im Beispiel enthaltene FG32VB.DLL liefert Funktionen, die den zeilenweisen Zugriff auf die Bildinformation einer Device Independend Bitmap (DIB) erlaubt. Damit werden die in der Sprache fehlenden Funktionen zur Arbeit mit Pointern ausgeglichen. Wie man mit den Bilddaten operieren kann zeigt eine Prozedur zum invertieren eines Grauwertbildes.

Die Datei WINMSVB.EXE enthält in komprimierter Form die Dateien:

Name	Original	Packed	Ratio	Date	Time	Attr	CRC
FG32IMG.CFG	30	24	80.0%	93-07-01	02:00:00	a--w	F27D
FG32VB.DLL	46720	20486	43.8%	93-07-01	02:00:00	a--w	D7EF
FGIMAGE.BAS	1527	426	27.9%	93-07-01	02:00:00	a--w	1086
FGIMAGE.EXE	11041	4297	38.9%	93-07-01	02:00:00	a--w	384B
FGIMAGE.FRM	6747	2479	36.7%	93-07-01	02:00:00	a--w	C423
FGIMAGE.MAK	63	61	96.8%	93-07-01	02:00:00	a--w	FD71
6 files	66128	27773	42.0%	93-07-01	02:00:00		

Die von Visual Basic benutzen Funktionen lassen sich folgendermaßen beschreiben:

Declare Function SNAPDIALOGS Lib "FG32VB.DLL"
(ByVal hWnd, ByVal i As Integer)

hWnd - ist eine Eigenschaft des Applikationsfensters
i - kennzeichnet den zu rufenden Dialog

i=0: ist eine Dialogboxfunktion für die Digitalisierung von Grauwertbildern. Neben einer langsamen Onlinedarstellung über MS-Windows Funktionen sind alle derzeitig einstellbaren Parameter erreichbar. Diese Funktion läuft auf allen Grafikkarten, wenngleich eine Grafikkarte mit wenigstens 256 Farben vorteilhaft für die Darstellung der Graustufen ist.

i=1: ist eine Dialogboxfunktion für die Digitalisierung von Echtfarbbildern mit 24 bit pro Pixel.. Eine bewegte Onlinedarstellung wird automatisch für ET4000 HiColor Grafikkarten aktiviert. Auf allen anderen Grafikkarten lassen sich Einzelbilder in der Dialogbox anzeigen.

Declare Function TakeFg32Img Lib "FG32VB.DLL"
(hin, hout, lpInfIn As FGINFO, lpInfOut As FGINFO, ByVal hWnd)

hin zeigt auf eine Handle eines Eingangsbildes. Derzeitig hat diese Funktion kein Eingangsbild, deshalb kann der Wert von hin=0 sein. Die Zeigerfunktion wird durch das Weglassen des ByVal Bezeichners erreicht.

hout zeigt auf eine Handle des Ausgangsbildes. Der Wert wird durch Aufruf der Windows API Funktion GlobalAlloc gewonnen.

lpInfIn zeigt auf eine FGINFO Typendeklaration für ein Eingangsbild und kann 0 sein

lpInfOut zeigt auf eine FGINFO Typendeklaration für das

Ausgangsbild

hWnd ist eine Eigenschaft des Applikationsfensters

TakeFG32Image stellt ein eingefrorenes oder ein aktuelles Grauwertbild in Abhängigkeit vom Status der Dialogbox aus SnapDialogs (i=0) bereit. Die Datenübernahme erfolgt in einen linearen Datenpuffer.

Declare Function TakeFg32ImgRgb Lib "FG32VB.DLL" (hin, hout, lpInfIn As FGINFO, lpInfOut As FGINFO, ByVal hWnd)

hin zeigt auf ein Zahlenfeld mit 3 Handles von 3 Eingangsbildern. Derzeitig hat diese Funktion keine Eingangsbilder, deshalb können die Werte von hin(1 To 3)=0 sein. Die Zeigerfunktion wird durch das Weglassen des ByVal Bezeichners erreicht.

hout zeigt auf ein Zahlenfeld von Handles von 3 Ausgangsbildern. Die Werte werden durch Aufruf der Windows API Funktion GlobalAlloc gewonnen.

lpInfIn zeigt auf eine FGINFO Typendeklaration für 3 Eingangsbilder und kann 0 sein

lpInfOut zeigt auf eine FGINFO Typendeklaration für 3 Ausgangsbilder (rot, grün, blau)

hWnd ist eine Eigenschaft des Applikationsfensters

TakeFG32ImageRgb stellt ein eingefrorenes oder ein aktuelles Farbbild in Abhängigkeit vom Status der Dialogbox aus SnapDialogs (i=1) bereit. Die Datenübernahme erfolgt in drei lineare Datenpuffer für die drei Farbkomponenten.

Declare Function GetBufLine Lib "FG32VB.DLL" (ByVal pbuf

As Long, ByVal xsize, ByVal y, a As Integer)

pbuf zeigt auf einen Datenpuffer, der beliebig lang sein kann (keine 64K Grenze). Ein solcher Datenpuffer kann mit der Windows API Funktion GlobalAlloc gewonnen werden und mit der Funktion GlobalLock wird der Wert für pbuf übergeben

xsize bezeichnet die Größe eines Segmentes, das aus dem Datenpuffer gewonnen werden soll. Im Beispiel ist das eine Bildzeile

y bezeichnet die Segmentnummer

a bezeichnet ein Datenfeld von Integer Werten, das mindestens die Anzahl xsize Elemente umfassen muß. Ein Element dieses Feldes wird mit jeweils einem Byte des Datenpuffers belegt

Das erste Byte des Puffers beginnt mit der Zeile 0.

Declare Function GetDibLine Lib "FG32VB.DLL" (ByVal pdib As Long, ByVal y, a As Integer)

pdib zeigt auf eine DIB, die beliebig lang sein kann (keine 64K Grenze). Ein solcher Datenpuffer kann mit der Windows API Funktion GlobalAlloc gewonnen werden und mit der Funktion GlobalLock wird der Wert für pbuf erhalten

y bezeichnet die Zeilennummer des Bildes

a bezeichnet ein Datenfeld von Integer Werten, das mindestens die Größe einer Bildzeile umfassen muß. Ein Element dieses Feldes wird mit jeweils einem Byte des Datenpuffers belegt

Diese Funktion liest eine Zeile des Bildes (DIB) in ein Zahlenfeld. Zum Aufruf dieser Funktion muß eine DIB schon einen gültigen BITMAPINFOHEADER haben.

Siehe auch MakeColorHeader und MakeGreyHeader.

Declare Function SetDibLine Lib "FG32VB.DLL" (ByVal pdib As Long, ByVal x, a As Integer)

pdib zeigt auf eine DIB, die beliebig lang sein kann (keine 64K Grenze). Ein solcher Datenpuffer kann mit der Windows API Funktion GlobalAlloc gewonnen werden und mit der Funktion GlobalLock wird der Wert für pbuf erhalten
y bezeichnet die Zeilennummer des Bildes
a bezeichnet ein Datenfeld von Integer Werten, das mindestens die Größe einer Bildzeile umfassen muß. Ein Element dieses Feldes wird mit jeweils einem Byte des Datenpuffers belegt

Diese Funktion schreibt eine Zeile des Bildes (DIB) die in einem Zahlenfeld gespeichert ist. Zum Aufruf dieser Funktion muß eine DIB schon einen gültigen BITMAPINFOHEADER haben.
Siehe auch MakeColorHeader und MakeGreyHeader.

Declare Function MakeGreyHeader Lib "FG32VB.DLL" (ByVal pdib As Long, ByVal x, ByVal y)

pdib zeigt auf eine DIB. Ein Datenpuffer für eine DIB kann mit der Windows API Funktion GlobalAlloc gewonnen werden und mit der Funktion GlobalLock wird der Wert für pdib erhalten
x bezeichnet die Anzahl der Pixel je Zeile des Bildes der DIB
y bezeichnet die Zeilenanzahl des Bildes der DIB

Diese Funktion generiert auf der Adresse pdib einen BIT-MAPINFOHEADER und eine Grauwertpalette

Declare Function MakeColorHeader Lib "FG32VB.DLL" (ByVal pdib As Long, ByVal x, ByVal y)

pdib zeigt auf eine DIB. Ein solcher Datenpuffer für eine DIB kann mit der Windows API Funktion GlobalAlloc gewonnen werden und mit der Funktion GlobalLock wird der Wert für pbuf erhalten
x bezeichnet die Anzahl der Pixel je Zeile des Bildes der DIB
y bezeichnet die Zeilenanzahl des Bildes der DIB
Diese Funktion generiert auf der Adresse pdib einen BIT-MAPINFOHEADER. Die DIB enthält keine Palette und ist für Farbwerte im RGB Format mit 24 Bit / Pixel vorbereitet.

Declare Function SizeWindow Lib "FG32VB.DLL" (ByVal hWnd, ByVal hdib As Integer)

hWnd ist eine Eigenschaft des Aplikationsfensters
hdib ist eine Handle für eine gültige DIB, die durch die Funktion GlobalAlloc erhalten wurde

Diese Funktion passt die Größe des Applikationsfensters der Größe des Bildes der DIB an.

Declare Function WriteBmpFromDib Lib "FG32VB.DLL" (ByVal dibptr As Long)

dibptr zeigt auf den Puffer einer DIB. Der Wert von dibptr wird durch die Windows API Funktion GlobalLock erhalten

Die Funktion schreibt eine Bilddatei unter dem Namen FGIMAGE.BMP . Dateien dieser Art können auch von anderen Programmen z.B. Paintbrush gelesen werden.

3.4. Programmierung in Pascal

3.4.1. Borland Turbo Pascal für Windows 7.0

Die Datei WINPAS70.EXE enthält in komprimierter Form folgende Dateien:

Name	Original	Packed	Ratio	Date	Time	Type	CRC
FGIMAGE.PAS	8934	2365	26.5%	93-07-01	02:00:00	-lh5-	BE7B
FGTPWLIB.DLL	47296	20755	43.9%	93-07-01	02:00:00	-lh5-	FB78
FGIMAGE.EXE	58900	27768	47.1%	93-07-01	02:00:00	-lh5-	C5C3
FGIMAGE.RES	178	108	60.7%	93-07-01	02:00:00	-lh5-	7361
FG32IMG.CFG	30	25	83.3%	93-07-01	02:00:00	-lh5-	8F45
5 files	115338	51021	44.2%	93-07-01	02:00:00		

function SnapDialogs (Window: HWnd; i: Word):Word;**far**;
external 'FGTPWLIB' **name** 'SNAPDIALOGS';

hWnd Handle des Applikationsfensters
i kennzeichnet den zu rufenden Dialog

i=0: ist eine Dialogboxfunktion für die Digitalisierung von Grauwertbildern. Neben einer langsamen Onlinedarstellung über MS-Windows Funktionen sind alle derzeitig einstellbaren Parameter erreichbar. Diese Funktion läuft auf allen Grafikkarten, wenngleich eine Grafikkarte mit wenigstens 256 Farben vorteilhaft für die Darstellung der Graustufen ist.

i=1: ist eine Dialogboxfunktion für die Digitalisierung von Echtfarbbildern mit 24 bit pro Pixel.. Eine bewegte Onlinedarstellung wird automatisch für ET4000 HiColor Grafikkarten aktiviert. Auf allen anderen Grafikkarten lassen sich Einzelbilder in der Dialogbox anzeigen.

function TakeFg32Img (lphin: LPHandle; lphout:

LPHandle; fg: LPFGINFO; fg: LPFGINFO;
Window: HWND):Word;**far**; **external**
'FGTPWLIB' **name** 'TAKEFG32IMG';

lphin zeigt auf eine Handle eines Eingangsbildes. Derzeitig hat diese Funktion kein Eingangsbild, deshalb kann der Wert von hin=0 sein. Die Zeigerfunktion wird durch das weglassen des ByVal Bezeichners erreicht.

lphout zeigt auf eine Handle des Ausgangsbildes. Der Wert wird durch Aufruf der Windows API Funktion GlobalAlloc gewonnen.

pfgi zeigt auf eine FGINFO Typendeklaration für ein Eingangsbild und kann 0 sein

pfgozeigt auf eine FGINFO Typendeklaration für das Ausgangsbild

Window Handle des Applikationsfensters

TakeFG32Image stellt ein eingefrorenes oder ein aktuelles Grauwertbild in Abhängigkeit vom Status der Dialogbox aus SnapDialogs (i=0) bereit. Die Datenübernahme erfolgt in einen linearen Datenpuffer.

function TakeFg32ImgRgb (lphin: LPHandle;
lphout: LPHandle; fg: LPFGINFO; fg: LPFGINFO;
Window: HWND):Word;**far**; **external**
'FGTPWLIB' **name** 'TAKEFG32IMGRGB';

hin zeigt auf ein Zahlenfeld mit 3 Handles von 3 Eingangsbildern. Derzeitig hat diese Funktion keine Eingangsbilder, deshalb können die Werte von hin (1..3) = 0 sein. Die Zeigerfunktion wird durch das weglassen des ByVal Bezeichners erreicht.

hout zeigt auf ein Zahlenfeld von Handles von 3

Ausgangsbildern. Die Werte werden durch Aufruf der Windows API Funktion GlobalAlloc gewonnen.

lpInfIn zeigt auf eine FGINFO Typendeklaration für 3 Eingangsbilder und kann 0 sein

lpInfOut zeigt auf eine FGINFO Typendeklaration für 3 Ausgangsbilder (rot, grün, blau)

hWnd Handle des Applikationsfensters

TakeFG32ImageRgb stellt ein eingefrorenes oder ein aktuelles Farbbild in Abhängigkeit vom Status der Dialogbox aus SnapDialogs (i=1) bereit. Die Datenübernahme erfolgt in drei lineare Datenpuffer für die drei Farbkomponenten.

```
function MakeGreyHeader (lpdib: LPDIBINFO; x: Word;
    y: Word):Word;far; external  'FGTPWLIB'
    name 'MAKEGREYHEADER';
```

lpdib zeigt auf eine DIB. Ein Datenpuffer für eine DIB kann mit der Windows API Funktion GlobalAlloc gewonnen werden und mit der Funktion GlobalLock wird der Wert für lpdib erhalten

x bezeichnet die Anzahl der Pixel je Zeile des Bildes der DIB

y bezeichnet die Zeilenanzahl des Bildes der DIB

Diese Funktion generiert auf der Adresse lpdib einen BITMAPINFOHEADER und eine Grauwertpalette

```
function MakeColorHeader (lpdib:LPDIBINFO; x: Word;
    y: Word):Word;far; external 'FGTPWLIB'
    name 'MAKECOLORHEADER';
```

lpdib zeigt auf eine DIB. Ein solcher Datenpuffer für eine DIB kann mit der Windows API Funktion GlobalAlloc

gewonnen werden und mit der Funktion GlobalLock wird der Wert für lpdib erhalten

x bezeichnet die Anzahl der Pixel je Zeile des Bildes der DIB

y bezeichnet die Zeilenanzahl des Bildes der DIB

Diese Funktion generiert auf der Adresse lpdib einen BITMAPINFOHEADER. Die DIB enthält keine Palette und ist für Farbwerte im RGB Format mit 24 Bit / Pixel vorbereitet.

```
function PlaceDibBits (buf: PCHAR; dib: PCHAR ;
     x: Word; y: Word):Word;far; external 'FGTPWLIB'
    name 'PLACEDIBBITS';
```

buf Pointer auf Puffer mit linearen Bilddaten

dib Pointer auf DIB

x Anzahl Pixel je Zeile

y Anzahl der Zeilen

Die Funktion kopiert Bilddaten in der für DIBs erforderlichen Reihenfolge.

```
function PlaceDibBitsRgb (bufr: PCHAR; bufg: PCHAR;
    bufb: PCHAR; dib: PCHAR ; x: Word;
    y: Word):Word;far; external 'FGTPWLIB'
    name 'PLACEDIBBITSRGB';
```

bufr Pointer auf Puffer mit roten Bilddaten

bufg Pointer auf Puffer mit grünen Bilddaten

bufb Pointer auf Puffer mit blauen Bilddaten

dib Pointer auf DIB

x Anzahl Pixel je Zeile

y Anzahl der Zeilen

Die Funktion kopiert Bilddaten in der für DIBs erforderlichen Reihenfolge.

function SizeWindow (Window: HWND; lpbi:
LPDIBINFO):Word;**far**; **external** 'FGTPWLIB'
name 'SIZEWINDOW';

hWnd Handle des Aplikationsfensters
hdibist eine Handle für eine gültige DIB, die durch die Funktion GlobalAlloc erhalten wurde

Diese Funktion passt die Größe des Applikationsfensters der Größe des Bildes der DIB an.

3.4.2. Borland Delphi 16-Bit

In Borland Delphi 16-Bit wird dem Programmierer ein einfaches Werkzeug zur Ansteuerung des HaSoTec Framegrabbers FG-32 zur Verfügung gestellt. Die im Lieferumfanges Framegrabbers FG-32 enthaltene Komponente FG32KOMP realisiert den Prozeß des Bildgrabbens und der Bildschirmdarstellung gleichermaßen. Dem Nutzer steht die Schnittstelle zum linearen Bildpuffer der Komponente zur Verfügung, die individuell ausgebaut werden kann. Das eigentliche Speicherobjekt TFG32Bitmap ist voll kompatibel zum Standardobjekttyp TBitmap und kann direkt zugewiesen werden Die FG32KOMP Komponente basiert auf der Bibliothek FGTPWLIB.DLL nebst dem Gerätetreiber Fg3xdrv und ist unter Microsoft Windows 3.x - 9x mit Borland Delphi 1.0 und den 16-Bit Compilern höherer Delphi Versionen lauffähig. Für die Nutzung der 32-Bit Delphi Compiler steht das in Kapitel 8 beschriebene OCX-Control zur Verfügung.

Zur Installation der Komponente sind folgende Schritte erforderlich

1. Installation der FG32komp Komponente in der Komponentenpalette
2. Installation der Entwicklungshilfe
3. Installation der Beispieldateien

Starten Sie das Installationsprogramm "setup.exe" aus dem DINSTALL Verzeichnis und folgen Sie den dort gegeben Hinweisen. Notieren Sie sich die von Ihnen gewählten Verzeichnisse, um diese bei der späteren Anmeldung in Delphi zur Hand zu haben. Beachten Sie dabei bitte die Dateistruktur Ihrer Delphi-Installation. Üblicherweise stehen alle Komponenten im Verzeichnis \\Delphi\Lib. Sie können auch ein anderes Verzeichnis verwenden, jedoch darf die Summe der Länge aller Verzeichnisse für Komponenten der VCL 128 Zeichen nicht überschreiten.

Zur Installation der FG-32 Komponente ist nach erfolgtem Start von Delphi im Menü Optionen der Punkt "...Komponente installieren" zu wählen. Damit erscheint eine Dialogbox und nach Wahl von "Hinzufügen" wird der Pfad zur Komponente angegeben. Sobald in der Dialogbox die Dateierweiterung "*.DCU" gewählt ist, wird die Komponente FG32KOMP.DCU auswählbar. Nach anschließender, automatischer Neuübersetzung der Komponentenbibliothek COMPLIB steht in der Menüleiste im Unterpunkt "Additional" die Komponente FGIMAGE zur Verfügung.

Voraussetzung für die erfolgreiche Übersetzung ist die Verfügbarkeit der Bibliothek FGTPWLIB.DLL im Verzeichnis der Komponente bzw. im Verzeichnis \\Windows\System. Auch bei späterer Anwendung ist auf die ständige Verfügbarkeit der Bibliothek FGTPWLIB.DLL zu

achten. Um dies sicherzustellen, empfehlen wir, diese DLL schon nach der Installation in das Windows Systemverzeichnis zu kopieren. Falls die Komponente bei Ausführung die Konfiguration FG32IMG.CFG nicht finden kann, werden Standardwerte verwendet und zur Neukonfiguration aufgefordert. Üblicherweise liegen DLL und Konfigurationsdatei mit der FG-32 Anwendung in einem gemeinsamen Verzeichnis.
Zur Installation der Hilfe verwenden Sie das zu Ihrem Delphi Paket gelieferte Programm "HelpInst". Wählen Sie nach Start dieses Programms die Datei delphi.hdx bzw. Ihren spezifischen Hilfeindex aus dem "\\delphi\Bin" Verzeichnis. Nun sollten alle installierten Schlüsselwortdateien angezeigt werden. Die neue Datei "FG32KOMP.KWF" kann durch Wahl des Menupunktes "Schlüsselwortdatei hinzufügen" installiert werden. Beachten Sie, das die eigentliche Hilfedatei (*.hlp) im Zugriff von Delphi stehen muß. Dies wird zum Beispiel durch Kopieren dieser Datei fg32komp.hlp in das \\Delphi\Bin Verzeichnis erreicht.

FG32Image ist ein Container, bestehend aus einer Zeichenfläche und dem Bilddatenbereich IMAGE vom Typ TFG32Bitmap, welcher zum Objekttyp TBitmap kompatibel ist. Obwohl nicht von ihr abgeleitet ähnelt FG32IMAGE stark der im Lieferumfang von Delphi enthaltenen Komponente TIMAGE.

Hilfsfunktionen der Unit

IncW (p:pointer, tooadd:word)	Hilfsfunktion zum Inkrementieren von Pointern über Segmentgrenzen hinweg

Datentypen von TFG32Image

Bezeichnung	Bereich	Beschreibung

Imgkinds	(NONE, TRUECOLOR, GREYSCALE, PALCOLOR)	Gibt den in der Bitmapkomponente enthaltenen Bildtyp an
Badkinds	(H300,H308,H310, H318,H320,H328, u.s.w. bis H378)	In diesem Mengentyp sind alle möglichen Werte für die Basisadresse des FG-32 definiert.
Anskinds	(Obere_Buchse, Untere_Buchse, Mini_DIN_Buchse)	Mengentyp für die Deklaration des aktuellen Videoeinganges der FG32.
TFG32Bitmap	s.a Übersicht zu TFG32Bitmap	Von TBitmap abgeleitete und zu ihr kompatible Klasse zur Verwaltung des Rasterbild-speichers sowie Implemen-tierung von Basisprozeduren der Framegrabberverwaltung. Sie können zusätzlich alle Merkmale der Basisklasse verwenden
TFG32Filtereve nt	procedure (Sender: TObject;Filterindex :word) of object;	Typ für das onFilter Ereignis

Eigenschaften von TFG32Image

Name der Eigenschaft	Typ	nur zur Lauf-zeit	Beschreibung
autosize	Boolean		Größe des Zeichenbereiches automatisch dem Bildinhalt anpassen

stretch	Boolean		Bilddarstellung automatisch der vorgegebenen Größe des Zeichenbereiches anpassen, dabei Bildinhalt entsprechend skalieren
grabbonly	Boolean		Bild vom Framegrabber in den linearen Bildpuffer hohlen, keine Darstellung
backpal	Boolean		Realisiert Palette als Hintergrundpalette
filterindex	Byte		Bei Verwendung eines Filters als Zuordnungs-index zu verwenden
FG32Eingang	AnsKinds		Eingangsbuchse des FG-32 Framegrabbers
Fg32Basisad	BadKinds		Basisadresse des FG-32 Framegrabbers einstellen
Image	TFG32Bitmap	x	Bildspeicherobjekt, kompatibel zu TBitmap
ImgTypes	ImgKinds	x	Gibt den Typ des derzeitig im linearen Bildpuffer befindlichen Bitmaps an
MulipleFg32	Boolean		True, falls mehrere FG-32 Framegrabber angesteuert werden sollen

Methoden von TFG32Image

Bezeichnung	Funktion
Greyimagedialog	Zeigt den Dialog zur Einstellung des FG-32 Treibers für Grauwert-Bilder

Colorimagedialog	Zeigt den Dialog zur Einstellung des FG-32 Treibers für Truecolor-Bilder
Loadbitmap(filename: String)	Lädt ein Windows Bitmap vom Filesystem. True Color Bitmaps werden in den linearen Bildpuffer übertragen
GrabbColorImage	Liest Farbbild vom Framegrabber
GrabbGreyImage	Liest Grauwertbild vom Framegrabber
Showabout	About Dialogbox

Ereignisse von TFG32Image

Bezeichnung	Funktion
onFilter	Das Ereignis onfilter tritt immer dann ein, wenn Daten aus dem linearen Puffer in die anzuzeigende Bitmap kopiert werden. Es ist so möglich, alle anzuzeigenden Daten entsprechend zu bearbeiten. Mit Hilfe des im Typ TFG32Filterevent definierten Bytewertes Filterindex kann eine Fallunterscheidung vorgenommen werden. Setzen Sie den Filterindex über die Eigenschaft von TFG32Image. Filterindex

Eigenschaften von TFG32Bitmap, nur zur Laufzeit, lesen

Name der Eigenschaft	Typ	Beschreibung
fgstruc	array[0..2] of FGRec; FGRec = Record res :Word; {Reserved} breite :Word; Hoehe:word; end;	Informationsstruktur des letzten digitalisierten Bildes als Feld für alle Farbkanäle.

linbuf	Linbuf: array[0..2] of THandle;	Linearer Bildpuffer zeilenweise von linker oberer Bildecke zu rechter unterer Bildecke. Index gilt in der Reihenfolge R-G-B (0-1-2.)
size	Longint;	Bestimmt die Größe eines Farbkanals in Bytes.

IV. Programmierung auf prozeduraler Ebene unter DOS

4.1. Programmierung in C
4.2. Microsoft C/C++ 7.0

In der Datei DOSMSC70.EXE sind folgende Dateien enthalten:

```
  Name          Original    Packed  Ratio   Date     Time   Attr  CRC
-------------  ---------  -------- ------ -------- -------- ---- ----
  DOSFG32.CFG        299        40  13.4% 93-07-01 02:00:00 a--w  6530
  DOSMSC70.C       21632      3261  15.1% 93-07-01 02:00:00 a--w  2759
  DOSMSC.EXE       47612     23375  49.1% 93-07-01 02:00:00 a--w  AD4E
  DOSMSC70.LIB     18332      7957  43.1% 93-07-01 02:00:00 a--w  88BD
  DOSMSC70.MAK       256        83  32.4% 93-07-01 02:00:00 a--w  61DF
-------------  ---------  -------- ------ -------- --------
     5 files       88246     34716  39.3% 93-07-01 02:00:00
```

Die Bibliothek DOSMSC70.LIB enthält folgende Funktionen:

void far pascal LIBMAIN (void);

Die Funktion initialisiert die Bibliothek und den FG-32. Wenn im aktuellen Verzeichnis eine DOSFG32.CFG Datei gefunden wird, werden die Werte dieser Datei übernommen. Wird keine Datei gefunden, gelten die im Treiber FG3xDRV voreingestellten Werte.

void far pascal SETUP (void)

Ein Setupmenue, funktionell äquivalent zum Programm FG32VGA wird gezeigt. Die Einstellungen können mit den [Pfeiltasten] erreicht und mit [Bild hoch] bzw. [Bild abwärts] modifiziert werden. Mit der [Esc]ape Taste wird das Menue verlassen bei

gleichzeitigem Speichern aller Werte in der Datei DOSFG32.CFG.

void far pascal **GREY320** (*pbuf*)

char far * *pbuf* - pointer auf einen Bildspeicher mit 320x240 Byte

Die Funktion dient der Digitalisierung von Grauwertbildern mit 320x240 Pixeln für Videoquellen im US-Standard.

void far pascal **GREY384** (*pbuf*)

char far * *pbuf* - pointer auf einen Bildspeicher mit 384x288 Byte

Die Funktion dient der Digitalisierung von Grauwertbildern mit 384x288 Pixel für Videoquellen im 50Hz-Standard.

void far pascal **GREY640** (*pbuf*)

char far * *pbuf* - pointer auf einen Bildspeicher mit 640x480 Byte

Die Funktion dient der Digitalisierung von Grauwertbildern mit 640x480 Pixeln für Videoquellen im US-Standard.

void far pascal **GREY768** (*pbuf*)

char far * *pbuf* - pointer auf einen Bildspeicher mit 768x576 Byte

Die Funktion dient der Digitalisierung von Grauwertbildern mit

768x576 Pixel für Videoquellen im 50Hz-Standard.

void far pascal **GREY320AV** (*pbuf, av*)

char far * *pbuf* - pointer auf einen Bildspeicher mit 320x240x2 Byte
int *av* - 2^{av}= Anzahl der gemittelten Bilder

Die Funktion dient der Digitalisierung von Grauwertbildern mit Averaging der Größe 320x240 Pixel für Videoquellen im US-Standard.

void far pascal **GREY384AV** (*pbuf* , *av*)

char far * *pbuf* - pointer auf einen Bildspeicher mit 384x288 Byte
int *av* - 2^{av}= Anzahl der gemittelten Bilder

Die Funktion dient der Digitalisierung von Grauwertbildern mit Averaging der Größe 384x288 Pixel für Videoquellen im 50Hz-Standard.

void far pascal **GREY640AV** (*pbuf, av*)

char far * *pbuf* - pointer auf einen Bildspeicher mit 640x480 + 65536Byte
int *av* - 2^{av}= Anzahl der gemittelten Bilder

Die Funktion dient der Digitalisierung von Grauwertbildern mit Averaging der Größe 640x480 Pixel für Videoquellen im US-Standard. Weil der erforderliche Speicher von 640x480x2 Bytes unter DOS normalerweise nicht verfügbar ist, werden 640 Kbyte

temporäre Dateien auf der Festplatte angelegt. Die Funktion wird damit erheblich verlangsamt.

Wenn XMS oder EMS Speicher zusätzlich verfügbar ist, sollte diese Funktion durch eine auf die konkrete Speichekonfiguration zugeschnittene Funktion ersetzt werden.
Nach Aufruf der Funktion kann der vom System bereitgestellte Systemspeicher auf 640x480 Byte reduziert werden.

void far pascal **GREY768AV** (*pbuf, av*)

char far * *pbuf* - pointer auf einen Bildspeicher mit 768x576 + 65536 Byte
int *av* - 2^{av}= Anzahl der gemittelten Bilder

Die Funktion dient der Digitalisierung von Grauwertbildern mit Averaging der Größe 768x576 Pixel für Videoquellen im 50Hz-Standard. Weil der erforderliche Speicher von 768x576x2 Bytes unter DOS normalerweise nicht verfügbar ist, werden 896 Kbyte temporäre Dateien auf der Festplatte angelegt. Die Funktion wird damit erheblich verlangsamt.
Wenn XMS oder EMS Speicher zusätzlich verfügbar ist, sollte diese Funktion durch eine auf die konkrete Speichekonfiguration zugeschnittene Funktion ersetzt werden.
Nach Aufruf der Funktion kann der vom System bereitgestellte Systemspeicher auf 768x576 Byte reduziert werden.

void far pascal **DISPGREY** (*pbuf, bits, xvga, yvga, ximg, yimg, xpos, ypos, xlen, ylen*)

char far * *pbuf* - pointer auf Bilddatenpuffer
int *bits* - Pixeltiefe des VGA Modes
int *xvga* - VGA x -Auflösung

int *yvga* - VGA y -Auflösung
int *ximg* - Bilddaten x -Auflösung
int *yimg* - Bilddaten y -Auflösung
int *xpos* - VGA x -Position
int *ypos* - VGA y -Position
int *xlen* - dargestellte x -Auflösung
int *ylen* - dargestellte y -Auflösung

Diese Funktion dient der Bilddarstellung von Grauwertbildern zu Testzwecken. Unterstützt werden 4- und 8-bit Modi von VGA Karten. Im 4-bit-Modus werden 16 Graustufen bei 640x480 Pixeln unterstützt. Bei vielen SVGA Karten funktioniert diese Funktion auch Problemlos bei Auflösungen von 800x600 Pixeln. Im 8-bit-Modus wird die Auflösung 320x200 unterstützt. Für Karten mit Tseng Labs ET4000 Controller und SVGA Karten mit gleichartiger Speicherbankumschaltung können auch Auflösungen von 800x600 und 1024x768 zum Einsatz kommen. Die VGA (x,y) Position sollte außerhalb der Standard VGA Modi (0,0) betragen. Bilder, die größer als die Bildschirmauflösung sind, lassen sich mit den Werten xlen und ylen beschneiden, um einen der VGA-Auflösung entsprechenden Ausschnitt darstellen zu können.

int far pascal **CHECKTSENG** (void)

Die Funktion liefert den Wert 0, wenn die Grafikkarte mit einem Tseng Labs ET4000 Controller bestückt ist.

void far pascal **SWITCHTSENG** (void)

Die Funktion schaltet eine Tseng Labs ET4000 VGA Karte in den Videomode 30H mit 800x600 Pixel bei 256 Farb- bzw. Grauwerten. Vor dem Einsatz dieser Funktion muß zum Schutz vor Zerstörung sichergestellt sein, dass der am System

angeschlossene Monitor diese Auflösung verträgt.

void far pascal **PALGREY16** (void)

Eine VGA-Palette mit 16 Grauwerten wird eingestellt.

void far pascal **PALGREY256** (void)

Eine VGA-Palette mit 256 Grauwerten wird eingestellt.

void far pascal **SETIMODE** (*imode*)

int *imode* - interlaced Mode 0 oder 1

Umschaltung auf alternative Halbbilderkennung. In hohen Auflösungen wird ein Bild aus zwei Halbbildern zusammengesetzt. Für die jeweilige Videoquelle sollte der Wert von imode ausprobiert werden, mit dem der geringste Zeilenversatz erreicht wird.

void far pascal **COLO320** (*pbuf*);

char far * *pbuf* - pointer auf einen Bildspeicher mit 320x240x3 Byte

Die Funktion dient der Digitalisierung von Farbbildern mit 320x240 24-bit-Pixeln für Videoquellen im US-Standard.

void far pascal **COLO384** (*pbuf*)

char far * *pbuf* - pointer auf einen Bildspeicher mit 384x288x3

Byte

Die Funktion dient der Digitalisierung von Farbbildern mit 384x288x24-bit-Pixeln für Videoquellen im 50Hz-Standard.

void far pascal **COLO640P1** (*pbuf*)

char far * *pbuf* - pointer auf einen Bildspeicher mit 640x120x3 Byte

Die Funktion dient der Digitalisierung von Farbbildern mit 640x480x24-bit-Pixeln für Videoquellen im US-Standard. Wegen des begrenzten Speichers unter DOS wird ein Bild zwar mit 640x480 Pixel digitalisiert, es wird jedoch nur 1/4 der Bilddaten (640x120 Pixel) in den Hauptspeicher übertragen. Die drei verbleibenden Viertel lassen sich mit der Funktion COLO640P2 übertragen.

void far pascal **COLO640P2** (*pbuf*)

char far * *pbuf* - pointer auf einen Bildspeicher mit 640x120x3 Byte

Überträgt weitere Bilddaten der Funktion COLO640P1. Normalerweise wird ein Bild der Größe 640x480x24bit durch einmaliges rufen der Funktion COLO640P1 und durch dreimaliges rufen dieser Funktion erhalten.

void far pascal **COLO768P1** (*pbuf*)

char far * *pbuf* - pointer auf einen Bildspeicher mit 768x144x3 Byte

Die Funktion dient der Digitalisierung von Farbbildern mit 768x576 24-bit-Pixeln für Videoquellen im 50Hz-Standard.
Wegen des begrenzten Speichers unter DOS wird ein Bild zwar mit 768x576 Pixeln digitalisiert, es wird jedoch nur 1/4 der Bilddaten (768x144 Pixel) in den Hauptspeicher übertragen. Die drei verbleibenden Viertel lassen sich mit der Funktion COLO768P2 übertragen.

void far pascal **COLO768P2** (*pbuf*)

char far * *pbuf* - pointer auf einen Bildspeicher mit 768x144x3 Byte

Überträgt weitere Bilddaten der Funktion COLO768P1. Normalerweise wird ein Bild der Größe 768x576x24bit durch einmaliges rufen der Funktion COLO768P1 und durch dreimaliges rufen dieser Funktion erhalten.

void far pascal **DISPCOLO** (*pbuf, bits, xvga, yvga, ximg, yimg, xpos, ypos, xlen, ylen*)

char far * *pbuf*	-	pointer auf Bilddatenpuffer
int *bits*	-	Pixeltiefe des VGA Modes
int *xvga*	-	VGA x -Auflösung
int *yvga*	-	VGA y -Auflösung
int *ximg*	-	Bilddaten x -Auflösung
int *yimg*	-	Bilddaten y -Auflösung
int *xpos*	-	VGA x -Position
int *ypos*	-	VGA y -Position
int *xlen*	-	dargestellte x -Auflösung
int *ylen*	-	dargestellte y -Auflösung

Diese Funktion dient der Grobdarstellung von Farbbildern zu

Testzwecken. Unterstützt werden 4- und 8-bit Modi von VGA Karten. Im 4-bit-Modus werden 16 Graustufen bei 640x480 Pixeln unterstützt. Bei vielen SVGA Karten funktioniert diese Funktion auch Problemlos bei Auflösungen von 800x600 Pixeln. Im 8-bit-Modus wird die Auflösung 320x200 unterstützt. Für Karten mit Tseng Labs ET4000 Controller und SVGA Karten mit gleichartiger Speicherbankumschaltung können auch Auflösungen von 800x600 und 1024x768 zum Einsatz kommen. Die VGA (x,y) Position sollte außerhalb der Standard VGA Modi (0,0) betragen. Bilder, die größer als die Bildschirmauflösung sind, lassen sich mit den Werten xlen und ylen beschneiden, um einen der VGA-Auflösung entsprechenden Ausschnitt darstellen zu können. Diese Funktion genügt nur Testzwecken, weil Farbbilder auf Grafikkarten mit 256 Farben in tragbarer Qualität nur in Kombination mit Ditheringverfahren und Verfahren zur Optimierung der Farbpalette realisierbar sind.
Die Bilddaten für diese Funktion müssen im 24-bit RGB Format vorliegen. Bei VGA Auflösungen mit 16 Farben werden 2+1+1 bit für die Farben Rot, Grün und Blau dargestellt. Bei VGA Auflösungen von 256 Farben werden 3+3+2 bit für die Farben Rot, Grün und Blau dargestellt.

void far pascal **PALCOLO16** (*void*)

Für VGA Auflösungen mit 16 Farben erzeugt diese Funktion eine Farbpalette, die zu Testzwecken im Zusammenhang mit der Funktion DISPCOLO als erste grobe Näherung einsetzbar ist.

void far pascal **PALCOLO256** (*void*)

Realisiert eine RGB Farbpalette mit der Farbaufteilung 3+3+2 bit.

void far pascal **ONLINEGREY** (*void*)

Diese Funktion ermöglicht eine bewegte Darstellung der Videoquelle mit 256 Grauwerten. Ein spezieller Modus des FG-32 kommt zum Einsatz, bei dem die Bildauflösung durch die Hardware 2:1 untersetzt wird. Auf schnellen Rechnern werden Bildraten von 25 Bildern/s erreicht. Mit den Pfeiltasten kann das sichtbare Fenster von 320x200 Pixeln im Grundraster 384x288 bewegt werden. Mit der Leertaste kann die Funktion abgebrochen werden, um beispielsweise mit nahtlosem Übergang ein Bild höherer Auflösung zu Digitalisieren.

void far pascal **ONLINECOLOR** (*void*)

Diese Funktion entspricht der Funktion ONLINEGREY mit dem Unterschied, dass eine Farbdarstellung erfolgt. Die Farbdarstellung basiert auf 8-bit-Daten je Pixel mit 4 bit Helligkeitsinformation 2 bit Grün-Blau Differenz und 2 bit Grün-Rot Differenz. Mit dieser geringen Farbtiefe ist nur eine sehr schlechte Farbqualität erreichbar. deshalb sollte diese Funktion nur zu Zwecken der bewegten Darstellung der Videoquelle auf Grafikkarten mit auf beschränkter Farbtiefe eingesetzt werden.

void far pascal **ONLINEPAL** (*r, g, b*);
int *r*
int *g*
int *b*
Eine geeignete Farbpalette für die Funktion ONLINECOLOR wird durch diese Funktion generiert. Die Parameter für die Farben Rot, Grün und Blau geben die Farbintensität des jeweiligen Farbkanals an. Der Wert 100 entspricht bei jedem Farbkanal der Normaleinstellung

4.3. Borland C++ 3.1, 4.0, 4.5

In der Datei DOSBLC31.EXE sind folgende Dateien enthalten:

Name	Original	Packed	Ratio	Date	Time	Attr	CRC
DOSBLC.EXE	39824	17899	44.9%	93-07-01	02:00:00	a--w	22A4
DOSBLC31.C	21395	3397	15.9%	93-07-01	02:00:00	a--w	5342
DOSBLC31.DSK	581	286	49.2%	93-07-01	02:00:00	a--w	638B
DOSBLC31.LIB	18447	7957	43.1%	93-07-01	02:00:00	a--w	88BD
DOSBLC31.PRJ	5208	1136	21.8%	93-07-01	02:00:00	a--w	9809
DOSFG32.CFG	299	42	14.0%	93-07-01	02:00:00	a--w	3660
6 files	85754	30717	35.8%	93-07-01	02:00:00		

Funktionell entsprechen die Prozeduren in der Bibliothek DOSBLC31.LIB der Beschreibung des Kapitels 2.1.1. Für das Demoprogramm wurde aus Gründen der Übersichtlichkeit auf die Anwendung von BGI Treiberfunktionen verzichtet.

4.4. Programmierung in Basic

4.4.1. Microsoft Quick Basic 4.5

Leider sind die deutsche und die englische Version von Quick Basic 4.5 bezüglich des Formates von Bibliotheken nicht miteinander kompatibel. Deshalb befinden sich zwei funktionell äquivalente Versionen für Quick Basic auf der Diskette.

Die englische Version ist in der Datei DOSQB45E.EXE enthalten:

```
  Name          Original    Packed  Ratio   Date     Time   Attr CRC
--------------  --------  -------- ------ -------- -------- ---- ----
  DOSFG32.CFG        299        40  13.4% 93-07-01 02:00:00 a--w 6530
  DOSQB45.BAS      20282      3322  16.4% 93-07-01 02:00:00 a--w 5385
  DOSQB45.EXE      25866     11448  44.3% 93-07-01 02:00:00 a--w 6A9D
  DOSQB45E.LIB     18447      8043  43.6% 93-07-01 02:00:00 a--w 2299
  DOSQB45E.QLB     23498      9768  41.6% 93-07-01 02:00:00 a--w 855E
  RUN.BAT             36        36 100.0% 93-07-01 02:00:00 a--w 24BE
--------------  --------  -------- ------ -------- --------
     6 files       88428     32657  36.9% 93-07-01 02:00:00
```

Die deutsche Version ist in der Datei DOSQB45D.EXE enthalten:

```
  Name          Original    Packed  Ratio   Date     Time   Attr CRC
--------------  --------  -------- ------ -------- -------- ---- ----
  DOSFG32.CFG        299        40  13.4% 93-07-01 02:00:00 a--w 6530
  DOSQB45.BAS      20282      3322  16.4% 93-07-01 02:00:00 a--w 5385
  DOSQB45.EXE      66240     39283  59.3% 93-07-01 02:00:00 a--w 5AEB
  RUN.BAT             36        36 100.0% 93-07-01 02:00:00 a--w E87F
  DOSQB45D.QLB     23498      9770  41.6% 93-07-01 02:00:00 a--w 9530
  DOSQB45D.LIB     18447      8043  43.6% 93-07-01 02:00:00 a--w 139A
--------------  --------  -------- ------ -------- --------
     6 files      128802     60494  47.0% 93-07-01 02:00:00
```

Der Basicquelltext beider Versionen ist identisch. Die Datei DOSQB45.EXE ist in der deutschen Version als eigenständige EXE-Datei übersetzt, während die englische Version die Basic Runtime BRUN45.EXE erfordert. Die Bibliothek DOSQB45x.QLB wird zur Arbeit in der Quick Basic Umgebung benötigt, die mit den in RUN.BAT gesetzten
Optionen geladen wird. Für die Kommandozeilenversion ist die Bibliothek DOSQB45x.LIB erforderlich. Die Funktionen der QLB

und LIB Dateien sind identisch.

Für folgende Funktionen gilt die Beschreibung der namensgleichen C-Prozeduren im Abschnitt 2.1.1.:

DECLARE SUB **LIBMAIN** ()
DECLARE SUB **SETUP** ()
DECLARE SUB **SETIMODE** (BYVAL imode AS INTEGER)
DECLARE SUB **CHECKTSENG** (SEG sm AS INTEGER)
DECLARE SUB **SWITCHTSENG** ()
DECLARE SUB **GREY320** (SEG buf AS INTEGER)
DECLARE SUB **GREY384** (SEG buf AS INTEGER)
DECLARE SUB **GREY640** (SEG buf AS INTEGER)
DECLARE SUB **GREY768** (SEG buf AS INTEGER)
DECLARE SUB **GREY320AV** (SEG buf AS INTEGER,
BYVAL av AS INTEGER)
DECLARE SUB **GREY384AV** (SEG buf AS INTEGER,
BYVAL av AS INTEGER)
DECLARE SUB **GREY640AV** (SEG buf AS INTEGER,
BYVAL av AS INTEGER)
DECLARE SUB **GREY768AV** (SEG buf AS INTEGER,
BYVAL av AS INTEGER)
DECLARE SUB **COLO320** (SEG buf AS INTEGER)
DECLARE SUB **COLO384** (SEG buf AS INTEGER)
DECLARE SUB **COLO640P1** (SEG buf AS INTEGER)
DECLARE SUB **COLO640P2** (SEG buf AS INTEGER)
DECLARE SUB **COLO768P**1 (SEG buf AS INTEGER)
DECLARE SUB COLO768P2 (SEG buf AS INTEGER)
DECLARE SUB DISPGREY (SEG buf AS INTEGER,
BYVAL bits AS INTEGER, BYVAL xvga AS BINTEGER,BY-
VAL yvga AS INTEGER, BYVAL ximg AS INTEGER,
BYVAL yimg AS INTEGER, BYVAL xpos AS INTEGER,
BYVAL ypos AS INTEGER, BYVAL xlen AS INTEGER,

```
    BYVAL ylen AS INTEGER)

DECLARE SUB DISPCOLO (SEG buf AS INTEGER,
    BYVAL bits AS INTEGER, BYVAL xvga AS INTEGER,
    BYVAL yvga AS INTEGER, BYVAL ximg AS INTEGER,
    BYVAL yimg AS INTEGER, BYVAL xpos AS INTEGER,
    BYVAL ypos AS INTEGER, BYVAL xlen AS INTEGER,
    BYVAL ylen AS INTEGER)
DECLARE SUB PALGREY16 ()
DECLARE SUB PALGREY256 ()
DECLARE SUB PALCOLO16 ()
DECLARE SUB PALCOLO256 ()
DECLARE SUB ONLINEGREY ()
DECLARE SUB ONLINECOLOR ()
DECLARE SUB ONLINEPAL (BYVAL r AS INTEGER,
    BYVAL g AS INTEGER, BYVAL b AS INTEGER)


DECLARE SUB VGATEXT ()
```

Diese Funktion schaltet die VGA-Karte über BIOS-Funktionen in den Textmodus.

```
DECLARE SUB VGACGA ()
```

Diese Funktion schaltet die VGA Karte über BIOS Funktionen in die Grafikauflösung 320x200 mit 256 Farben.

```
DECLARE SUB VGAVGA ()
```

Diese Funktion schaltet die VGA Karte über BIOS Funktionen in die Grafikauflösung 640x480 mit 16 Farben.

4.5. Programmierung in Pascal

4.5.1. Borland Pascal 7.0

In der Datei DOSPAS70.EXE sind folgende Dateien enthalten:

```
  Name          Original    Packed  Ratio   Date     Time   Attr CRC
--------------  --------  -------- ------ -------- -------- ---- ----
  DOSFG32.CFG        299        40  13.4% 93-07-01 02:00:00 a--w 6530
  DOSLIBTP.OBJ     17015      7402  43.5% 93-07-01 02:00:00 a--w 0196
  DOSPAS.EXE       24480     10246  41.9% 93-07-01 02:00:00 a--w A5D4
  DOSPAS.PAS       18757      3124  16.7% 93-07-01 02:00:00 a--w 1C3F
--------------  --------  -------- ------ -------- --------
     4 files       60551     20812  34.4% 93-07-01 02:00:00
```

Die Datei FG31SVHS enthält eine geänderte Version für die Einschaltung der Mini-DIN Buchse des FG-31.
Nachfolgende Prozeduren sind im Abschnitt 2.1.1. unter den namensgleichen C-Funktionen beschrieben:

```
procedure LIBMAIN ( x: Integer ); far;
procedure SETUP   ( x: Integer ); far;
procedure SETIMODE (imode : Integer);far;
function CHECKTSENG (x: Integer) : Integer; far;
procedure SWITCHTSENG ( x: Integer); far;
procedure GREY320 (buf: pointer ); far;
procedure GREY384 (buf: pointer ); far;
procedure GREY640 (buf: pointer); far;
procedure GREY768 (buf: pointer); far;
procedure GREY320AV (buf: pointer; av: Integer); far;
procedure GREY384AV (buf: pointer; av: Integer); far;
procedure GREY640AV (buf: pointer; av: Integer); far;
procedure GREY768AV (buf: pointer; av: Integer); far;
procedure COLO320 ( buf: pointer); far;
procedure COLO384 (buf: pointer); far;
procedure COLO640P1 (buf: pointer); far;
procedure COLO640P2 (buf: pointer); far;
procedure COLO768P1 (buf: pointer); far;
```

procedure **COLO768P2** (buf: pointer); far;
procedure **DISPGREY** (buf: pointer; bits: Integer;
xvga: Integer; yvga: Integer; ximg: Integer;
yimg: Integer; xpos: Integer; ypos: Integer;
xlen: Integer; ylen: Integer); far;
procedure **DISPCOLO** (buf: pointer; bits: Integer;
xvga: Integer; yvga: Integer; ximg: Integer;
yimg: Integer; xpos: Integer; ypos: Integer;
xlen: Integer; ylen: Integer); far;
procedure **PALGREY16** (x: Integer); far;
procedure **PALGREY256** (x: Integer);far;
procedure **PALCOLO16** (x: Integer); far;
procedure **PALCOLO256** (x: Integer);far;
procedure **ONLINEGREY** (x: Integer); far;
procedure **ONLINECOLOR** (x: Integer); far;
procedure **ONLINEPAL** (r:Integer; g:Integer; b:Integer); far;

Folgende Funktionen benutzen BIOS-Interrupts um den Bildschirmmode umzuschalten:

procedure **SWITCHVGA** (x: Integer);far;
schaltet die VGA-Karte auf 640x480 Pixel mit 16 Farben.

procedure **SWITCHCGA** (x: Integer);far;
schaltet die VGA-Karte auf 320x200 Pixel mit 256 Farben.

procedure **SWITCHTEXT** (x: Integer);far;
schaltet die VGA Karte in den Textmode.

Leider benutzt Turbo Pascal eine eigene Initialisierung der VGA Karte. Wenn die Grafikauflösung über BIOS Interrupts umgeschaltet wird, sind die Turbo-Pascal Funktionen zum Bildschirm löschen und WriteLn nicht mehr einsetzbar. Zur Abhilfe dienen

nachfolgende Prozeduren der Bibliothek:

procedure **VGACLS** (x: Integer);far;
Bildschirm löschen 640x480

procedure **CGACLS** (x: Integer);far;

Bildschirm löschen 320x200

procedure **VGATEXTCLS** (x: Integer);far;
Bildschirm löschen Textmode.

procedure **WriteVgaLn** (x: string; attr: Integer);far;
Die Funktion entspricht WriteLn, die Integer Variable übergibt das Farbattribut für den darzustellenden String.

procedure **VgaInteger** (y:Integer;x:Integer;num:Integer);far;
Ausgabe eines dreistelligen Integer Wertes num (Bereich 0...999) auf die Bildschirmposition (x,y)

V.
Low level Programmierung

Als Low- Level- Programmierung werden im Zusammenhang mit den Framegrabber APIs direkte Aufrufe der Device- Driver bezeichnet. Für WinMe/98/95/31 und Dos realisiert der Device-Driver FG3xDRV.EXE (x=0...5 entsprächend Framegrabbertyp) eine Schnittstelle, die mit dem Software Interrupt 60H aufrufbar ist. Für WinMe/98/95/31 und Dos gibt es keine Daten- Trasfer-Funktionen im Device- Driver, weil die Daten direkt aus Portadressen als sequentieller Datenstrom auslesbar sind. Unter WinXP/2000/NT gibt es solche Funktionen, die im Abschnitt 1.1.2. beschrieben sind.

5.1. Aufbau eines Funktionsaufrufs

Der Treiber FG3xDRV.EXE belegt nach dem Aufruf das Softwareinterrupt 60H .
Die Grundstruktur für einen Funktionsaufruf sieht in einem 80x86 Assembler folgendermaßen aus:

```
mov     ax, 9709h       ;9209h bei FG30 WinMe/98/95/3.x/Dos
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,9709h  ;9209h bei FG30 WinMe/98/95/3.x/Dos
```

wird das Register AX mit der hexadezimalen Zahl 9709 geladen. Durch die Zahl 9709h wird FG3xDRV 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.

In anderen Betriebssystemen wird der int 60h- Aufruf durch einen Treiberaufruf ersetzt, bei dem die Prozessor- Registerinhalte in Form von Variablen zum Treiber gesendet und vom Treiber empfangen werden. Abschnitt 1.1.2 beschreibt den Aufruf für WinXP/2000/NT und zusätzlich Daten- Transfer- Befehle.

5.1.1. Tabellarische Übersicht

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

N - Windows XP/ 2000/NT und Linux
O - OS/2
W - Windows Me/ 9x/ 3.xx
D - DOS

H - “History”, Funktion nicht für Neuentwicklungen verwenden

ax=9709h			Eingabe		Ausgabe	
bx	**Funktionsname**	**Umg.**	**cx**	**dx**	**cx**	**dx**
00	DrvInit	NOWD	-	-	9709	vers
01	DrvPutClientX	WO	topx	lenx	-	-
02	DrvPutClientY	WO	topy	leny	-	-
03	DrvGetClientX	WO	-	-	topx	lenx
04	DrvGetClientY	WO	-	-	topy	leny
05	DrvSetAdjustment	NOWD	cl:sat ch:cont	dl:brig	-	-
06	DrvGetAdjustment	NOWD	-	-	cl:sat ch:cont	dl:brig
07	DrvSetXRAM	NOWD	index	data	-	-
08	DrvIniXRAM	NOWD	-	-	-	-
09	DrvSetBase	NOWD	basis	dwn	-	-
10	DrvSetXYoffs	NOWD	xoffs	yoffs	-	-
11	DrvSetGain	NOWD	gain	offset	-	-
12	DrvGetGain	NOWD	-	-	gain	offset
13	DrvSwitchGrabber	N	index		type 0-2	base
14	DrvDefineGrabber	N	typ,index	basis	present	-
15	DrvSetDevCapsXY	H	xres	yres	-	-
16	DrvGetExRAM	NWD	index	-	-	data
17	DrvSetExRAM	NWD	index	data	-	-
18	DrvIniExRAM	NWD	-	-	-	-
19	DrvPeekExRAM	NWD	index	-	-	data
20	DrvPokeExRAM	NWD	index	data	-	-
21	Reserved	-	-	-	-	-
22	DrvGetPal	H	-	-	-	-
23	DrvPutPal	H	offs	-	-	-
24	DrvSetPalGrey	H	-	-	-	-
25	DrvSetPixBits	H	bits	-	-	-
26	DrvGetPixBits	H	-	-	bits	-
27	DrvOnlineRpt	H	-	-	status	-
28	DrvAcqColDiff	H	xres	yres	status	-
29	DrvSetColOffs	H	cxoffs	cyoffs	-	-

ax=9709h		Umge-	Eingabe		Ausgabe		
bx	Funktionsname	bung	cx	dx	cx	dx	
30	DrvGetXRAM	NOWD	index	-	-	data	
31	DrvGetColOffs	NOWD	-	-	cxoffs	cyoffs	
32	DrvAcqColDiff2	H	xres	yres	status	-	50Hz
33	DrvAcqColDiff4	H	xres/2	yres	status	-	50Hz
34	DrvSetPalCol8	H	R+G	Blue	-	-	
35	DrvAcqColDiff60	H	xres	yres	status	-	60Hz
36	DrvAcqColDiff260	H	xres	yres	status	-	60Hz
37	DrvSwitchInput	NOWD	input	-	-	-	
38	DrvGetSwSet	NOWD	input	-	swset	-	
39	DrvSetSwSet	NOWD	input	swset	-	-	
40	DrvGetInput	NOWD	-	-	input	-	
41	DrvGetBasis	NOWD	-	-	basis	-	
42	DrvGetWaits	H	-	-	waits	-	
43	DrvEingCpy	H	src	dst	-	-	
44	DrvGetCardType	NOWD			colflag	ycflag	
45	DrvAcqGreyBig	H	-	-	status	-	
46	DrvAcqGreyBig60	H	-	-	status	-	
47	DrvAcqGreySmall60	H	-	-	status	-	
48	DrvUserOutput	H	bits	-	-	-	
49	DrvSetFrameType	NOWD	frtype	-	-	-	
50	FlmSetSize	NOWD	fxsize	fysize	-	-	
51	FlmSetTopLeft	NOWD	xpos	ypos	-	-	
52	FlmSetStatus	NOWD	xstatus	ystatus	-	-	
53	FlmFirstFrame	NOWD	-	-	status	-	
54	FlmNextFrame	NOWD	-	-	status	-	
55	FlmIniNextFrame	NOWD	-	-	-	-	
56	FlmWaitNextFrame	NOWD	-	-	status	-	
57	FlmGetStatus	NOWD	maske	-	status	-	
58	FlmReadBlind	NOWD	count	basisoffs	-	-	
60	FlmAcq	NOWD	-	-	-	basis	
61	FlmWait	NOWD	-	-	-	basis	
62	FlmReRead	NOWD	-	-	-	basis	
63	RealModeRead	D	CDwords	auf dx:di	-	-	

64	ReadDword	NOWD	-	-	loword	hiword
ax=9709h		Umge-	Eingabe		Ausgabe	
bx	Funktionsname	bung	cx	dx	cx	dx
80	DrvVgaDispCga	DH	-	-	-	-
81	DrvVgaIniXRAM	DH	-	-	-	-
82	DrvVgaAcq768H	DH	-	-	status	-
83	DrvVgaAcq768	DH	-	-	status	-
84	DrvVgaAcq384	DH	-	-	status	-
85	DrvVgaGetOffs	DH	-	-	offsx	offsy
86	DrvVgaSetOffs	DH	offsx	offsy	-	-
87	DrvVgaDispCgaColor	H	-	-	-	-
88	DrvVgaColImages	H	imagebuf	-	status	-
89	DrvVgaGetDither	H	-	-	dflag	-
90	DrvVgaSetDither	H	ditherflag	-	-	-
91	DrvVgaGetShift	H	-	-	sixpos	siypos
92	DrvVgaSetColPal	H	-	-	-	-
93	DrvVgaSaveBmp	H	hwndfile	segment	-	-
100	DrvHcDisp	H	-	-	status	-
101	DrvHcIniXRAM	H	-	-	-	-
102	DrvHcSetXRAM	H	segm	offs	-	-
103	DrvHcGetXRAM	H	segm	offs	-	-
104	DrvHcSetScreenParm	H	segm	offs	-	-
105	DrvHcGetScreenParm	H	segm	-	offs	-
106	DrvHcSetOffs	H	colxoffs	colyoffs	-	-
107	DrvHcGetOffs	H	-	-	colxoffs	colyoffs
108	DrvHcGetParm123	H	-	-	seg	offs
109	DrvHcSetImgType	H	frtype	size	-	-
110	DrvHcGetImgType	H	-	-	frtype	size
111	DrvHcDispBig	H	segment	offset	status	-
112	DrvHcDispSmall	H	segment	offset	status	-
113	DrvHcRedispBig	H	segment	offset	status	-
114	DrvHcSetPosBig	H	bigshiftx	bigshifty	status	-
115	DrvSetIMode	NOWD	mode	-	-	-
116	DrvGetIMode	NOWD	-	-	mode	-

5.1.2. Beschreibung der Treiberfunktionen

Funktion 0 ab Version 2.32
Funktionsname: DrvInit

Eingabe:	ax	Treiberkennung (MASM: 9709H Basic &H9709 Pascal: $9709 C: 0x9709)
Rückgabe:	dx	Treiberversion * 100h

Diese Funktion sollte jedes Programm sofort nach Programmstart rufen.

Funktion 1
Funktionsname: DrvPutClientX

Eingabe:	cx	X - Koordinate der Fensterposition der linken oberen Ecke
	dx	X - Länge des Fensters
Rückgabe:		keine

Die X - Lage eines MS-Windows oder OS/2 Fensters wird durch diese Funktion dem Treiber mitgeteilt.

Funktion 2
Funktionsname: DrvPutClientY

Eingabe:	cx	Y - Koordinate der Fensterposition der linken oberen Ecke
	dx	Y - Ausdehnung des Fensters
Rückgabe:		keine

Die Y - Lage eines MS - Windows oder OS/2 Fensters wird durch

diese Funktion dem Treiber mitgeteilt.

Funktion 3
Funktionsname: DrvGetClientX

Eingabe:		keine
Rückgabe:	cx	Rückgabe der Fensterposition X der linken oberen Ecke des zuletzt an den Treiber übermittelten Fensters
	dx	Rückgabe der X - Ausdehnung des Fensters

Die X - Koordinaten eines MS - Windows oder OS/2 Fensters, die dem Treiber zuletzt mtgeteilt wurden, lassen sich mit dieser Funktion abfragen.

Funktion 4
Funktionsname: DrvGetClientY

Eingabe:		keine
Rückgabe:	cx	Rückgabe der Fensterposition Y der linken oberen Ecke
	dx	Rückgabe der Y - Fensterausdehnung

Die Y Koordinaten eines MS - Windows oder OS/2 Fensters, die dem Treiber zuletzt mitgeteilt wurden, lassen sich mit dieser Funktion abfragen.

Funktion 5
Funktionsname: DrvSetAdjustments

Eingabe:	cl=Sättigung, ch=Kontrast, dl=Helligkeit
Rückgabe:	keine

Der Wertebereich ist 0 bis 255 für Helligkeit, 0 bis 127 für Kontrast (außerhalb dieses Bereiches: inverse Helligkeit), 0 bis 63 für Sättigung (auserhalb des Bereiches Vorzeichenänderungen der Farbkanäle U und V).
Mit Funktion 8 werden die Einstellungen aktiv.

Funktion 6
Funktionsname: DrvGetAdjustments

Eingabe: keine
Rückgabe: cl=Sättigung, ch=Kontrast, dl= Helligkeit

Der Wertebereich ist äquivalent zu Funktion 5.

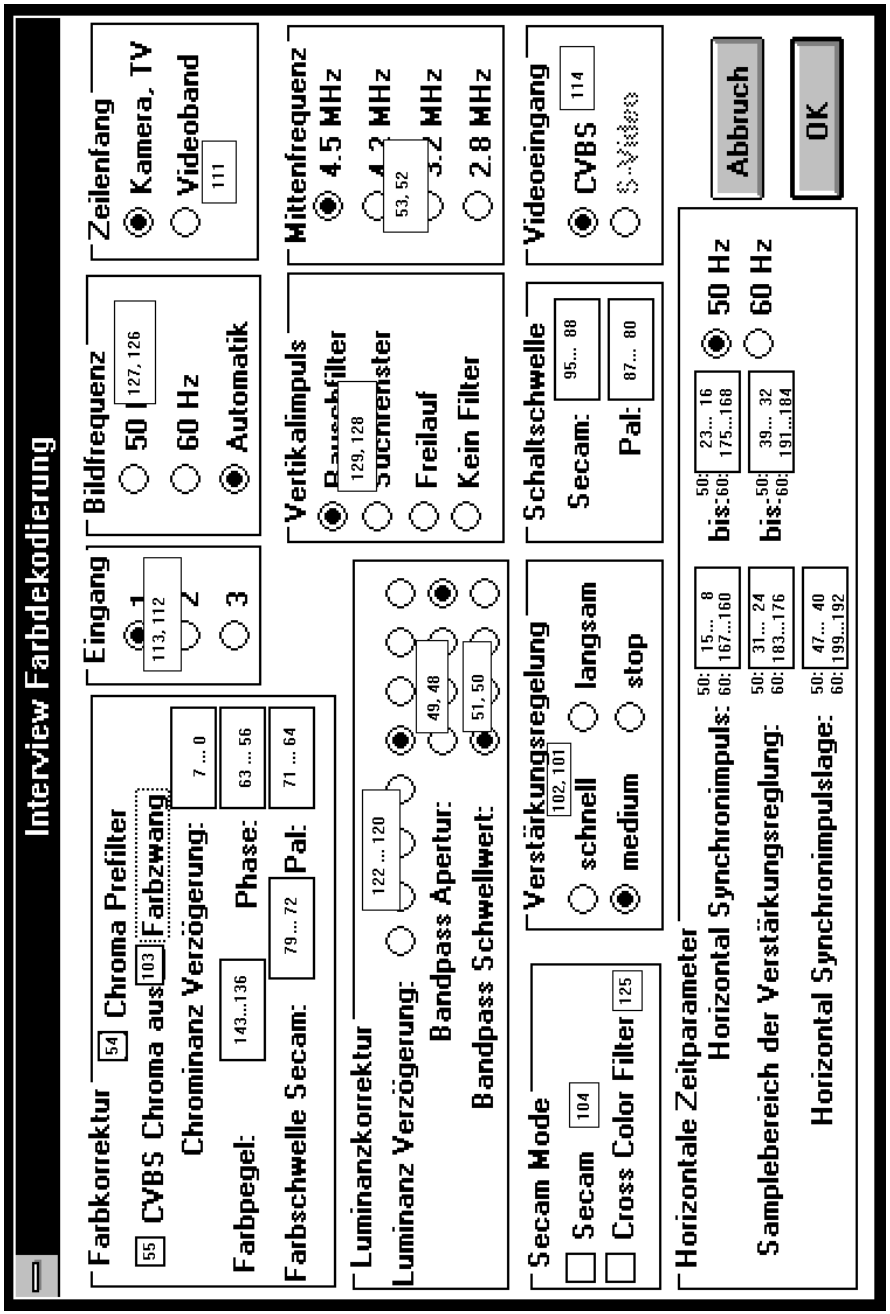
Funktion 7
Funktionsname: DrvSetXRAM

Eingabe: cx XRAM Adresse
dl XRAM Daten
Rückgabe: keine

Mit dieser Funktion wird ein Zwischenspeicher mit Parametern für die Farbdekodierung beschrieben. Der Speicher hat eine Länge von 200 bits, die mit dieser Funktion byteweise geändert werden können. Die Adresse eines bits bestimmt sich aus:

Bitposition in dl + 8 * cx

Die folgende Abbildung zeigt die Dialogbox zur Farbdekodierung mit Eintragungen zur Bitbelegung der Elemente.
Siehe auch Funktion 30.



Funktion 8
Funktionsname: DrvIniXRAM

Eingabe: keine
Rückgabe: keine

Die Funktion aktiviert alle Änderungen die im XRAM z.B. mit der Funktion 7 vorgenommen wurden.

Funktion 9
Funktionsname: DrvSetBasis

Eingabe: cx Basisadresse des FG-3x
dx dwn
Rückgabe: keine

Gibt die Basisadresse vor, auf der ein FG-3x angesprochen werden soll. Wenn mehrere Karten im System laufen, kann mit dieser Funktion zwischen den Karten umgeschaltet werden.
In den Betriebssystemen Windows XP/ 2000 und NT wird außerdem der Wert dwn ausgewertet. Der Paramater **dwn=1** bewirkt, dass eine Neuinitialisierung der Karte **nicht** erfolgen soll. Damit ist die Umschaltung zwischen den Karten schneller möglich. Für den Fall, dass die angesprochene Karte eine PC-Card ist, kann diese Funktion mit dwn =0 benutzt werden, um eine zur Laufzeit des Programms eingesteckte PC-Card zu initialisieren. Diese Initialisierung sollte wenigstens 1x mit Programmstart erfolgen, damit eine später eingesteckte PC-Card mit Neustart des Programms initialisiert werden kann.

Funktion 10
Funktionsname: DrvSetXYoffs

Eingabe: cx X - offset der Bildlage
dx Y - offset der Bildlage
Rückgabe: keine

Über diese Funktion lässt sich die Bildlage exakt einstellen. Die Offsetwerte beziehen sich auf die Bild- und Zeilensynchronimpulse. Für die korrekte Funktion der Digitalisierung muss sichergestellt sein, dass die Anzahl der Zeilen bzw. die Anzahl der Pixel je Zeile, die im Hardwarefenster abgefordert werden auch tatsächlich im Bild enthalten sind. Die Digitalisierung einer Zeile muss also vor dem Horizontalen Synchronimpuls, die Digitalisierung der letzten Zeile eines Bildes vor dem vertikalen Bildsynchronimpuls abgeschlossen sein. Deshalb ist es günstig für unbekannte Videoquellen zunächst niedrige Offsetwerte einzustellen, etwa cx=80, dx=5 . Nach Erhalt stabiler Bilder sollten dann die Werte langsam bis zur korrekten Bildposition erhöht werden.
Solange die genannten Bedingungen eingehalten werden, kann für Bildausschnitte die Lage der Offsetwerte auch beliebig im Bild liegen. Die mitgelieferte Software verwendet mit Ausnahme der Auflösung 592x442 (FG32CLIP, ET4HICOL) keine Bildausschnitte.

Funktion 11
Funktionsname: DrvSetGain

Eingabe: cx gain
dx offset
Rückgabe: keine

Der Wertebereich ist 0...255.

Funktion 12
Funktionsname: DrvGetGain

Eingabe: keine
Rückgabe: cx gain
dx offset

Der Wertebereich ist 0...255.

Funktion 13
Funktionsname: DrvSwitchGrabber

Eingabe: cx index [0..7]
Rückgabe: cx typ (0=FG30,1=FG31,2=FG32, 3=FG33, 4=FG34)
dx baseaddr

Unter Windows XP/ 2000/ NT gibt es diese Funktion, um sehr schnell zwischen mehreren Karten umzuschalten. Die Reihenfolge der Karten wurde mit Funktion 14 festgelegt oder entspricht der Treibervoreinstellung:

Karte	index	Typ	Basisadresse
1	0	FG-32	300H
2	1	FG-30	310H
3	2	FG-31	318H
4	3	FG-32	700H ,dann 4 freie Einträge

Um die Kompatibilität mit bisheriger Software zu erhalten, muss diese Funktion für den Betrieb nur einer Karte nicht verwendet werden. In der Initialisierungsphase wird durch DrvSetBasis bewirkt, dass mit den Adressen 300H,310H,318H und 700H die oben genannten Kartentypen assoziiert werden.

Funktion 14
Funktionsname: DrvDefineGrabber

Eingabe: cl typ (0=FG30,1=FG31,2=FG32

3=FG33, 4=FG34)
ch index [0...7]
dx baseaddr
Rückgabe: cx present

Mit dieser Funktion lassen sich die verwendeten Kartentypen festlegen. Diese Funktion ist nur erforderlich, wenn die in Funktion 13 beschriebenen Voreinstellungen geändert werden müssen. Die Rückgabe mit cx=0 zeigt an, dass die Karte vorhanden ist. Vor Ausführung sollte eine Karte mit anderem index durch Funktion 13 aktiviert werden.

Funktion 15
Funktionsname: DrvSetDevCapsXY

Eingabe: cx x Auflösung des aktuellen Videomodes
dx y Auflösung des aktuellen Videomodes
Rückgabe: keine

Diese Funktion ist nur für den Fall von Interesse, dass ein Fenster im Bildspeicher der Grafikkarte beschrieben werden soll. In diesem Fall ist die X-Auflösung besonders wichtig für die korrekte Funktion der Darstellungsroutinen. Einige MS-Windows Treiber für die Auflösung 800x600 mit 256 Farben verwenden 1024 Pixel pro Zeile und beschneiden das Bild nur in der Darstellung. In diesem Fall muß der Wert cx=1024 übergeben werden.

Funktion 16
Funktionsname: DrvGetExRAM

Eingabe: cx=offset (0...255) und 0FFFFH
Rückgabe: dx=8-bit-Ergebnis

Liest den status der durch Funktion 17 änderbaren Werte. Offset

0FFFFH bewirkt das Auslesen aus der Hardware und Zwischenpufferung im Treiber. Mit Offsets von 0 bis 255 sind die 8-Bit-Werte des Puffers lesbar.

Funktion 17
Funktionsname: DrvSetExRAM

Eingabe: cx=Offset dx=8-bit-wert
Rückgabe: keine

Alle wesentlichen Videoparameter sind mit den Funktionen 7 DrvSetXRAM, 8 - DrvIniXRAM und 30 DrvGetXRAM manipulierbar. Die Karten FG-30-II, FG-33, FG-33-II, FG-34 und FG-35 haben zusätzliche Einstellmöglichkeiten, die mit dem Programm FG3xClip ab Version 4.86 mit der Tastenkombination Strg-F8 ausprobiert werden können. In einem mehrseitigen Dialog, der sich auch während einer Livebilddarstellung öffnet, sind alle 256 Bytes manipulierbar. Anders als beim XRAM speichert der Treiber die Einstellungen des ExRAMs nicht für jeden Videoeingang, sondern liest erst mit Aufruf von Funktion 16 mit cx=0xffff den aktuellen Hardwarestatus in einen temporären 256-Byte-langen Zwischenpuffer. Diese Werte sind mit Funktion 16 lesbar und können nach deren Manipulation mit dieser Funktion in den Puffer zurückübertragen werden. Erst mit Funktion 18 werden alle 256 Bytes aktiviert. Die Benutzung dieser Funktion erlaubt Hardwareeinstellungen, bei der die elementaren Funktionen der Karte bis zu einem Neustart beeinträchtigt werden. Es ist deshalb ratsam vor der Manipulation den bisherigen Status zu speichern und bei Bedarf zurückzusetzen, so wie es im FG3xClip durch “Rücksetzen auf vorherige Einstellungen” möglich ist.
Hinweis 1:
Normalerweise wird vor der Verwendung der Funktionen 16,17 und 18 der gewünschte Videoeingang eingeschaltet und mit

Funktion 8 aktiviert. Die dann mit Funktion 16 gelesenen Werte enthalten schon die für diesen Eingang wesentlichen Parameter.
Hinweis 2:
Nicht jeder gesetzte Wert wird als Konstante behandelt. Einige der Werte werden in Regelschleifen verwendet und haben nach erneuter Abfrage schon einen geänderten Wert. Mit dem Status solcher Werte lässt sich eine beschleunigte Umschaltung von Videoeingängen bewirken, denn die Regelschleifen würden als Anfangswert den zuletzt als stabil ermittelten Wert verwenden, was zu einer Verringerung der Einschwingzeit beiträgt.

Funktion 18
Funktionsname: DrvIniExRAM

Eingabe: keine
Rückgabe: keine

Aktiviert die durch Funktion 17 geänderten Werte.

Funktion 19
Funktionsname: DrvPeekExRAM

Eingabe: cx=offset
Rückgabe: dx=8-bit-Ergebnis

Direktes Auslesen eines Ex-Registers aus der Hardware.
Der Wert wird nicht in den Zwischenpuffer der Funktionen 16,17,18 übernommen.

Funktion 20
Funktionsname: DrvPokeExRAM

Eingabe: cx=offset, dx=8-bit-Wert

Rückgabe: keine

Direktes Setzen eines Ex-Registers in die Hardware.
Der Wert wird nicht in den Zwischenpuffer der Funktionen 16,17,18 übernommen.

Funktion 22
Funktionsname: DrvGetPal

Eingabe: keine
Rückgabe: keine

Zwischenspeichern der VGA Palette. Diese Funktion erfordert eine registerkompatible Grafikkarte.

Funktion 23
Funktionsname: DrvPutPal

Eingabe: keine
Rückgabe: keine

Rücksetzen der Palette auf die durch Funktion 22 zwischen-gespeicherten Werte.

Funktion 24
Funktionsname: DrvSetPalGrey

Eingabe: keine
Rückgabe: keine

Setzt die VGA-Palette auf 256 Graustufen.

Funktion 25
Funktionsname: DrvSetPixBits

Eingabe: cx Anzahl der bits pro Pixel
Rückgabe: keine

Teilt dem Treiber die Pixeltiefe für Funktionen der Onlinedarstellung mit. Onlinedarstellungen über den Treiber werden mit 8 oder 16 bit/Pixel unterstützt.

Funktion 26
Funktionsname: DrvGetPixBits

Eingabe: keine
Rückgabe: cx Anzahl der bits pro Pixel
Abfrage des zuletzt eingestellten Wertes von Funktion 25.

Funktion 27
Funktionsname: DrvOnlineRpt

Eingabe: keine
Rückgabe: cx grab status
cx 0= erfolgreich

Schnelle Wiederholung der zuletzt durchgeführten Onlinedarstellung. Diese Funktion erbringt einen Zeitgewinn durch Auslassung aller zur wiederholten Darstellung nicht nötigen Initialisierungen.

Funktion 28 **bis Version 4.1**
Funktionsname: DrvAcqColDiff

Eingabe: cx x - Auflösung

	dx	y - Auflösung
Rückgabe:	cx	0= erfolgreich

Diese Funktion soll für Neuentwicklungen nicht verwendet werden. Diese Funktion wird durch die Funktionen 50, 51, 52, 57, 53 ersetzt:

Halbbild					ger.	unger.		Interl.		2x16	- Bit
Funktion/Register	50cx	50dx	51cx	51dx	52cx		52dx	57	53	Pixel	sind
592x442 YUV 50Hz	296	221	0	0		72e9h	2	ja	ja	2	YUYV
768x288 YUV 50Hz	384	288	0	0	72b9h	72c9h	2	-	ja	2	YUYV
384x288 YUV 50Hz	384	288	0	0	72b9h	72c9h	2	-	ja	1	YUYV

Bis Version 4.1:
Auslösung der Digitalisierung eines Bildes im Farbdifferenzmode für 50Hz TV Standards.
Anschließend können sequenziell YUV Daten erhalten werden. Y bezeichnet eine 8 bit Helligkeitsinformation (Luminanz). U bezeichnet die Farbdifferenz Grün-Rot. V bezeichnet die Farbdifferenz Grün-Blau. Wenn der 4:2:2 Standard eingestellt ist erhält man beim sequentiellen Lesen abwechselnd YU und YV Werte.

Funktion 29 **bis Version 4.1**
Funktionsname: DrvSetColOffset

Eingabe:	cx	X - Offset
	dx	Y - Offset
Rückgabe:	keine	

Für die Funktionen zur Farbbilddigitalisierung wird ein zur Funktion 10 äquivalenter offset gesetzt. Unterschiedliche Offsetwerte für Grauwert- und Farbdigitalisierung waren nur für die

Softwareversion 1.00 erforderlich.
Ab FG3xDRV Version 2.00:
Um die Kompatibilität zu zukünftigen Versionen zu bewahren, sollte nur noch die Funktion 10 zum Einsatz kommen.

Funktion 30
Funktionsname: DrvGetXRAM

Eingabe:	cx	XRAM Adresse
Rückgabe:	dl.cl	XRAM Daten

Mit dieser Funktion wird ein Zwischenspeicher mit Parametern für die Farbdekodierung gelesen. Der Speicher hat eine Länge von 200 bits, die mit dieser Funktion byteweise gelesen werden können. Die Adresse eines bits bestimmt sich aus:

Bitposition in dl + 8 * cx

Um einzelne Bitpositionen zu ändern ist vor der Modifikation einzelner bits das entsprechende Byte mit dieser Funktion zu lesen und nach erfolgter Modifikation mit der Funktion 7 zurückzuschreiben.

Funktion 31
Funktionsname: DrvGetColOffs

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

Die mit Funktion 10 übergebenen Werte zurücklesen.

Funktion 32
Funktionsname: DrvAcqColDiff2

Eingabe:	cx	x - Auflösung
	dx	y - Auflösung
Rückgabe:	cx	0= erfolgreich

Diese Funktion soll für Neuentwicklungen nicht verwendet werden. Diese Funktion wird durch die Funktionen 50, 51, 52, 57, 53 ersetzt:

Halbbild					ger.	unger.		Interl.		2x16	- Bit
Funktion/Register	50cx	50dx	51cx	51dx	52cx		52dx	57	53	Pixel	sind
592x442 YUV 50Hz	296	221	0	0	72e9h		2	ja	ja	2	YUYV
768x288 YUV 50Hz	384	288	0	0	72b9h	72c9h	2	-	ja	2	YUYV
384x288 YUV 50Hz	384	288	0	0	72b9h	72c9h	2	-	ja	1	YUYV

Bis Version 4.1:
Die Funktion erfüllt die gleichen Aufgaben wie die Funktion 28. Der Unterschied zur Funktion 28 besteht darin, dass die Digitalisierung sofort mit dem nächsten Bildsynchronimpuls beginnt und die Hardware zur Halbbilderkennung abgeschaltet ist. Diese Funktion ist für 50Hz TV- Standards ausgelegt.

Funktion 33
Funktionsname: DrvAcqColDif4

Eingabe:	cx	x - Auflösung /2
	dx	y - Auflösung
Rückgabe:	cx	0= erfolgreich

Diese Funktion soll für Neuentwicklungen nicht verwendet werden. Diese Funktion wird durch die Funktionen 50, 51, 52, 57, 53 ersetzt:

Halbbild					ger.	unger.		Interl.		2x16	- Bit
Funktion/Register	50cx	50dx	51cx	51dx	52cx		52dx	57	53	Pixel	sind
592x442 YUV 50Hz	296	221	0	0	72e9h		2	ja	ja	2	YUYV
768x288 YUV 50Hz	384	288	0	0	72b9h	72c9h	2	-	ja	2	YUYV
384x288 YUV 50Hz	384	288	0	0	72b9h	72c9h	2	-	ja	1	YUYV

Funktion 34
Funktionsname: DrvSetPalCol8

Eingabe:	cl	rot
	ch	grün
	dl	blau
Rückgabe:	keine	

Diese Funktion setzt eine Farbpalette mit 256 Farben für den 8-Bit Colormode. Die Standardeinstellung für alle Farbkanäle erfolgt bei Übergabe des Wertes 100. Andere Werte bewirken eine Abschwächung oder Verstärkung des Farbgehaltes des jeweiligen Farbkanals.

Funktion 35
Funktionsname: DrvAcqColDiff60

Eingabe:	cx	x - Auflösung
	dx	y - Auflösung
Rückgabe:	cx	0= erfolgreich

Diese Funktion soll für Neuentwicklungen nicht verwendet werden. Diese Funktion wird durch die Funktionen 50, 51, 52, 57, 53 ersetzt:

Halbbild					ger.	unger.		Interl.		2x16	Bit
Funktion/Register	50cx	50dx	51cx	51dx	52cx		52dx	57	53	Pixel	sind
592x442 YUV 60Hz	296	221	-20	-4	72e9h		2	ja	ja	2	YUYV
640x240 YUV 60Hz	320	240	-20	-4	72b9h	72c9h	2	-	ja	2	YUYV
320x240 YUV 60Hz	320	240	-20	-4	72b0h	72c0h	2	-	ja	1	YUYV

Bis Version 4.1:
Auslösung der Digitalisierung eines Bildes im Farbdifferenzmode für 60Hz TV Standards. Analog zu Funktion 28 erhält man die Daten im YUV Format.

Funktion 36
Funktionsname: DrvAcqColDiff260

Eingabe:	cx	x - Auflösung
	dx	y - Auflösung
Rückgabe:	cx	0= erfolgreich

Diese Funktion soll für Neuentwicklungen nicht verwendet werden. Diese Funktion wird durch die Funktionen 50, 51, 52, 57, 53 ersetzt:

Halbbild					ger.	unger.		Interl.		2x16	Bit
Funktion/Register	50cx	50dx	51cx	51dx	52cx		52dx	57	53	Pixel	sind
592x442 YUV 60Hz	296	221	-20	-4	72e9h		2	ja	ja	2	YUYV
640x240 YUV 60Hz	320	240	-20	-4	72b9h	72c9h	2	-	ja	2	YUYV
320x240 YUV 60Hz	320	240	-20	-4	72b0h	72c0h	2	-	ja	1	YUYV

Bis Version 4.1:
Auslösung der Digitalisierung eines Bildes im Farbdifferenzmode für 60Hz TV Standards. Analog zu Funktion 32 erhält man die Daten im YUV Format. Nach Aufruf dieser Funktion können die Bilddaten 16-bit-sequentiell von der Basisadresse gelesen werden.

Funktion 37
Funktionsname: DrvSwitchInput

Eingabe:	cx	Eingang der aktiviert werden soll
Rückgabe:	keine	

Mit dieser Funktion wird auf einen der Eingänge des FG-32 geschaltet. Jeder Eingang verfügt über eine eigene XRAM Seite mit individuellen Parametern für die Farbdekodierung. Die zum Eingang gehörige XRAM Seite wird bei dieser Operation ausgewählt. Wichtig sind auch die Anmerkungen unter 6.1. Nach der Umschaltung des Eingangs können ggf. Änderungen der XRAM Werte erfolgen. Die Umschaltung wird erst mit Ausführung von Funktion 8 wirksam.

Funktion 38
Funktionsname: DrvGetSwSet

Eingabe:	cx	Eingang
Rückgabe:	cx	swset

Diese Funktion liest für den Eingang 0,1 oder 2 die Schalterstellung des Videorekorderbetriebs (bit 1 von swset) und des S-Video-Betriebs (bit 0 von swset).

Funktion 39

Funktionsname: DrvSetSwSet

Eingabe:	cx	Eingang
	dx	swset
Rückgabe:		keine

Diese Funktion setzt für den Eingang 0,1 oder 2 die Schalterstellung des Videorekorderbetriebs (bit 1 von swset) und des S-Video-Betriebs (bit 0 von swset). Der Wert 0 bedeutet den ausgeschalteten und der Wert 1 den eingeschalteten Zustand des jeweiligen Schalters.
Für die Framegrabber FG-30 PC-Card, FG-31 und FG-32 gibt es virtuelle S-Videoeingänge, die mit cx=1, cx=4 und cx=7 in Funktion 37 eingestellt werden können. Die Einschaltung des S-Video-Betriebes soll über die Umschaltung der Eingänge mit Funktion 37 bewirkt werden und nicht durch das Setzen des S-Video-Betriebes für die Eingänge cx=0, 2, 3, 5, 6 und 8

Funktion 40
Funktionsname: DrvGetInput

Eingabe:		keine
Rückgabe:	cx	Eingang

Diese Funktion gibt als Wert den aktiven Eingang (0...8) an. Dabei bedeutet cx= 0 die Cinchbuchse, cx=2,3,5,6,8 sind weitere Komposit-Videoeingänge. cx=1,4 oder 7 sind S-Videoeingänge. Wir ein S-Videoeingang cx=n (z.B. 4) benutzt, dann entfallen die umliegenden zwei Kompositeingänge n-1 und n+1 (3 und 5). Zwei Kompositeingänge lassen sich sozusagen zu einem S-Videoeingang zusammenfassen.

Funktion 41
Funktionsname: DrvGetBasis

Eingabe:		keine
Rückgabe:	cx	Basisadresse

Diese Funktion gibt als Wert die Basisadresse des FG-32 zurück. Die aktuelle Basisadresse muß vor Aufruf dieser Funktion gesetzt worden sein oder beträgt 310H.

Funktion 42
Funktionsname: DrvGetWaits

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

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

Funktion 43
Funktionsname: DrvEingCpy

Eingabe:	cx	Quelle (0...2)
	dx	Ziel (0..2)
Rückgabe:		keine

Die Funktion kopiert den XRAM Inhalt für Eingang <Quelle> auf Eingang <Ziel>.

Funktion 44
Funktionsname: DrvGetCardType

Eingabe: keine

Rückgabe:	cx	colflag
	dx	ycflag

Diese Funktion teilt durch colflag=1 dem Nutzerprogramm mit, dass die verwendete Grabberkarte farbfähig ist. Eine Grabberkarte, die Y/C-Signale verarbeiten kann, liegt vor, wenn ycflag=1 ist. Nur für ISA-Bus-Karten gibt es Bestückungsvarianten bei denen diese Optionen nicht vorhanden sind. Alle PC-Cards und PCI-Bus Karten sind grundsätzlich farb- und Y/C-fähig.

Funktion 45
Funktionsname: DrvAcqGreyBig

Eingabe:		keine
Rückgabe:	cx	0= erfolgreich

Digitalisierung von Grauwerten 768x576 Pixel. Nach Aufruf dieser Funktion können die Bilddaten 16-bit-sequentiell von der Basisadresse gelesen werden. Die Bilddaten folgen in der Reihenfolge ungerades Halbbild 768x288 und gerades Halbbild 768x288.

Funktion 46
Funktionsname: DrvAcqGreyBig60

Eingabe:		keine
Rückgabe:	cx	0= erfolgreich

Digitalisierung von Grauwerten 640x480 Pixel. Nach Aufruf dieser Funktion können die Bilddaten 16-bit-sequentiell von der Basisadresse gelesen werden. Die Bilddaten folgen in der Reihenfolge ungerades Halbbild 640x240 und gerades Halbbild 640x240.

Funktion 47
Funktionsname: DrvAcqGreySmall60

Eingabe:		keine
Rückgabe:	cx	0= erfolgreich

Digitalisierung von Grauwerten 640x240 Pixel. Nach Aufruf dieser Funktion können die Bilddaten 16-bit-sequentiell von der Basisadresse gelesen werden. Die Bilddaten entsprechen einem 2:1 verkleinerten Bild (640x480).

Funktion 48
Funktionsname: DrvUserOutput

Eingabe:	cx	bits
Rückgabe:	-	

Diese Funktion setzt die user I/O bits. Nach Systemstart haben alle I/O bits den Wert 1.

Funktion 49
Funktionsname: DrvSetFrameType

Eingabe:	cx	Bildtyp
Rückgabe:		keine

Setzen des Bildtyps 1=ungerades Halbbild, 2=gerades Halbbild 3=nächstes Halbbild.

Funktion 50
Funktionsname: DrvFlmSetSize

Eingabe:	cx	x - Filmbildgröße

	dx	y - Filmbildgröße
Rückgabe:		keine

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

Die folgende Tabelle zeigt einige Beispiele für interlaced und noninterlaced Bilder und Bildausschnitte. Die Beispiele entsprechen den Bildformaten, die im FG32CLIP (Kapitel 3 und 8) gegrabbt werden können.

Halbbild					ger.	unger.		Interl.		2x16	- Bit
Funktion/Register	50cx	50dx	51cx	51dx	52cx		52dx	57	53	Pixel	sind
768x576 YUV 50Hz	384	288	0	0		72e9h	2	ja	ja	2	YUYV
592x442 YUV 50Hz	296	221	0	0		72e9h	2	ja	ja	2	YUYV
768x288 YUV 50Hz	384	288	0	0	72b9h	72c9h	2	-	ja	2	YUYV
384x288 YUV 50Hz	384	288	0	0	72b9h	72c9h	2	-	ja	1	YUYV
768x576 555 50Hz	384	288	0	0		72ebh	2	ja	ja	2	2x555
384x288 555 50Hz	384	288	0	0	723fh	725fh	2	-	ja	2	2x555
768x576 565 50Hz	384	288	0	0		727bh	2	ja	ja	2	2x565
384x288 565 50Hz	384	288	0	0	72bbh	72cbh	2	-	ja	2	2x565
768x576 Grau 50Hz	384	288	0	0		72e0h	2	ja	ja	4	YYYY
768x288 Grau 50Hz	384	288	0	0	72b0h	72c0h	2	-	ja	4	YYYY
384x288 Grau 50Hz	384	288	0	0	72b2h	72c2h	2	-	ja	4	YYYY

Halbbild					ger.	unger.		Interl.		2x16	Bit
Funktion/Register	50cx	50dx	51cx	51dx	52cx		52dx	57	53	Pixel	sind
640x480 YUV 60Hz	320	240	-20	-4		72e9h	2	ja	ja	2	YUYV
592x442 YUV 60Hz	296	221	-20	-4		72e9h	2	ja	ja	2	YUYV
640x240 YUV 60Hz	320	240	-20	-4	72b9h	72c9h	2	-	ja	2	YUYV
320x240 YUV 60Hz	320	240	-20	-4	723fh	725fh	2	-	ja	1	YUYV
640x480 555 50Hz	320	240	-20	-4		72ebh	2	ja	ja	2	2x555
320x240 555 50Hz	320	240	-20	-4	72bbh	72cbh	2	-	ja	2	2x555
640x480 565 50Hz	320	240	-20	-4		727bh	2	ja	ja	2	2x565
320x240 565 50Hz	320	240	-20	-4	72bbh	72cbh	2	-	ja	2	2x565
640x480 Grau 60Hz	320	240	-20	-4		72e0h	2	ja	ja	4	YYYY
640x240 Grau 60Hz	320	240	-20	-4	72b0h	72c0h	2	-	ja	4	YYYY

320x240 Grau 60Hz	320	240	-20	-4	72b2h	72c2h	2	-	ja	4	YYYY

FG-30-II, FG-33, FG-34, FG-35 haben alle Formate. FG-30, FG-31, FG32 haben kein interlaced RGB und kein 565 RGB.

Funktion 51
Funktionsname: DrvFlmSetTopLeft

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

Setzt obere linke Ecke bezüglich des Bildgrundrasters. Wirkt nur auf die Funktionen 53, 54, und 55.

Weil die Bildlage für 50Hz- und 60Hz- Fernsehsignale sich unterscheidet, entsprechen die Werte cx=-20 und dx=-4 bei 60Hz den Werten cx=0, dx=0 für 50Hz Signale.

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:

Halbbild:	**ungerade**	**gerade**	**nächstes**
8 - bit - Grau 1:2	72B2	72C2	72E2
8 - bit - Grau 1:1	72B0	72C0	72E0
8 - bit - Farbe 1:2	7232	7252	7272

8 - bit - Farbe 1:1	7230	7250	7270
5+6+5-bit Farbe 1:2	72BF	72CF	72EF außer FG31,32
5+6+5-bit Farbe 1:1	723F	725F	727F außer FG31,32
5+5+5-bit Farbe 1:2	72BB	72CB	72EB
5+5+5-bit Farbe 1:1	723B	725B	727B außer FG30ISA
YUV 1:1	72B9	72C9	72E9

Es ist zu beachten, dass normalerweise die Auflösungen Grau 1:1, YUV 1:1 und 5+5+5 Farbe 1:1 im interlaced Mode arbeiten und das Bild Halbbildweise ausgelesen wird. Das sequentielle Auslesen erfolgt von der Basisadresse.

Das ystatus Wort zeigt die Anzahl der vollständigen Halbbilder, die zur Digitalisierung im Videosignal gescannt werden. Soll beispielsweise ein beliebiges Halbbild digitalisiert werden, hat ystatus den Wert 1, d.h. das nächste Halbbild wird sofort übernommen. Soll ein bestimmtes Halbbild digitalisiert werden, muß ystatus den Wert 2 haben, den nur 2 aufeinanderfolgende Halbbilder enthalten genau 1 Halbbild des gewünschten Typs. Für den Interlaced mode wird die Einstellung nächstes Halbbild bei ystatus=2 gewählt. Wenn die gewünschten Bildausschnitte klein genug sind oder die Pipeline schnell genug gelesen werden kann, dann können Einstellungen für ystatus gewählt werden, um mehrere aufeinanderfolgende Bilder zu Digitalisieren. Die Pipeline hat eine Größe von >480KByte x 16. Für einen Bildausschnitt von 256x256 Pixeln mit 5+5+5 RGB bit/Pixel könnten also >7 aufeinanderfolgende Bilder in der Pipeline zwischengespeichert werden.

Funktion 53
Funktionsname: DrvFlmFirstFrame

Eingabe: keine
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-Grauwert Modus, 8-bit-Farbmodus, YUV-Farbmodus und 16-bit-RGB Modus. Die Bilddaten werden sequentiell aus folgenden Adressen gelesen:

bei 16-bit-Zugriff (nur FG-30)

Grauwerte:	Basisadresse 2 Pixel je Wort
Farbe 8-bit:	Basisadresse 2 Pixel je Wort
Farbe YUV:	Basisadresse YU, YV abwechselnd
Farbe RGB:	Basisadresse RGB je Wort

bei 32-bit-Zugriff (FG-31 bis FG-34)

Grauwerte:	Basisadresse 4 Pixel je 32-bit-Wort
Farbe 8-bit:	Basisadresse 4 Pixel je 32-bit-Wort
Farbe YUV:	Basisadresse YUYV je 32-bit- Wort
Farbe RGB:	Basisadresse 2 RGB-16-bit-Pixel je 32-bit-Wort

Werden Bilder im Interlaced Mode gegrabbt, so erhält man beide Halbbilder im Videospeicher nacheinanderstehend. Um das gerade und das ungerade Halbbild zuordnen zu können gibt es zwei Möglichkeiten.

Variante 1:
Warten auf gerades Halbbild vor Auslösung der Digitalisierung:

```
m1:     mov   ax,9709h
        mov   bx,57
        int   60h
        and   cx,2000h
        jnz   m1
```

Zur Gewährleistung von FG-30(ISA)- Kompatibilität kann auch die bitmaske 0010h statt 2000h verwendet werden.

Variante 2:
Mit Beginn der Digitalisierung (cl.and80h Funktion 57) wird der jetzt anliegende Bildtyp ermittelt und in Abhängigkeit davon behandelt.

Funktion 54 **ab Version 2.1.**
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 **ab Version 2.35.**
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 **ab Version 2.35**
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 **ab Version 4.10**
Funktionsname: DrvFlmGetStatus

Eingabe: -
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:	ODD zeigt das aktuelle Halbbild des Kamerasignals an.
1000H:	RDY2=0 zeigt, dass das zweite digitalisierte Halbbild abrufbar ist.
0400H:	RDY=0 zeigt, dass das erste digitalisierte Halbbild abrufbar ist.

<u>FG-32 bis FG-35</u>

0100H	User Input von Pin 7 Sub-D-Buchse bei FG32 und FG-34

<u>FG-30-II (und FG-30-I mit ausgewählten FG30DRV Versionen)</u>
diese I/O Bits besitzen pull-up- Widerstände und können mit inputbits über das cx Register gesetzt werden. Um diese I/Os als TTL-Eingang zu nutzen, müssen die entsprechenden Bitpositionen auf 1 (high) gesetzt werden. Das kann mit Funktion 48 erfolgen.

0008H	User I/O von Pin 12, 15 poliger Frontstecker

0004H User I/O von Pin 11, 15 poliger Frontstecker

Funktion 58 **ab Version 4.10**
Funktionsname: DrvFlmBlindRead

Eingabe: cx Anzahl
dx offset zur Basisadresse
Rückgabe: keine

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

Funktion 60 **ab Version 4.10**
Funktionsname: DrvFlmAcq

Eingabe: cx
dx Basisadresse
Rückgabe: keine

Diese Funktion dient zum Starten des Grabvorgangs.

Funktion 61 **ab Version 4.10**
Funktionsname: DrvFlmWait

Eingabe: cx
dx Basisadresse
Rückgabe: keine

Diese Funktion wartet bis der Grabvorgang beginnt.

Funktion 62 **ab Version 4.10**
Funktionsname: DrvFlmReRead

Eingabe: cx
dx Basisadresse
Rückgabe: keine

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
Funktionsname: RealModeRead

Eingabe: cx Anzahl DWORDs
dx:di target pointer
Rückgabe: keine

Diese Funktion transferiert bis zu 16384 Dwords in ein Real-Mode

64K Segment.

Funktion 64
Funktionsname: ReadDword
Eingabe: keine
Rückgabe: cx- low word, dx- high word

Funktion 68 **nur FG-35**
Funktionsname: DrvSetExternalPort

Eingabe: cl 8-Bit-Datenwort
dx offset
Rückgabe: keine

Externe serielle Ports können mit dieser Funktion gesetzt werden. Der Offsetwert in dx ist die Portnummer, die Nummerierung beginnt mit 0.

Funktion 69 **nur FG-35**
Funktionsname: DrvGetExternalPort

Eingabe: dx offset
Rückgabe: cl 8-Bit-Datenwort

Externe serielle Ports können mit dieser Funktion gelesen werden. Der Offsetwert in dx ist die Portnummer, die Nummerierung beginnt mit 0.

Funktion 80
Funktionsname: DrvVgaDispCga

Eingabe: keine
Rückgabe: keine

Digitalisierung und Darstellung im VGA Mode 13H von Grauwerten 320x200 Pixel. Die 2:1 untersetzte Bildgröße beträgt 384x288 Pixel. Die Darstellung läßt sich mit den Pfeiltasten innerhalb dieses Grundrasters verschieben.

Funktion 81
Funktionsname: DrvVgaIniXRAM

Eingabe: keine
Rückgabe: keine

Diese Funktion entspricht ab Version 1.01 der Funktion 7.

Funktion 82
Funktionsname: DrvVgaAcq768HalbVga

Eingabe: keine
Rückgabe: cx 0= erfolgreich

Grauwertdigitalisierung 768x288.

Funktion 83
Funktionsname: DrvVgaAcq768

Eingabe: keine
Rückgabe: xx 0= erfolgreich

Grauwertdigitalisierung 768x576.

Funktion 84
Funktionsname: DrvVgaAcq384

Eingabe: keine
Rückgabe: cx 0= erfolgreich

Grauwertdigitalisierung 384x288.

Funktion 85
Funktionsname: DrvVgaGetOffs

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

Rückgabe der Offseteinstellung von Funktion 86.

Funktion 86
Funktionsname: DrvVgaSetOffs

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

Einstellung der Offsetwerte für die Grauwertdigitalisierung. Siehe auch Funktion 10 und Funktion 29. Nur für die Version 1.00 war eine getrennte Einstellung der Offsetwerte für die Grauwert- und Farbdigitalisierung erforderlich.

Ab FG3xDRV Version 2.00:
Um die Kompatibilität zu zukünftigen Versionen zu bewahren, sollte nur noch die Funktion 10 zum Einsatz kommen.

Funktion 87
Funktionsname: DrvVgaDispCgaColor

Eingabe: keine
Rückgabe: keine

Farbdarstellung im VGA 13H Mode. Wegen der begrenzten Auflösung von 320x200x8 bit und dem für die Onlinedarstellung gewählten 8-bit-Farbmodus erhält man mit dieser Funktion nur grobe Farb- und Helligkeitsinformationen. Zur Anwendung kommt ein YUV Pixelformat bestehend aus 4 bit Helligkeitsinformation, 2 Bit U und 2 bit V . Die in diesem Format auftretende Farbbegrenzung wirk sich in einem hohen Farbrauschen aus. Wenn diese Funktion wegen der hohen Darstellungsgeschwindigkeit (bis zu 25 Bilder/s) zum Einsatz kommt, dann sollte die Software sofort nach Einfrieren eines Bildes die Darstellung durch ein Bild ersetzen, das mit höherer Farbauflösung digitalisiert wurde und durch anschließende Farbreduktion dargestellt wird.

Funktion 88
Funktionsname: DrvVgaColImages

Eingabe: cx Segment
Rückgabe: cx 0= erfolgreich

Diese Funktion darf nur im Real Mode der CPU angesprochen werden. Ab Segmentadresse in cx sollen 256KByte lückenloser Hauptspeicher bereit stehen.

Die Funktion führt folgende Operationen durch:

1. Digitalisierung eines Farbbildes 384x288 x 24bit RGB.
2. 288 x zeilenweise Übernahme des Bildes auf die Adresse (Segment+3000H):0000
3. 288 x zeilenweise 8 - bit - Farbreduktion und Übertragung von 8 - bit - Farbwerten ab Adresse (Segment + 0000H):0000H
4. 288 x zeilenweise 4 - bit - Farbreduktion und Übertragung von 8 - bit - Farbwerten ab Adresse (Segment + 2000H):0000H

Funktion 89
Funktionsname: DrvVgaGetDither

Eingabe:		keine
Rückgabe:	cx	0 - kein dithering, 1-dithering

Abfrage des Ditheringstatus für Funktion 88.

Funktion 90
Funktionsname: DrvVgaSetDither

Eingabe:	cx	0 - kein dithering, 1 -dithering
Rückgabe:		keine

Setzen des Ditheringstatus für Funktion 88.

Funktion 91
Funktionsname: DrvVgaGetShift

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

Rückgabe der zuletzt eingestellten Bildlage der Funktionen zur Onlinedarstellung im VGA-Mode 13H (Pfeiltasten bewirken Positionierung von 320x200 Pixeln im Grundraster 384x288 Pixel bei den Funktionen zur Onlinedarstellung).

Funktion 92
Funktionsname: DrvVgaSetColPal

Eingabe:		keine
Rückgabe:		keine

Generiert und setzt eine Palette entsprechend der zuletzt ausgeführten Funktion 88 für den VGA-Mode 13H.

Funktion 93
Funktionsname: DrvVgaSaveBmp

Eingabe:	cx	Dateihandle
	dx	Segmentadresse
Rückgabe:		keine

Hilfsfunktion zum Speichern eines TrueColor Bildes 384x288 im *.BMP Format für Programme im Real Mode der CPU, die keinen Hauptspeicher für die 324 KByte umfassenden Bilddaten bereitstellen können. Ein Bild wurde z.B. mit der Funktion 88 digitalisiert. Die Bilddaten befinden sich noch im Bildspeicher des FG-32. Die Bilddaten werden zeilenweise, beginnend mit der letzten Zeile des Bildes auf die Adresse: Segmentadresse:0000H übertragen. Auf dieser Adresse steht ein temporärer Puffer von 1152 Bytes bereit, der durch diese Funktion zeilenweise in die geöffnete, und schon mit einem BITMAPFILEHEADER und BITMAPINFOHEADER versehene Datei mit der Dateihandle <cx> geschrieben wird.

Die Datei muß anschließend noch geschlossen werden.

Funktion 100
Funktionsname: DrvHcDisp

Eingabe: keine
Rückgabe: cx 0= erfolgreich

Onlinedarstellung 384x288 für ET4000 im HiColor Mode mit der Auflösung 800x600.

Funktion 101
Funktionsname: DrvHcIniXRAM

Eingabe: keine
Rückgabe: keine

Entspricht ab Version 1.01 der Funktion 7.

Funktion 102
Funktionsname: DrvHcSetXRAM

Eingabe: cx Segment
dx Offset
Rückgabe: keine

Die dem aktuellen Eingang entsprechende XRAM Seite wird mit 200 bit Daten ab Adresse cx:dx belegt.

Funktion 103
Funktionsname: DrvHcGetXRAM

Eingabe: cx Segment
dx Offset
Rückgabe: keine

Die dem aktuellen Eingang entsprechende XRAM Seite wird gelesen und 200 bit Daten werden ab Adresse cx:dx abgelegt.

Funktion 104
Funktionsname: DrvHcSetScreenParm

Eingabe: cx Segment
dx Offset
Rückgabe: keine

Aktualisiert die im Treiber zwischengespeicherten Farbwerte für Schriften und Dialogboxelemente im ET4000 HiColor Mode nach Vorgaben von Adresse cx:dx.

Funktion 105
Funktionsname: DrvHcGetScreenParm

Eingabe: cx Segment
dx Offset
Rückgabe: keine

Liest die im Treiber zwischengespeicherten Farbwerte für Schriften und Dialogboxelemente im ET4000 HiColor Mode und überträgt diese ab Adresse cx:dx.

Funktion 106
Funktionsname: DrvHcSetOffs

Eingabe: cx x - Offset

dx y - Offset
Rückgabe: keine

Einstellung der Offsetwerte für die Farbdigitalisierung.
Siehe auch Funktion 10 und Funktion 29. Nur für die Version 1.00 war eine getrennte Einstellung der Offsetwerte für die Grauwert- und Farbdigitalisierung erforderlich.
Ab FG3xDRV Version 2.00:
Um die Kompatibilität zu zukünftigen Versionen zu bewahren, sollte nur noch die Funktion 10 zum Einsatz kommen.

Funktion 107
Funktionsname: DrvHcGetOffs

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

Rückgabe der Offseteinstellung von Funktion 106.

Funktion 108
Funktionsname: DrvHcGetParm123

Eingabe: keine
Rückgabe: cx Segment
dx Offset

XRAM Daten aller drei Seiten werden ab Adresse cx:dx im Real Mode der CPU lesbar und modifizierbar bereitgestellt.

Funktion 109
Funktionsname: DrvHcSetImgType

Eingabe: cx Bildtyp
dx Größe
Rückgabe: keine

Bildtypdefinition Bildtyp=0: reserviert, Bildtyp=1: ungerades Halbbild, Bildtyp=2: gerades Halbbild, Bildtyp=3: nächstes Halbbild. Größendefinition für die im Programm ET4HICOL genutzte Bildgröße Größe=0: 384x288 Größe=1: 592x442.

Funktion 110
Funktionsname: DrvHcGetImgType

Eingabe: keine
Rückgabe: cx Bildtyp
dx Größe
keine

Rückgabe der zuletzt eingestelltem Werte von Funktion 109.

Funktion 111
Funktionsname: DrvHcDispBig

Eingabe: cx Segment
dx Offset
Rückgabe: cx 0= erfolgreich

Digitalisierung und Darstellung im ET4000 HiCOLOR Mode 800x600 eines Einzelbildes 592x442 im 24-bit-RGB Mode. Die jeweils letzten 3 bit eines Farbkanals werden ab Adresse cx:dx abgelegt und können zum Speichern von True Colorbildern zusammen mit den im Bildwiederholspeicher befindlichen Bilddaten rekombiniert werden.

Funktion 112
Funktionsname: DrvHcDispSmall

Eingabe: cx Segment
dx Offset
Rückgabe: cx 0= erfolgreich

Digitalisierung und Darstellung im ET4000 HiCOLOR Mode 800x600 eines Einzelbildes 384x288 im 24-bit-RGB Mode. Die Bilddaten werden parallel zur Darstellung ab Adresse cx:dx abgelegt und können zum Speichern oder Bearbeiten von True-Color-Bildern verwendet werden.

Funktion 113
Funktionsname: DrvHcRedispBig

Eingabe: cx Segment
dx Offset
Rückgabe: keine

Erneute Darstellung eines noch im Speicher des FG-32 befindlichen 592x442 Bildes im ET4000 HiCOLOR Mode 800x600 im 24-bit-RGB Mode. Die jeweils letzten 3 bit eines Farbkanals werden ab Adresse cx:dx abgelegt und können zum Speichern von True Colorbildern zusammen mit den im Bildwiederholspeicher befindlichen Bilddaten rekombiniert werden.

Funktion 114
Funktionsname: DrvHcSetPosBig

Eingabe: cx x Offset
dx y Offset
Rückgabe: keine

Die relative Lage eines True-Color-Bildes der Größe 592x442 wird für die Funktion 111 übergeben.

Funktion 115
Funktionsname: DrvSetIMode

Eingabe: cx Mode
Rückgabe: keine

Die Betriebsart der Halbbildinterpretation wird zwischen den Modes 0 und 1 umgeschaltet.

Funktion 116
Funktionsname: DrvGetIMode

Eingabe: keine
Rückgabe: cx Mode

Rückgabe der Betriebsart der Halbbildinterpretation.

5.2. Hinweise zum Aufrufen von Treiberfunktionen

5.2.1. Microsoft Visual C++ 1.0... 1.52
5.2.2. 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. Die Rot gekennzeichneten Zeilen sind nur für Win 3.x erforderlich.

```
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, 9709h       ;Treiberkennung
            mov     bx,0            ; initialisiere FG3xDRV
            int     60h
            cmp     bx, 9709h       ;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 WinMe/98/95 und Dos Programme würde eine Initialisierung mit:

```
_asm    {
        mov     bx,0
        mov     ax,9709h
        int     60h
        }
```

genügen, wenn sichergestellt ist, dass der Treiber FG3xDRV.EXE von der Autoexec.bat zum Systemstart aufgerufen wurde. Dazu muss die Framegrabber- Karte nicht unbedingt installiert sein.

Bilddaten sind sequentielle Datenströme, die beim FG-31...35 aus der ersten 32-Bit-Portadresse der Karte lesbar sind und bei FG30 aus einer der ersten 16-Bit-Portadresse. Zum Lesen von 16-bit I/O ports kann die C-Funktion inpw verwendet werden. Da diese Compiler keine 32-bit-Assembler Kommandos im Inline Assembler erlauben, kann die Hilfsprozedur inp_dword verwendet werden:

```
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,9709h
        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.

5.2.3. Microsoft C PDS/6.0
5.2.4. Microsoft Quick C 2.5
5.2.5. Microsoft Quick C für Windows

```
#include <dos.h>

union REGS inreg,outreg;

int    DrvInit ()
{
inreg.x.ax = 0x9709;          /* 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 32-bit Daten lesen:

```
void ReadBuffer (pbuffer, maxbuffer, basis)
int  far * pbuffer;
int maxbuffer, basis;
{
for (i=0;i<maxbuffer;i++)
        {
        inreg.x.ax=0x9709;
        inreg.x.bx=64;
        int86 (0x60, &inreg, &outreg;
        *pbuffer++ = outreg.x.cx;
        *pbuffer++ = outreg.x.dx;
        }
}
```

5.2.6. 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 aus:

```
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, 9709h       /*Treiberkennung */
            mov     bx,0            /*initialisiere FG3xDRV */
            int     60h
            cmp     bx, 9709h       /*Kennung = bx:erfolgreich */
            jnz     short nodpmi
            }
 installed=-1;
 nodpmi:                            /*Label im C-Segment */
}
```

5.3. 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 1024 Worten a 32-bit z.B. für 16 bpp könnte so als 2048x16bit Puffer gelesen werden:

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

INARY%(AXREG%)=&H9709
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 FG3xDRV 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%) = &H9709   '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)= &H9709              '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 = &H9709                              'Kennung
rgi.bx = 0                                'fkt 0:  init
CALL   INTERRUPT (&H60, rgi, rgo)
IF  rgo.bx <> &H9709   THEN  GOTO  nodpmi installed% = 1
.... weiter nach mit erfolgreicher Aktivierung des Treibers
nodpmi:
.....  weiter nach fehlerhafter Aktivierung des Treibers
```

5.4. 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.

5.5. Microsoft Macro-Assembler 6.0
5.6. Microsoft Macro-Assembler 5.1
5.7. 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, 9709h         ;Treiberkennung
      mov     bx,0              ; initialisiere FG3xDRV
      int     60h
      pop     di
      pop     ds
      cmp     cx, 9709h         ;Kennung in cx = erfolgreich
      jnz     nodpmi
      mov     ds:[di], word ptr 1

nodpmi:
      popa
```



```
cEnd
```

5.8. Turbo Pascal für DOS
5.9. 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;*

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.

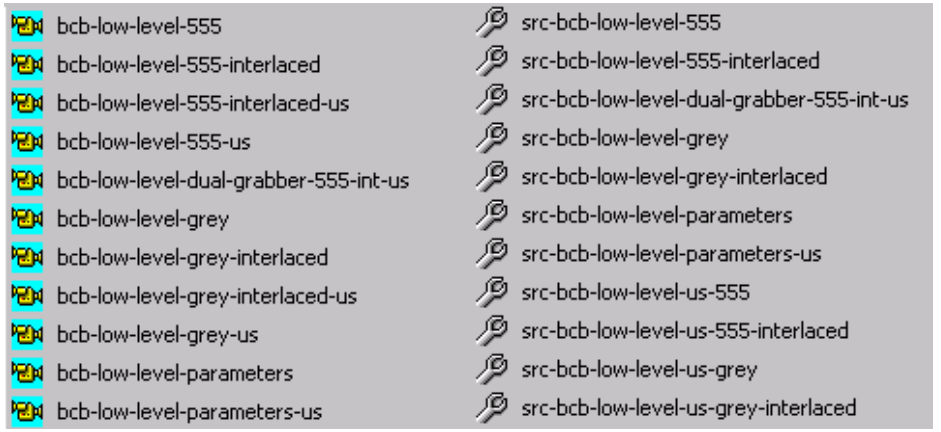
VI.
Beispiele zur low level Programmierung

6.1. Low Level Programmierung in C

Häufig benötigt man nur ein bestimmtes Bildformat und wenige Einstellmöglichkeiten. Der direkte Aufruf des Device- Treibers erweist sich dann als vorteilhaft, da keine weiteren Komponenten benötigt werden.
Die Namensgebung erklärt das Bildformat des Beispiels. Msvc-low-level

beschreibt den Compiler, für das das Beispiel geschrieben ist, in diesem Fall Microsoft Visual C (Version 2.0-6.0 und .NET). 555 steht für Farbbilder mit 32K Farben, d.h. 16 bit/Pixel. Grey steht für 8-Bit Grauwertbilder. Interlaced beschreibt die Bildaufnahme eines Vollbildes mit 768x576 Pixeln, Interlaced-us ist das Äquivalent für US-Norm- (60Hz-) Videoquellen mit 640x480 Pixeln. Ohne den Zusatz interlaced werden Auflösungen von 384x288 Pixel und mit Zusatz -us für US-Norm-Quellen 320x240 Pixel gegrabbt und dargestellt.
Mit Kamerasymbol sind die ausführbaren Dateien gekennzeichnet, das Augensymbol öffnet bei installiertem Microsoft Visual C Compiler das Projekt und den Compiler. Die Compilerumgebung lässt sich leicht prüfen, indem das unveränderte Projekt einmal übersetzt wird. Dabei wird ein Unterverzeichnis ..\RELEASE angelegt, in dem sich nun die neu übersetzte Datei FGIMAGE.EXE befindet. Die von HaSoTec gelieferte FGIMAGE.EXE befindet sich im darüber-liegendem Verzeichnis. Es kann so geprüft werden, ob beide Programme das gleiche Ergebnis liefern.

Ab Version 4.81 liegen alle Beispiele für FG32...35 auch in einer Fassung für Borland C++ Builder 6 vor. Zwischen beiden Compilern bestehen unter WinMe/9x Unterschiede im Inline-Assembler, die für die Stabilität der Software



von Bedeutung sind.
Der erweiterte Umfang der Beispiele ist ab Version 4.81 auch für Microsoft Visual C++ enthalten.
Die Beispiele ...low-level-parameters... zeigen, wie Einstellungen des Framegrabbers gespeichert, geladen und manipuliert werden können. Das Beispiel ...low-level-dual-grabber-555... zeigt, wie unter Windows XP/2000/NT mit 2 Karten Bilder quasiparallel gegrabbt werden können. Das Beispiel kann Quellen in US-Norm oder 50Hz mit 640x480 Pixeln verarbeiten und ist für andere Formate leicht modifizierbar.

6.1.1. Low Level Programmierung in C für WinXP/2000/NT

In der WM_CREATE Routine wird Speicher für das Bild angefordert und eine Device Independend Bitmap (DIB) mit den der Bildgröße entsprechenden Parametern initialisiert.
Außerdem wird der Device Driver FG32DRV.SYS geöffnet und mit folgenden Funktionen initialisiert:

```
FG30DRV (0,&cx,&dx);        //init device driver
cx=baseaddr;
dx=0;                       //enable download
FG30DRV (9,&cx,&dx);        //set baseaddr
FG30DRV (8,&cx,&dx);        //init grabber
```

Funktion 0 ist von jedem Programmstart aufzurufen. Funktion 9 setzt die

Basisadresse. Die Karten FG30...32 operieren auf fest eingestellten Adressen, deshalb genügt Funktion 9 um die Karten zu initialisieren oder um zwischen mehreren Karten umzuschalten. FG-30-I, FG-31 und FG32 benötigen downloads, um konfiguriert zu werden. Diese Downloads sind nur einmal nach Systemstart erforderlich und dauern ca. 1 Sekunde. Wird Funktion 9 mit dx ungleich Null gerufen, dann erfolgt eine schnelle Umschaltung ohne Download. Die Karte FG-30-II benötigt kein Download. FG-33, FG34 und FG35 sind plug&play Karten, die vom System konfiguriert werden. Falls bekannt ist, dass im System die Adressen 300H...30FH unbelegt sind, kann statt Funktion 9:

```
cx=300H;
dx=0;                       //enable download
FG30DRV (9,&cx,&dx);        //set baseaddr
FG30DRV (41,&cx,&dx);       //get baseaddr
basis=cx;
```

eingesetzt werden. Dabei wird ein Download auf Adresse 300H versucht und, falls ein Framegrabber FG-32 vorhanden ist, konfiguriert. Die Abfrage der Basisadresse ergibt dann den Wert 300H und falls kein FG-32 vorhanden ist und mindestens ein FG-33, FG34 oder FG35 installiert ist, wird die Basisadresse der zuerst gefunden Karte zurückgegeben. Diese Sequenz ist somit geeignet, um verschiedene Systeme mit jeweils einer Karte FG-32 bis FG-35 mit einem einzigen Programm anzusprechen. Feste Adressen kommen in den Beispielen für FG-30 und FG-31 zum Einsatz, für alle übrigen Karten wird das gleiche Beispiel installiert, indem die genannte Kommando- Sequenz verwendet wird.

Funktion 8 initialisiert die Videoparameter. Das muss mindestens einmal nach Systemstart erfolgen. Es können davor auch Einstellungen geändert werden. Funktion 8 aktiviert die (ggf. geänderten) Einstellungen des Treibers durch Übertragung in die Chipsätze der Karte. Werden später Einstellungen geändert, dann werden diese erst mit erneutem Aufruf von Funktion 8 aktiv.

Der eigentliche Grabvorgang geschieht durch die Funktionen 50,51,52 und 53. Als Grabvorgang wird das Auslösen der Digitalisierung eines nächsten Bildes in den Speicher der Framegrabberkarte verstanden. Um ein Bild zu erhalten ist später noch das Auslesen dieses Speichers erforderlich, das nachfolgend als Bilddatentransfer bezeichnet wird.

Die Funktionen 50, 51 und 52 legen Bildausschnitt und Bildformat fest. Für die

volle Auflösung muss im interlaced Mode (1:1) gearbeitet werden. 1:2 verkleinerte Bilder, die auf einem Halbbild basieren sind ebenfalls möglich. Damit ergeben sich zwei Grundraster für 50Hz: 768x576 und 384x288 und zwei Grundraster für 60 Hz: 640x480 und 320x240. Innerhalb dieser Grundraster lassen sich rechteckige Bildausschnitte festlegen, es werden dann nur die im Ausschnitt liegenden Daten in den karteninternen Speicher übertragen und später ausgelesen. Qualitätsmindernde Skalierungen auf bestimmte Bildgrößen lassen sich mit dieser Funktionalität vermeiden. Dazu ein Beispiel: Es sollen Bilder der Größe 280x200 Pixel verabreitet werden. Lösung: Es wird das Grundraster von 384x288 (50Hz) oder 320x240 (60Hz) verwendet und eine Bildausschnit von 280x200 eingestellt. Statt einer Skalierung von 384x288 (oder 320x240) auf 280x200 wird die Kameraanordnung so gewählt, dass Kameraabstand und Objektiv im Ausschnitt statt im Grundraster das gewünschte Objekt zeigen.
Im Interlaced mode wird die Funktion 57 verwendet, um die Digitalisierung in einem geraden Halbbild zu starten, damit beim Auslesen die Halbbilder in der richtigen Reihenfolge kommen.

Die Bilddatentransfer- Funktionen für Windows XP/2000/NT sind tabellarisch in Abschnitt 1.1 dieses Kapitels beschrieben.
Die Bilddarstellung erfolgt mit der Windows API Funktion SetDIBBitsToDevice. Mit DirectX oder DrawDibDraw- Funktionen können schnellere und skalierbare Darstellungen erreicht werden.
Zwei weitere Beispiele: msvc-low-level-parameters und msvc-low-level-parameters-us zeigen, wie eine einfache Live-Bild-Darstellung per Timerfunktion erreichbar ist und wie Videoparameter gespeichert, modifiziert und zurückgeladen werden.

6.1.2. Low Level Programmierung in C für WinMe/98/95

Die im vorangegangenem Abschnitt beschriebenen Beispiele sind auch für WinMe/98/95 verfügbar. Das Öffnen des Device Drivers ist hier nicht erforderlich.
Alle int- 60h- Aufrufe sind im Abschnitt 5.1 erläutert.
Der Bilddatentransfer erfolgt mit insd- (bzw. FG-30:insw-) Befehlen, mit dem eine Anzahl Dwords (Words) von der Framegrabberkarte in den Bildpuffer transferiert werden. Für die 1:2 Bildformate genügt ein einziger Aufruf. Im interlaced Mode ist das schon ein wenig aufwendiger. Das soll am Beispiel low-level-interlaced-555 erklärt werden:

Die im Dual-Port-Speicher der Karte liegenden Bilddaten sind in gleicher Reihenfolge lesbar, wie diese von der Kamera gesendet werden. Das ungerade Halbbild 768x288 kommt zuerst und wird mit:

```
                mov     edi,pimg
                mov     ecx,288
odd00:          push    ecx
                mov     ecx,768/2
                rep     insd            ;fill buffer with image data
                add     edi,768*2
                pop     ecx
                loop    odd00
```

transferiert. Im Bildpuffer wird dazu jede zweite Zeile freigelassen. Rep insd transferiert jeweils 384 Dword mit jeweils 2 Pixeln a 16 bit zum pointer edi. Also genau eine Zeile. Nach jedem Transfer werden 384 Dwords übersprungen und die genannten Operationen 288 mal wiederholt.
Zwischen den Halbbildern werden zwei Zeilen mehr digitalisiert. Diese Zeilen werden jetzt blind gelesen, um den Anschluss zum nächsten Halbbild herzustellen:

```
                mov     ax,9709h
                mov     bx,116
                int 60h                 ;drvgetimode
                and     cx,cx
                jz      rb01
readblind:      mov     bx,384                  ;384 dwords=1
zeilerb00:              in      eax,dx
                dec     bx
                jnz     rb00
                dec     cx
                jnz     readblind
rb01:
```

Die Funktion 116 ist auf den Wert 2 voreingestellt. Für Videosignale mit inkorrekter Zeilenzahl oder Synchronimpulsen kann dieser Wert im Bereich 0...4 verändert werden, damit die jeweils richtigen Zeilen der Halbbilder nebeneinander liegen.
Das zweite Halbbild wird nun in die übersprungenen Bildpuffersegmente gefüllt:

```
                mov     edi,pimg
                mov     ecx,288
```



```
even00:         push    ecx
                mov     ecx,768/2
                add     edi,768*2
                rep     insd            ;fill buffer with image data
                pop     ecx
                loop    even00
```

Die Reihenfolge von insd und add edi,384 ist dazu lediglich getauscht worden. Wie in den XP/2000/NT Beispielen wird auch hier SetDibBITsToDevice zur Darstellung verwendet. Mit geringen Modifikationen ist das Beispiel auch für Windows 3.x verwendbar. Dazu muss die Reihenfolge der Zeilen umgekehrt transferiert werden und das negative Vorzeichen des Wertes pbi->bmiHeader.biHeight entfernt werden. Weil 16-Bit-Daten nicht verarbeitet werden, müssen die Pixel zur Farbdarstellung zu 24-Bit-RGB Werten konvertiert werden.

6.2. Low Level Programmierung in Pascal

Die in 6.1.1. beschriebenen Beispiele liegen in einer Fassung für Delphi vor. Die Beispiele sind mit Delphi 6 übersetzt worden. Unter älteren Delphi Versionen müssen die Projekte neu erzeugt werden.