Allgemein:
Der Vertrieb und die Weiterentwicklung wurde 1998 beendet. und lief bis dahin unter der Firma **SHD** GmbH .
Das gesamte Produkt ist somit nach 10Jahren alleiniges Eigentum des Entwicklers.
Das gesamte CNC Paket wurde **komplett** entwickelt von:

Reinhard Heuberger
Watzmannring 71
85748 Garching
Tel: 089-3205134
E-Mail: Reinhard.Heuberger@gmx.de

Programm Architektur: Die Software Produkte wurden mit GFA-Basic für DOS und mit MASM-Assembler erstellt. Uneingeschränkt lauffähig ist die Software nur unter DOS, bedingt durch das direkte Programmieren der parallelen Schnittstelle durch Assembler Code. Der grafische SimulationsTeil läuft allerdings auch problemlos aus einer MS-DOS Box unter Windows XP. Weitere Infos dazu unter folgenden Kapitel "Bootable-DOS-CD".

Das gesamte Paket ist im Ordner **SHD** hinterlegt.
mit folgende **Unterordner:**

BILDER : JPG Dateien zur Produkt Übersicht.

Bootable-DOS-CD : Die Software läuft nur aktiv unter DOS wegen der eingesetzten Assembler Routinen. In diesem Ordner ist eine Anleitung mit allen nötigen Programme, um eine bootbare CD mit DOS und RAM Disk zu erstellen. Dies ermöglicht somit auch den Einsatz der Software auf modernen Rechnern.

Distribution-Floppys : In diesem Ordner sind Unterordner für jedes Produkt mit den dazugehörigen Floppy-Disk Inhalt. Auslieferfertig mit Installations Programm.

Dokumentation : Hier ist die gesamte Dokumentation hinterlegt in WORD Format.

Fuer-ATARI : Hier befinden sich alle Ordner mit Software für die ATARI Systeme.
Info: Ein grosser Teil Software wurde ursprünglich für die ATARI Computer entwickelt und später auf die INTEL System portiert, da ATARI den Computer Markt aufgegeben hat.

SHD-Disk-Kopie-#1=C und **SHD-Disk-Kopie-#2=D** : Kompletter Disk Abzug , also komplette Entwicklungs-Umgebung Der System Disk C und Daten Disk D für MS-DOS und Windows 3.1.

SHD-SOURCEN : Hier sind alle Programme als Quellen Code unter den jeweiligen Produkt Namen als Unterordner hinterlegt.

Die Entwicklungs-Umgebung

Generell: Die gesamte Software wurde in GFA Basic und Microsoft Assembelr (MASM) geschrieben. Die GFA-Programme haben die Endung ".GFA" und als List-Datei die Endung ".LST".

Der Disk Abzug im Ordner: **SHD-Disk-Kopie-#1=C**
Die GFA-Basic Entwicklungs Umgebung wurde mit der Batch Datei **GFA.BAT** gestartet:
cd gfa
gfabas37
cd ..
Ab diesen Zeitpunkt befindet man sich in der **GFA-Basic Oberfläche** und kann Programme
erstellen, ändern und sofort laufen lassen.
Die GFA-Basic Programme befinden sich im Ordner **GFA** und sind dort nochmals nach Produkt
in eigenen Ordnern zu finden. Z.B. HPGL-Produkt unter C:\GFA\HPGL .

Besonderheiten beim Produkt **DIN66025** und **DIN_JUN** : Wegen der eingeschränkten Address Möglichkeit unter MS DOS sind diese Produkte in der OVERLAY Technik geschrieben und werden von einer Batch-Datei gesteuert: Beispiel: DINJUN.BAT :

```
echo off
:anfang
root
if errorlevel 9 goto ende
if errorlevel 8 goto label_1
if errorlevel 7 goto label_2
if errorlevel 6 goto label_3
if errorlevel 5 goto label_4
if errorlevel 4 goto label_5
if errorlevel 3 goto label_6
goto anfang
:label_1
modul1
goto anfang
:label_2
modul2
goto anfang
:label_3
modul3
goto anfang
:label_4
modul4
goto anfang
:label_5
modul5
goto anfang
:label_6
modul6
goto anfang
:ende
echo on
exit
```

Die Kommunikation innerhalb den Modulen wird durch die Datei **cncinf.dat** gesteuert.

Will man die Programme unter GFA Basic laufen lassen, so muß man immer erst den Progrmmteil ROOT.GFA laden und starten um die Datei **cncinf.dat** zu erzeugen. Danach können die anderen Module geladen und ausgeführt werden.

KUNDEN Registrierung

Die Kunden Registrierung erfolgt über die Datei: **PERSOENL.ICH** . Diese Datei wird mit dem Verschlüsselungs Programm GEHEIM im Ordner GFA\123TOOLS erstellt. Bei Auslieferung des Produkts wird die Datei **NOCHNI.CHT** mitgeliefert. Diese Datei enthält eine Aufforderung zur Registrierung.
Jedes Programm überprüft beim Start das Vorhandensein einer dieser Dateien. Wird keine Datei gefunden, so wird das Programm beendet mit der Fehlermeldung: Programm kann NICHT gestartet werden , uzureichende Systemparameter.

Programm Strukturen bei Assembler Routinen Einbindung.

Generell : Die Programm Struktur ist modular aufgebaut in drei Schichten:
1) Grafisches User-Interface 2) Programm Code **3) Hardware Steuer-Rountinen**

Durch Austausch der Schicht 3 kann unterschiedliche Hardware angesteuert werden. Z.B: Über die seriell Schnittstelle für die Produkte der Firma SYSTEC. Die Programme dazu sind im Ordner \SHD\SHD-Disk-Kopie-#1=C\GFA\9SYSTEC
Die Programme HPGL und DIN3D wurden für Produkte von Fa. SYSTEC portiert.

Alle Programme , also **GRAVUR , HPGL, DIN3D, DIN66025** laufen auf der SHD Hardware , SHD Adapter BOX und SHD 19“ Einschub (Fa. ISEL Pin-Kompatibel)
Die Assembler-Rountinen sind in \SHD\SHD-SOURCEN\ADAPTER-BOX-Version

Assembler Routinen Einbindung: Trotz der Mächtigkeit und Komfortabilität von GFA-Basic (Pre-Code Interpreter) war es sehr schwierig (im Gegensatz zu C) den Assembler Code einzubinden. Um dies dennoch komfortabel zu realisieren stehen einige Hilf Programme zur Verfügung: \SHD\SHD-Disk-Kopie-#1=C\GFA\123TOOLS
Dort findet sich das Programm **1CONVERT.GFA** , welches den Assembler Code nach GFA “konvertiert“. Am Ende der Assembler Rountinen ist das Statement:
endb1 dd 1 dup(12345678h) . Diese Zahlenkombination ist dann der weitere Anhaltspunkt zur Berechnung der Indexe unter GFA Basic.

Siehe dazu das Programm: \SHD\SHD-Disk-Kopie-#1=C\GFA\123TOOLS\STEP_0/1/2
Die Inizialisierung der Assembler Rountine erfolgt mit Aufruf von PROCEDURE Init()
Der Assembler-Code wird von den DATA Anweisungen in das Array mcc%() geladen.

Der Aufruf von der Assembler Rountine unter GFA Basic sieht dann so aus:
CALL (**mc%**) (**_EAX**=xstepp%,**_EBX**=ystepp%,**_ECX**=zstepp%,**_EDX**=speed%)
wobei gilt:
mc% = Pointer zu 32bit Integer Array ,mcc% welches den Assembler Code beinhaltet. GFA-Pointer Berechnung: mc%=V:mcc%(0)
_EAX bis **_ECX** = Intel 32 Bit Register für die Daten-Übergabe x/y/z-stepp% u. Speed%

Hinweis : Es handelt sich immer um 32 bit Integer Werte.

Architektur und Datenstrukturen

Alle Programme, also das **GRAVUR , HPGL, DIN3D und DIN66025** Paket nehmen die Parameter aus der Datei **CNCPARAM.DA*** , welche mit dem Programm **CNCSETUP** erstellt wird. Eine CNCPARAM Datei sieht dann wie folgt aus :

```
DIN3D: KUNDENSPEZIFISCHES CNC-PARAMETERFILE ( SHD-BOX-Version )
200,200,200                     = Schritt-Auflösung pro mm , x,y,z Achse
4825406,4825406,4825406         = Speed-Faktor , x,y,z Achse
4000,500,150                    = Beschleunigungs/Brems/Speed Charakteristik
600,400,50                      = Maschinen Größe in mm , x,y,z Acchse
0,0,0,0,0                       = R E S E R V E D
2177,1971,43236                 = R E S E R V E D
C:\                             = Ramdisk Definierung
Datei erstellt am: 07.02.1995   = Erstellungs Datum
```

Nach dem Einlesen der Datei wird u.a. die Assembler Routine mit den dementsprechenden Parametern versehen, Maximal-Speed, Anfahr-Speed, Beschleunigungs/Brems Faktor .
Die Beschreibung des SETUP Programms : **SHD\Dokumentation\Install+setup\INSTALL.DOC.**

DATENSTRUKTUREN

Allgemein:

- Die Berechnung der Koordinaten erfolgt immer mit REAL-Werten.
  X : xwert **Y**: ywert **Z**: zwert
- Die Bewegung der Achsen erfolgt immer relativ mit 32Bit Integer Werten:
  X : xstepp% **Y**: ystepp% **Z**: zstepp%
- Die Physikalische Position wird immer in der Assembler Rountine in Schritt Werten aktualisiert/mitgeführt: **X** : mcc%(xphp&) **Y**: mcc%(yphp&) **Z**: mcc%(zphp&)

Durch dieses Konzept kann immer die genaue Position ermittelt werden, auch wenn der Fräsvorgang unterbrochen wird, da immer Zugriff auf die physikalische Position besteht. Weiterhin ist sichergestellt, daß nie(!) ein Schrittverlust durch Berechnugs Fehler auftritt

Auszug: So werden immer die Schritte berechnet:

```
xstepp%=CINT(xwert*xr%)+xnull%-mcc%(xphp&)
ystepp%=CINT(ywert*yr%)+ynull%-mcc%(yphp&)
zstepp%=CINT(zwert*zr%)+znull%-mcc%(zphp&)+fr_lae%
```

xstepp%, ystepp%, zstepp%	= Relative Bewegung in Schritten, X,Y,Z Achse
xwert, ywert, zwert	= Koordinaten der X,Y,Z Achse, =Real Wert.
xr%, yr%, zr%	= Auflösung der Achsen : Schritte/mm
xnull%, ynull%, znulll%	= Nullpunkt Koordinaten, X,Y,Z Achse
mcc%(zphp&), mcc%(zphp&), mcc%(zphp&)	= Physikalische Position in Schritten
fr_lae%	= Fräser Längen Korrektur Wert.
mcc%()	= Integer Array(Beinhaltet Assembler Routine).
xphp&, yphp&, zphp&	= Pointer für X,Y,Z Achse

CINT GFA-Basic: Wandelt eine Fließkommazahl in einen gerundetetn Integerwert.

Um eine **3D Linear Bewegung** durchzuführen ist es nun nur notwendig die Assembler Routine auzurufen, was mit der Procedure **do_line** realisiert ist.
Die Procedure **do_line** sieht dann wie folgt aus :

```
PROCEDURE do_line
  IF mcc%(done&)=0
    CALL (mc%) (_EAX=xstepp%,_EBX=ystepp%,_ECX=zstepp%,_EDX=speed%)
  ENDIF
  IF simulieren!=FALSE
    IF eschr!=TRUE
      @singlestep
    ENDIF
  ENDIF
RETURN
```

Für eine **3D Circular (HELIX) Bewegung** ist es dann demzufolge nur notwendig , die **Procedure do_g2g3(m_x,m_y,k_radius,from_x,from_y,z_hub)** aufzurufen.
Dabei gilt :

m_x	= Kreis-Mittelpunkt , X-Koordinate.
m_y	= Kreis-Mittelpunkt , Y-Koordinate.
k_radius	= Kreis Radius.
from_x / from_y	= Wird bei bei grafischer Simulatiuon benützt
z_hub	= Höhe / Tiefe

Auszug aus **Procedure do_g2g3 :**

```
> PROCEDURE do_g2g3(m_x,m_y,k_radius,from_x,from_y,z_hub)
  '
  ' Kreis - Fräs,- Ein/Austauch Routine für X & Y & Z Achse.
  '
  LOCAL tempoldz%,xschritt%,yschritt%,zschritt%,ic%
  LOCAL zstepp,from_z,to_z,temp!
  IF mcc%(done&)=0
    IF z_hub=0                              = Keine Helix Interpolation
      zstepp%=0
      i=kanf
      FOR ic%=kr_a% TO kr_e% STEP kr_s%
        xschritt%=CINT((COSQ(i)*k_radius+m_x)*xr%)
        yschritt%=CINT((SINQ(i)*k_radius+m_y)*yr%)
        xstepp%=xschritt%-mcc%(xphp&)
        ystepp%=yschritt%-mcc%(yphp&)
        CALL (mc%) (_EAX=xstepp%,_EBX=ystepp%,_ECX=zstepp%,_EDX=kspeed%)
        EXIT IF mcc%(done&)=1
        i=i+k_schritt
      NEXT ic%
      xstepp%=CINT(pkrendx*xr%)-mcc%(xphp&)
      ystepp%=CINT(pkrendy*yr%)-mcc%(yphp&)
      IF mcc%(done&)=0
        CALL (mc%) (_EAX=xstepp%,_EBX=ystepp%,_ECX=zstepp%,_EDX=kspeed%)
      ENDIF
    ELSE
    .
    .....hier kommt Helix-Teil , siehe Quellen-Code
```

Hinweis: Für die Vor-Berechnung der Circular Bewegung sind die Proceduren :
PROCEDURE **kreisefraesen** , PROCEDURE **get_winkel** und
PROCEDURE **get_circ_param** notwendig.

Details dazu im Quellen-Code : SHD\SHD-SOURCEN\ADAPTER-BOX-Version

Thema: Portierung und Aufwand

Um die Software für eine andere I/O Umgebung als die Parallele Schnittstelle zu portieren sind prinzipiell 2 Aktionen erforderlich:

1) Änderung und Anpassen der Assembler Routine auf eine andere I/O Hardware
2) Und/Oder: Änderung der Proceduren **do_line** und **do_g2g3**

Zu 1) : Ein Beispiel dazu im Ordner: SHD\SHD-SOURCEN\INTERFACE-PCBUS-Version Die Software wurde bereits auf eine AT-Bus Karte mit MC8255 Chip Satz portiert. Dazu wurden lediglich die Assembler Routinen umgeschrieben **Hinweis**: Das Produkt ging nie in Serie und wurde nie verkauft.

Zu 2) : Ein Beispiel dazu im Ordner: SHD\SHD-Disk-Kopie-#1=C\GFA\9SYSTEC Die beiden Software Pakete HPGL und DIN3D wurden bereits für die Steuer/Treiber Einheiten der Firma SYSTEC portiert, welche über die Serielle Schnittstelle angesteuert werden. Die SYSTEC Steuer-Einheiten stehen je nach Leistungsbedarf für **Servo und/oder Schrittmotoren** zur Verfügung. Der Treiber für die Schnittstelle wurde von Firma V24_OEM bezogen. Infos mit Beispielen dazu: SHD\SHD-Disk-Kopie-#1=C\GFA\V24_OEM und weitere Infos sind zu finden im Ordner: SHD\SHD-Disk-Kopie-#1=C\DEV

Schritt-Motor-Ansteuerung SHD Hardware :

Über die parallele Drucker Schnittstelle wird die SHD Adapter Box und die 19“ Karte (Pin kompatibel ist zu den ISEL Fräsmaschinen/Steuerungen) angesteuert. Bei der Adapter Box sind zusätzlich Open Collector Ausgänge vorhanden um Endstufen im professionellen Bereich mit Opto Coupler Eingänge anzusteuern (z.B. Fa Berger/Lahr, Fa. Pytron). Eine genaue Beschreibung ist im Ordner : SHD\Dokumentation\Install+setup

Andere Schritt-Motor Ansteuerung:

Derzeit gibt es mehrere Anbieter von Schrittmotoren Ansteuerungen in unterschiedlicher Ausführung. Getestet wurde die **3D-Step** Steuerung von Fa. **NC-Step** (http://www.nc-step.de/). Beim Einsatz dieser Steuerung wird die SHD Hardware nicht benötigt. Die Steuerung wird direkt an die parallele Schnittstelle angeschlossen, allerdings ist die PIN-Belegung des parallelen Ports anders. Die Unterschiedlichen PIN Belegungen :

PIN 2, Data Bit 0	SHD: Takt X	3D-Step: Richtung X
PIN 3, Data Bit 1	SHD: Takt Y	3D-Step: Takt X
PIN 4, Data Bit 2	SHD: Takt Z	3D-Step: Richtung Y
PIN 5, Data Bit 3	SHD: Takt C	3D-Step: Takt Y
PIN 6, Data Bit 4	SHD: Richtung X	3D-Step: Richtung Z
PIN 7, Data Bit 5	SHD: Richtung Y	3D-Step: Takt Z
PIN 8, Data Bit 6	SHD: Richtung Z	3D-Step: frei (opt. Richt. C)
PIN 9, Data Bit 7	SHD: Richtung C	3D-Step: frei (opt. Takt C)
PIN10, Acknowledge	SHD: End/Referenz X	3D-Step: End/Referenz Z
PIN11, Busy	SHD: STOP/SW-Controll	3D-Step: Enable = „Stop“
PIN12, Paper Out	SHD: End/Referenz Y	3D-Step: End/Referenz Y
PIN13, Select	SHD: End/Referenz Z	3D-Step: End/Referenz Z
PIN15, Error	SHD: End/Referenz C	3D-Step: frei (opt. End/Referenz C)

Hardware Anpassung:

Um diesen Unterschied auszugleichen müssen einige Adern gekreuzt werden nach obiger Tabelle. Die Nummerierungen beziehen sich auf den D-SUB Stecker. (Z.B. PC: Pin 6 --> 3D-Step:Pin 2)
Die Funktion des **PIN11(Busy)** ist nicht direkt übertragbar. Aus Software Sicht hat dieser Pin immer eine Stop Funktion zur Folge mit Rückrechnung der noch ausstehenden/verlorenen Schritte . Die 3D-Step Hardware schaltet (Busy =aktiv=Stop) zusätzlich die Endstufen automatisch stromlos. Letztendlich muß nach einen Stop sowieso eine Referenzfahrt gemacht werden und somit ist funktionell kein Unterschied .

Alternativ kann man die Assembler Routine umschreiben und sich somit ein Adapter-Kabel sparen.

Dazu ein Beispiel wie die Assembler Routinen an die Vorgaben der **3D-Step** Karte von Fa. **NC-Step** angepasst werden können. Die Programme sind im Ordner:
SHD\SHD-SOURCEN\ADAPTER-BOX-Version\ASSEMBLER-Routinen-Adapter-Box\
SHBOX_02.ASM = Assembler Source Programm
SHBOX_02.LST = Assembler Listing .

Assembler Programm Auszug von **SHD Adapter Box** Version :

```
0075        mov     al,00001000b    ; Set X - forward bit
              .
0088        or      al,00010000b      ; SET Y - forward bit.
              .
0099        or      al,00100000b      ; Set Z - forward bit.
             .
00D7        mov     cs:[di]+36,BYTE PTR 00000001b ; Setup sequenze if
00DC        mov     cs:[di]+40,BYTE PTR 00000010b ;  X-Axis is
00E1        mov     cs:[di]+44,BYTE PTR 00000100b ;       main-axis
             .
00FC        mov     cs:[di]+36,BYTE PTR 00000010b ; Setup sequenze if
0101        mov     cs:[di]+40,BYTE PTR 00000001b ;  Y-Axis is
0106        mov     cs:[di]+44,BYTE PTR 00000100b ;       main-axis
             .
011C        mov     cs:[di]+36,BYTE PTR 00000100b ; Setup sequenze if
0121        mov     cs:[di]+40,BYTE PTR 00000001b ;  Z-Axis is
0126        mov     cs:[di]+44,BYTE PTR 00000010b ;       main-axis
```

Das Assembler Programm muß wie folgt für die **3D-Step Karte** angepaßt werden:

```
0075        mov     al,00000001b    ; Set X - forward bit
              .
0088        or      al,00000100b      ; SET Y - forward bit.
              .
0099        or      al,00010000b      ; Set Z - forward bit.
             .
00D7        mov     cs:[di]+36,BYTE PTR 00000010b ; Setup sequenze if
00DC        mov     cs:[di]+40,BYTE PTR 00001000b ;  X-Axis is
00E1        mov     cs:[di]+44,BYTE PTR 00100000b ;       main-axis
             .
00FC        mov     cs:[di]+36,BYTE PTR 00001000b ; Setup sequenze if
0101        mov     cs:[di]+40,BYTE PTR 00000010b ;  Y-Axis is
0106        mov     cs:[di]+44,BYTE PTR 00100000b ;       main-axis
             .
011C        mov     cs:[di]+36,BYTE PTR 00100000b ; Setup sequenze if
0121        mov     cs:[di]+40,BYTE PTR 00000010b ;  Z-Axis is
0126        mov     cs:[di]+44,BYTE PTR 00001000b ;       main-axis
```

Weitere Schritte:
-1- Assembler Programm neu mit masm übersetzen
-2- Assembler Programm In GFA-Basic Code konvertieren (mit Programm 1CONVERT.GFA)
-3- Den konvertieren GFA Code (Data lines) ins das jeweilige Programm neu laden/mergen.

********** Fertig, schon läuft alles mit der 3D-Step Karte **********