fischertechnik Interface

umFish20.DLL

Handbuch zu Version 2.0

Ulrich Müller

Inhaltsverzeichnis

Dokumentname : umFish20.doc. / Fisch6.WMF Druckdatum : 03.06.2002

umFish20.DLL

Allgemeines

umFish20.DLL ist eine in VC++ 6.0 geschriebene 32bit DLL, die Zugriffe auf die fischertechnik-Interfaces 30520 (einschl. CVK fischertechnik Interface von Cornelsen) und 30566 (parallel) und 30402 (seriell und Extension Module 16554) – beide auch im Master-Slave-Betrieb - ermöglicht. Dazu werden eine Reihe von Basisfunktionen angeboten, die aus den Programmiersprachen Visual Basic, C/C++ und Delphi genutzt werden können.

umFish20.DLL ist eine Basis für eigene Entwicklungen von komplexeren Funktionen. Alternativ werden mit umFish20Ex (für Einsteiger) und FishFace (für Fortgeschrittete) Komponenten mit komplexeren Funktionen angeboten, die ihrerseits auf umFish20.DLL aufsetzen. Die Form der Parameterübergabe und des Funktionsaufrufes wurde gewählt um eine Nutzung dieser Basisfunktionen aus unterschiedlichen Programmiersprachen zu ermöglichen (getestet wurde mit VB6, VC++6.0, C++Builder 5.0 (hier mit umFish20Load.h) und Delphi4).

umFish20.DLL ist unter den Betriebssystemen Windows95/98/Me, Windows NT 4.0 ab SP3 und Windows 2000 einsetzbar. umFish20.DLL kann, wie DLL-üblich, im Pfad der Anwendung oder im Win/System-Verzeichnis liegen, eine Registrierung ist nicht erforderlich. Zum Betrieb eines parallelen Interfaces ist ggf. ein zusätzlicher WinRT-Treiber erforderlich.

umFish20.DLL ist eine erweiterte Version umFish.DLL, sie ist zu umFish nicht kompatibel. Neu der ftDCB-Kontrollblock und ein Impuls-Counter. umFish20.DLL fragt jetzt, in wählbaren Intervallen(MultiMediaTimer), das jeweiligen fischertechnik Interface kontinuierlich ab. Das ermöglicht eine exaktere Positionsbestimmung besonders, in Verbindung mit dem Impul-Counter, bei Robots. Für "normale" Anwendungen kann weiterhin umFish.DLL eingesetzt werden, es ist in der Handhabung einfacher.

Ulrich Müller, D-33100 Paderborn, Lange Wenne 18, Fon 05251/56873, Fax 05251/55709
eMail : ulrich.mueller@owl-online.de
Homepage : www.ftcomputing.de

Hinweise

Refresh

Die fischertechnik Interfaces benötigen ca. alle 300 mSek ein Refresh des Status der (Motor)Ausgänge. Sonst werden die Ausgänge abgeschaltet, das geschieht um das Modell vor in die Irre gelaufenen Programmen zu schützen. Da umFish20.DLL intern in kürzeren Abständen auf das Interface zugreift, ist diese Funktion (das Abschalten) praktisch deaktiviert.

Interface Parameter

Zur Ansteuerung des Interface benötigt um20Fish einen Kontollblock (ftDCB), der beim OpenInterface mit aktuellen und Defaultwerten (teilweise in Abhängigkeit vom Interface) besetzt wird. Die Werte sind so gewählt, daß bei normalem Einsatz keine Änderungen erforderlich sind. Das heißt : kein Slave Interface (ECount=8), kein Scannen der Analogeingänge (AnalogScan=0).

Die parallelen Interfaces unter Win95 / 98 / Me werden durch die Rechnergeschwindigkeit beeinflußt. Hier ist ggf. eine Korrektur der Parameter LPTDelay (Default 10, Rechner ab Pentium 400 höher) und LPTAnalog (Default 5, dto.).

Eventuelle Modifikationen sollten vor dem OpenInterface erfolgen.

Unterbrechbarkeit

Wird das Interface in einer engen Schleife angesteuert z. B. Loop zur Abfrage eines Digitaleinganges, ist das Programm nicht mehr unterbrechbar. D.h. es reagiert nicht mehr auf Control-Buttons und macht meist auch keine Bildschirmanzeige mehr. Hier ist programmtechnisch für eine Unterbrechbarkeit durch Abgabe des Zeitscheibe an das Betriebssytem (VB : DoEvents, Delphi : Application.ProcessMessage, C/C++ : weiß nicht).

Abbrechbarkeit

Endlos-Abfrageschleifen oder auch langlaufende Funktionen können von außen nicht einfach beendet werden. Das kann zum Crash des Modells führen. Hier ist programmtechnisch vorzusorgen z. B. durch regelmäßige Abfrage der Escape-Taste (GetAsyncKeyState) oder eigene Mechanismen. Im Abbruchfalle sollte CloseInterface aufgerufen werden, um umFish20.DLL sauber zu beenden.

Master-Slave-Betrieb

Sowohl die parallelen (Universal Interface) wie auch das serielle(Intelligent Interface) Interface erlauben einen Master-Slave-Betrieb. Das heißt die Kopplung (siehe Handbuch des jeweiligen Interfaces) eines zweiten (Slave) Interfaces an das erste (an den Rechner angeschlossene Interface, Master). Default ist der Betrieb ohne Slave. Soll mit Slave gearbeitet werden so ist ftDCB.ECount=16 einzustellen.

Einsatz von Treibern für den Betrieb des parallelen Interfaces

Windows NT / 2000 benötigen zum Betrieb des parallelen Interfaces zusätzlich zur FishFace-Software einen Kernel0-Treiber. In diesem Fall WinRT.SYS (zur Installation siehe Readme). Das Einrichten (Festlegen des LPT-Ports, Start des Treibers) des Treibers muß unter Administratorrechten geschehen. Bei privaten Rechnern ist das normalerweise gegeben, im Netzwerk oder auch in Schulen nicht. In diesen Fällen muß der Treiber so eingerichtet werden, daß er beim Booten automatisch gestartet wird. Mit den umFish.DLL-Funktionen Start-/Stop-Driver kann das (mit Administratorrechten) auch im Anwendungsprogramm geschehen.

Unter Windows 95/98/Me kann das parallele Interface ebenfalls mit einem zusätzlichen Kernel0-Treiber (WRTdev0.VxD) betrieben werden. Bei Rechnern ab ca. 500 MHz ist das zu empfehlen, hier das erforderlicher Timing unabhängig vom Rechnertakt ist. Zur Installation siehe Readme.

Lampen am Interface

Die 4 Digitalausgänge des Interfaces sind primär zum Schalten von Motoren in zwei Laufrichtungen vorgesehen. Doch bietet sich zusätzlich die Möglichkeit, Geräte, die nur ein- und ausgeschaltet werden müssen – wie z. B. Lampen an nur einen Pol eines Digitalausganges und die Erde anzuschließen. Auf diese Weise ist es möglich 8 Lampen mit einem Interface zuschalten. Hierfür gibt es die umFish20.DLL-Funktion SetLamp.

Funktionen

ftDCB-Kontrollblock

Die Informationen zum Betrieb des Interfaces werden in einem Kontrollblock gehalten, der vor dem ersten Aufruf von OpenInterface anzulegen ist :

- hCom — Handle zum COM- bzw. LPT-Port, wird bei erfolgreichem Open gesetzt und bei Close auf Null gesetzt, sollte nicht verändert werden.
- PortID — nur LPT, I/O-Adresse des LPT-Ports, von OpenInterface gesetzt, sollte nicht geändert werden.
- LPTDelay — Ausgabeverzögerung an dem LPT-Port, von OpenInterface auf den

  Defaultwert = 10 gesetzt, kann ggf. vor dem Open verändert werden. Schnelle Rechner größerer Wert.
  Wird nur bei LPT1-3 ausgewertet
- LPTAnalog — Skalierung des Analogwertes, von OpenInterface auf den Defaultwert = 5 gesetzt, kann ggf. vor dem Open verändert werden. Der angezeigte Wertebereich sollte zwischen 0 und 1024 liegen, bei zu kleinen Werten ein größeres LPTAnalog wählen.
- ECount — Anzahl E-Eingänge (Default=8, ohne Slave, =16, mit Slave)
- FID — MultiMediaTimer : Handle, sollte nicht geändert werden.
- PollInterval — PollInterval des MM-Timers (milliSekunden). Hier ist Gefühl angesagt, man sollte zunächst mit den Defaultwerten arbeiten. Man kann sie nach dem OpenInterface auslesen.
- AnalogScan — Abfrage auch der Analogeingänge (Default=0, mit=1)
- OutputStatus — Status aller M-Ausgänge.
  Jeweils 2 bit pro Ausgang, das niederwertige bit :
  "Linkslauf" (ftiLinks), das höherwertige bit "Rechtslauf" (ftiRechts).
  "Aus" (ftiAus) beide 0.
  Die bits dürfen nicht gleichzeitig gesetzt sein.
  bit 0/1 : Motor 1 - bit 6/7(14/15) : Motor 4(8).
  Siehe auch Sonderfall Lampen.
- InputStatus — Status aller E-Eingänge.
  bit 0 entspricht Eingang 1, bit 7(15) Eingang 8(16).
- Analogs(0..1) — Werte der A-Eingänge (EX, EY)
- Counters(1..16) — Impuszähler an den E-Eingängen

Der Kontrollblock wird im Anwendungsprogramm (sprachspezifisch, siehe Nutzung) als Struktur mit 32bit-Worten dargestellt. Eine simultaner Einsatz mehrerer Interfaces ist möglich, es müssen dann entsprechend viele Kontrollblöcke angelegt und mit OpenInterface besetzt werden.

Alle Funktionen von umFish20.DLL erwarten als ersten Parameter die Adresse des Kontrollblocks (Ausnahme Start-/StopDriver).

Sonderfall Lampen.

Geräte, die nur einen Ein-/Ausstatus kennen, können einpolig an einen M-Ausgang (zweiter Pol Masse) angeschlossen werden. Sie können dann über den Befehl SetLamp gesteuert werden. Sollen sie gemeinsam (z. B. bei einer Verkehrsampel) geschaltet werden, können

sie auch über OutputStatus geschaltet werden. Dazu sind die entsprechenden einzelnen Lampenbits im OutputStatus zu setzen, sie werden von umFish20 bei der nächsten Abfrage ausgewertet.

Es gibt hier Unterschiede zwischen den Interfaces :
Paralleles Interface : Im nicht geschalteten Zustand sind die Lampen aus. M1 vorderer (gelber) Kontakt : Lampe 1, hinterer (orange) Kontakt Lampe 2 ...
Serielles Interface (30402) : im nicht geschalteten Zustand sind die Lampen **an**. M1 vorderer Kontakt : Lampe 2, M1 hinterer Kontakt Lampe 1 ...

Also genau hinsehen. Beim seriellen Interface ergibt sich der etwas irritierende Effekt, dass alle Lampen im Ruhezustand eingeschaltet sind. Außerdem ist zu bedenken, dass mit SetMotors stets **alle** M-Ausgänge geschaltet werden.

Befehle

Allgemein : der Return Code inputstatus liefert im positiven Fall den aktuellen Wert des InputStatus, wenn der Befehl fehlgeschlagen ist den Code ftiFehler.

umOpenInterface

[inputstatus =] umOpenInterface(ftDCB, PortName)

Öffnen der Verbindung zum Interface, besetzen des Kontrollblocks. PortName (string) : COMx bzw LPT1 – 3. . Bei Windows NT / 2000 muß beim parallelen Port LPT angegeben werden. Bei Windows 95/98/Me kann LPT angebeben werden (wenn LPT1-3 nicht "funktioniert"). In beiden Fällen muß dann WinRT.SYS bzw. WRTdev0.VxD installiert sein, siehe Readme.

umCloseInterface

[inputstatus =] umCloseInterface(ftDCB)

Schließen der Interfaceverbindung, freigeben des Ports.

umVersion

longinteger = umVersion

Numerischer Wert für die aktuelle Version (z. B. 201 für Version 2.01)

umGetInput

bool = umGetInput(ftDCB, InputNr)

Abfrage des ausgewählten E-Eingangs. True : Eingang ist geschaltet. Abgefragt werden können neben dem Standard Taster auch weitere Quellen wie z. B. die Photodiode, die ab einem bestimmten (zu erprobenden) Lichtwert auf True schaltet.

umSetMotor

[inputstatus =] umSetMotor(ftDCB, MotorNr, Richtung)

Schalten eines M-Ausganges. Richtung ftiLinks, ftiRechts oder ftiAus.

umSetLamp

[inputstatus =] umSetLamp(ftDCB, LampNr, OnOff)

Schalten eines ”halben” M-Ausganges. Das Gerät (Lampe, Magnet) wird mit einem Pol des Ausganges und der Erde verbunden. LampNr 1 – 8(16) entsprechend M1-M4(8), M1 (rot) ist L1. OnOff = ftiEin bzw. ftiAus

---------- nur Windows NT, paralleles Interface ----------

umStartDriver

[res =] StartDriver()

Starten des WinRT.SYS Treibers. Die Funktion entspricht der Menü-Funktion Start | Einstellungen | Systemsteuerung | Geräte | WinRT. WinRT.SYS muß für manuellen Start installiert sein. Es sind Administratorrechte erforderlich.

umStopDriver

[res =] StopDriver()

Anhalten des WinRT.SYS Treiber. Sonst wie StartDriver.

Defaultmäßig wird WinRT.SYS mit ”Allow Conflicts” installiert. Ein sonst erforderliches Anhalten der normalen (ParPort) Treiber ist dann nicht erforderlich. Die Funktionen sind sinnvoll, wenn man nur gelegentlich mit dem Interface arbeitet und nicht ständig den zusätzlichen Treiber mitlaufenlassen will.

Befehlsäquivalente

Die Befehle umGetInput, umSetMotor und umSetLamp greifen nicht direkt auf das Interface zu, sondern fragen/setzen den entsprechenden Status in ftDCB ab und maskieren ihn, sie sind also reine Komfort Funktionen. Die umFish-Befehle GetInputs, SeMotors, ClearMotors und GetAnalog lassen sich 1:1 auf einen ftDCB Zugriff abbilden und werden deswegen nicht mehr angeboten :

GetInputs	ftDCB.InputStatus
SetMotors	ftDCB.OutputStatus(=xxxx)
ClearMotors	ftDCB.OutputStatus=0
GetAnalog	ftDCB.Analogs(n)

Impulszähler

Alle Impulse auf E-Eingänge (Wechsel von Status Ein zu Aus und umgekehrt) werden in ftDCB.Counters(n) laufend gezählt. Sie können jederzeit abgefragt werden. Sie sollten vor Beginn eines Auftrages an das Modell auf Null zurück gesetzt werden. Typischer Einsatzfall ist die Positionsbestimmung bei Robots.

Programmiertechniken

Hier werden Programmiertechniken zur Lösung typischer Aufgabenstellungen weitgehend unabhängig von einer Programmiersprache (aber doch in der Nähe von Visual Basic) skizziert.

Programmrahmen

umFish20 benötigt pro Interface jeweils einen statischen Kontrollblock (der parallele Betrieb von Interfaces – also nicht Master/Slave – ist problemlos möglich. Die Konstellation LPT1 mit parallelem Interface (Master/Slave) und COM1 mit Intelligent Interface (Master/Slave) erlaubt schon den simultanen Betrieb von vier Interface an einem normal ausgestatteten PC.

Initialisieren des Interface durch OpenInterface. Vorher ggf. Parameteranpassungen :
ftDCB.AnalogScan=1 (Default=0) mit Scannen der Analog-Eingänge
ftDCB.ECount=16 (Default=8) mit Slave.
Beim parallelen Interface ggf. auch noch LPTDelay/LPTAnalog, siehe OpenInterface

Beenden des Interface-Betriebs und Freigabe der Schnittstelle durch CloseInterface.

Der Modellbetrieb selber wird häufig in einem Programmzyklus erfolgen. Man sollte dabei für die Unterbrechbarkeit (siehe Abschnitt Hinweise) sorgen und zusätzlich eine NotAus-Funktion zur Gewährleistung der Abbrechbarkeit (Hinweise) vorsehen.

Warten auf Taster

```
Do
  DoEvents                          ' --- Unterbrechbarkeit
Loop Until umGetInput(ftDCB, 1) ' --- Ende, wenn Taster 1 gedrückt
```

Anzeige von Analogwerten

```
Do
  DoEvents                        ' --- Unterbrechbarkeit
  Display ftDCB.Analogs(0)        ' --- Anzeige EX
Loop Until (GetAsyncKeyStatus(VK_ESCAPE) <> 0) ' --- Abbrechbarkeit
```

NotHalt/Reset

```
umSetMotor(ftDCB, 1, ftiLinks) ' --- Motoren starten
Do
  ....
  DoEvents                        ' --- Unterbrechbarkeit
Loop Until umGetInput(ftDCB, 8) ' -- NotHalt
ftDCB.OutputStatus = 0          ' --- Alle Motoren (M-Augänge) aus
```

Fahren zum Endtaster

```
umSetMotor(ftDCB, m, ftiLinks)  ' --- Motor starten
Do
  DoEvents                          ' --- Unterbrechbarkeit
Loop Until umGetInput(ftDCB, d) ' --- Ende bei Taster d
umSetMotor(ftDCB, m, ftiAus)    ' --- Motor aus
```

Anfahren einer Position

Ab hier wird es komplizierter, für Visual Basic oder Delphi Programmierer lohnt der Umstieg auf umFish20Ex bzw. FishFace (FishFa50.OCX / FishFa50.DCU). VC++ Programmierer können die entsprechenden Quellen mal durchsehen.

Nutzung

Visual Basic

Der CodeModul umFish.BAS mit folgenden Deklarationen sollte in das Projekt eingebunden werden :

```
Public Declare Function umOpenInterface Lib "umFish20.dll" _
  (ft As ftDCB, ByVal ComName$) As Long
Public Declare Function umCloseInterface Lib "umFish20.dll" _
  (ft As ftDCB) As Long
Public Declare Function umGetVersion Lib "umFish20.dll" () As Long
Public Declare Function umGetInput Lib "umFish20.dll" _
  (ft As ftDCB, ByVal InputNr&) As Boolean
Public Declare Function umSetMotor Lib "umFish20.dll" _
  (ft As ftDCB, ByVal MotorNr&, ByVal Richtung&) As Long
Public Declare Function umSetLamp Lib "umFish20.dll" _
  (ft As ftDCB, ByVal LampNr&, ByVal OnOff&) As Long

Public Declare Function umStartDriver Lib "umFish20.dll" () As Long
Public Declare Function umStopDriver Lib "umFish20.dll" () As Long

Public Declare Function GetAsyncKeyState Lib "user32" (ByVal vKey As
Long) As Integer
Public Const VK_ESCAPE = &H1B
Public Declare Sub Sleep Lib "kernel32" (ByVal dwMsec&)
Public Declare Function GetTickCount Lib "kernel32" () As Long

Public Type ftDCB
  comI As Long                  ' --- PortHandle / OpenFlag (=0)
  PortID As Long                '     IO-Adresse bei LPT
  LPTDelay As Long              '     LPT Verzögerungsfaktor
  LPTAnalog As Long             '     LPT Analogfaktor
  ECount As Long                '     Anzahl E-Eingänge

  FID As Long                   ' --- MultiMediaTimer Handle
  PollInterval As Long          '     PollInterval des MM-Timers
  AnalogScan As Long            '     Pollen auch der
Analogeingänge (0/1)

  OutputStatus As Long          ' --- Status der M-Ausgänge
  InputStatus As Long           '     Status der E-Eingänge
  Analogs(0 To 1) As Long       '     Werte der A-Eingänge
  Counters(1 To 16) As Long     '     Impulszähler an den E-
Eingängen
End Type

Public Const ftiFehler = &H1FFFF
Public Const ftiAus = 0
Public Const ftiEin = 1
Public Const ftiLinks = 1
Public Const ftiRechts = 2
```

```
Public ftiDCB As ftDCB, ft As ftDCB
```

```
DoEvents
```

sollte an geeigneten Stellen in den Code eingefügt werden um die Unterbrechbarkeit der Anwendung in engen Abfrageschleifen (z.B. warten auf Endtaster) zu gewährleisten.

Die True bzw False Werte von GetInputs entsprechen den VisualBasic Konventionen.

```
If GetAsyncKeyState(VK_ESCAPE) <> = 0
```

sollte an geeigneten Stellen in den Code eingefügt werden um die Abbrechbarkeit der Anwendung in engen Schleifen oder bei langlaufenden Befehlen zu gewährleisten.

Visual C++

In den Workspace des Projektes sollte die Source umFish.h und das Link-File umFish.lib eingefügt werden. umFish.lib muß auch bei den Linker-Optionen des Projektes eingetragen werden :

```
typedef struct {
  HANDLE hCom;                  // --- PortHandle / Openflag ( = 0)
  DWORD PortID;                 //     IO-Adresse bei LPT
  DWORD LPTDelay;               //     LPT Verzögerungsfaktor
  DWORD LPTAnalog;              //     LPT Analogfaktor
  DWORD ECount;                 //     Anzahl E-Eingänge (8/16)
   DWORD FID;                    // --- MultiMediaTimer Handle
   DWORD PollInterval;           //     Pollinterval des MM-Timers
   DWORD AnalogScan;             //     Pollen auch der Analogeingänge
(0/1)
   DWORD OutputStatus;           // --- Status der M-Ausgänge
   DWORD InputStatus;            //     Status der E-Eingänge
  DWORD Analogs[2];             //     Werte der A-Eingänge
   DWORD Counters[16];           //     Impulszähler an den E-
Eingängen
} ftiDCB;

DWORD __stdcall umOpenInterface(ftiDCB &Interface, LPCSTR ComName);
DWORD __stdcall umCloseInterface(ftiDCB &Interface);
DWORD __stdcall umGetVersion();
BOOL  __stdcall umGetInput(ftiDCB &Interface, DWORD InputNr);
DWORD __stdcall umSetLamp(ftiDCB &Interface, DWORD LampNr, DWORD
OnOff);
DWORD __stdcall umSetMotor(ftiDCB &Interface, DWORD MotorNr, DWORD
Direction);
DWORD __stdcall StartDriver();
DWORD __stdcall StopDriver();

const DWORD ftiAus    = 0;
const DWORD ftiEin    = 1;
const DWORD ftiLinks  = 1;
const DWORD ftiRechts = 2;

const DWORD ftiFehler = 0x0001FFFF;
```

Anwendungsbeispiel : umFish20VC.cpp als einfache Console-Application.

ACHTUNG : umFish.DLL ist unabhängig von einer Version von VC++, umFish20.LIB dagegen hat bei verschiedenen Versionen ein unterschiedliches Format. Mitgeliefert wird umFish.LIB für VC++ 6.0 (weiterer Formate ggf. beim Autor). Für den Borland C++Builder sollte umFish20Load.h eingesetzt werden (s.u.).

VC++6.0 deutsch :

- Datei | Neu | Projekte | Konsolanwendung : Name und Verzeichnis umFish20VC, leeres Projekt, Fertigstellen
- Manuelles kopieren der nochfolgenden Dateien in das neu angelegte Projektverzeichnis
- Projekt | Dem Projekt hinzufügen | Dateien : umFish20.h, umFish20.lib, umFish20VC.cpp
- Erstellen : umFish20VC.EXE (F7)
- Erstellen | Ausführen von umFish20VC.EXE (CTL+F5)

Anmerkung 1 :

umFish20.LIB (VC++ 5.0) wird aus umFish20.LIB (VC++6.0) wie folgt (unter DOS / Konsole) erstellt :

LIB /CONVERT /LINK50COMPAT /OUT:umFish50.LIB umFish.LIB

dabei sollten alle Beteiligten am besten im gleichen Verzeichnis liegen (das spart Pfadangaben): Lib.EXE, Link.EXE, MSPDB60.DLL (aus C:\Programme\Visual Studio ...) und umFish20.LIB

Anmerkung 2 :

umFish20.DLL kann auch dynamisch geladen werden. Sinnvoll, wenn umFish20.lib nicht eingesetzt werden kann, z. B. bei Borland C++Builder.

Dazu ist anstelle von umFish20VC.h umFish20Load.h und anstelle von umFish20VC.cpp umFish20L.cpp zu verwenden. umFish20.lib entfällt. Projekterstellung unter VC++ 6.0 sonst wie oben

Delphi

Einbinden folgender Deklarationen z.B. im Interfaceteil der zu einer Form gehörenden Unit :

```
const
  ftiAus=0; ftiEin=1; ftiLinks=1; ftiRechts=2;
  ftiFehler=$0001FFFF; ftiTrue=-1; ftiFalse=0;

type TftiDCB = record
  hCom:          LongInt;          // --- PortHandle / Openflag ( = 0)
  PortID:        LongInt;          //     IO-Adresse bei LPT
  LPTDelay:      LongInt;          //     LPT Verzögerungsfaktor
  LPTAnalog:     LongInt;          //     LPT Analogfaktor
  ECount:        LongInt;          //     Anzahl E-Eingänge (8/16)

  FID:           LongInt;          // --- MultiMediaTimer : Handle
  PollInterval: LongInt;           //     Pollinterval des MM-Timers
  AnalogScan:    LongInt;          //     Pollen auuch der
                                   //     Analogeingänge (0/1)

  OutputStatus: LongInt;           // --- Status der M-Ausgänge
  InputStatus:  LongInt;           //     Status der E-Eingänge
  Analogs:      array[0..1] of LongInt; //  Werte der A-Eingänge
  Counters:     array[1..16] of LongInt; // Impulse an den
                                          // E-Eingängen
  end;

var
  ftiDCB: TftiDCB;

function umOpenInterface(var ft: TftiDCB; PortName: string):LongInt;
       stdcall; external 'umFish20.dll';
function umCloseInterface(var ft: TftiDCB):LongInt;
       stdcall; external 'umFish20.dll';
function umGetVersion:LongInt;
       stdcall; external 'umFish20.dll';
function umGetInput(var ft: TftiDCB; InputNr:LongInt):LongInt;
       stdcall; external 'umFish20.dll';
function umSetMotor(var ft: TftiDCB; MotorNr,
Direction:LongInt):LongInt;
       stdcall; external 'umFish20.dll';
function umSetLamp(var ft: TftiDCB; LampNr, OnOff:LongInt):LongInt;
       stdcall; external 'umFish20.dll';
function umStartDriver:LongInt; stdcall; external 'umFish20.dll';
function umStopDriver:LongInt;  stdcall; external 'umFish20.dll';
```

Siehe auch die Source umFish20.PAS.

```
Application.ProcessMessage;
```

sollte angeeigneten Stellen in den Code eingefügt werden um die Unterbrechbarkeit der Anwendung in engen Abfrageschleifen (z.B. warten auf Endtaster) zu gewährleisten.