fischertechnik Interface

umFish20.DLL

Manual for version 2.0

Ulrich Müller

Inhaltsverzeichnis

Dokumentname : umFish20E.doc. / Fisch6.WMF Druckdatum : 26.09.2004

umFish20.DLL

General

umFish20.DLL is a DLL written in VC++ 6.0 (32bit). It enables access to the fischertechnik interfaces 30520, 30566 and the CVK (Cornelsen Interface (parallel, use in master/slave combination is possible) and 30402 (serial and Extension Module 16 554). Therefore some basic functions are supplied to be used from the programming languages Visual Basic, C/C++ and Delphi.

umFish20.DLL is a basis for development of own more complex funtions. Alternatively you can use umFish20Ex.BAS / umFish20Ex.PAS and FishFa50.OCX for advanced programming. Both are using umFish20.DLL. The parameter list of umFish20.DLL is designed to be used for using this basis functions from different programming languages. It is tested with VB6, VC++6.0 / 5.0, CBuilder and Delphi4. Additional versions of umFish20.DLL are mswFish.DLL for use with MSWLogo and javaFish.DLL for use with Java (SDK 1.2.2 ...)

umFish20.DLL will run with Windows95/98/Me and Windows NT4/2000. umFish20.DLL may located, as usual in Windows, in the application path or in Windows\system32. It needs no registration. Running a parallel Interface under Windows NT/2000 the kernel 0 Driver WinRT.SYS (included) must be copies to WinNT\System32\Driver and registered by doubleclicking on WinRTSYS.REG. You can change some parameters according your wishes :
”AllowConflict” = dword:1 (true to 0 false, in that case you must stop the printer driver)
”IoPortAddress” = dword:378 (LPT1) be be changed to 278 or 3BC (LPT 2/3)
”Start” = dword:00000003 means manual start (SystemControl | Devices) me be changed to 00000002 (start with boot). For Windows 95/98/Me you can use WRTdev0.VxD and WinRTVxD.REG.

umFish20.DLL is an extended version from umFish.DLL not compatible to umFish.DLL. The ftiDCB control block has a new structure, the impulse counter is new. umFish20.DLL now comes with a polling concept (the MultiMediaTimer is used). That means the interface is scanned (polled) in short intervals (typical 10 msec) for its inputs (E. and EX/EY) and the M-exits are refreshed. In this way a more exact control of a robots position is possible (especial in connection with the impulse counters). More "simple" application can be operated with umFish.DLL further on because of the handling of umFish.DLL is simplier.

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

Notes

Refresh

fischertechnik interfaces need to have the state of the (motor)outputs refereshed every 300 msec. Otherwise all motor outputs will be switched off. This is done to protect models from confused programs. umFish20.DLL is polling the interfaces in shorter intervals, that means this function is pratically disabled.

Interface Parameters

umFish20.DLL uses a control block (ftiDCB) for communication with the interface. OpenInterface will initialize it with common values. Normaly there is no need to change them. That means : no slave interface (ECount = 8, with slave = 16), no scan of EX/EY (AnalogScan = 0, with scan = 1).

If you encounter strange behaviour of the LPT-interface (with fast computers and Windows 95/98/Me), try other (greater) values for LPTDelay and LPTAnalog. The better way is using the special driver WRTdev0.VxD.

The modification of interface parameters should be done before using OpenInterface.

Interrupts

When operating the interface in a small loop (e.g. looking for the input state), the program is no longer interruptable. This means that it no longer reacts to control buttons and the screen is usually not updated. In this case you must share the timeslice of the process so that the program is interruptable at a later point: (VB : DoEvents, Delphi : Application.ProcessMessage, C/C++ : ?). The umFish commands themselves are monitored by an internal timer and will be aborted with an error message if necessary.

Master Slave Operating

The Universal (parallel) and the Intelligent (serial) Interface can be extented to double the capacity by an second Interface (Slave) connected to the first (Master, connected with the computer). The parallel Interface always opererated in a internal Master-Slave Mode, the serial Interface must have set ECount = 16 (default = 8, no extension).

Using drivers to operate the parallel interface

Windows NT/2000 need for running the parallel interface in addition to umFish.DLL the kernel0 Driver WinRT.SYS (for installation look readme). The installation (copy WinRTSYS.REG, and modify it) always must be done with administrator rights. For private use in most cases you can run it with administrator rights to. But in networks or with school computers there a no admin rights to run it. In that case WinRT.SYS must be registered vor starting with boot. Operating under administrator rights Start/StopDriver can be used to activate the driver.

With Windows 95/98/Me the parallel Interface can be operated more save with an additional Kernel0-driver (WRTdev0.VxD). With computer with 500 MHz and more it should be done.

Using lamps with the parallel Interface

The 4 digital outputs of the parallel interface are constructed for running motors (right, left, stop) – like the serial interface. In addition on the parallel interface you can install double the number of lamps (or other on/off devices) on connection only one contact of the digital output and second the ground with a lamp (counted M1 red = 1 to M4(8) = 8(16)). For switching you can use the umFish20.DLL function SetLamp.

Data and Functions

ftDCB control block

The data for operating the interface are stored in a control block. The control block must be declared before using umFish20.DLL. The control block contains the following fields :

• hCom	Handle of the actual COM or LPT port. Set with successful OpenInterface and should not be changed. Set to 0 with CloseInterface.
• PortID	Used only with LPT : I/O address of the LPT port. Set by OpenInterface and shoud not be changed.
• LPTDelay	LPT port : Output delay. OpenInterface sets it to 10 (Pentium 400 ore less). On faster computers greater values may be useful. Windows 95/98/Me only, needs portname LPT1-3.
• LPTAnalog	LPT port : scaling the analog value. OpenInterface setzs it to 5. May be changed to greater values. The displayed values should be in the range 0 – 1024. Depends of the LPTDelay value.
• ECount	Number of E-Inputs (default = 8 (without slave) = 16 with slave).
• FID	MultiMediaTimer : Handle, should not be changed.
• PollInterval	Interval (msec)for polling the interface, used by the MultiMediaTimer. Changes must be done cautious. Try it first with the default value. Can be read after OpenInterface.
• AnalogScan	Scanning the analog inputs (EX/EY) to (default = 0, with = 1).
• OutputStatus	State of all (motor, M) outputs. The bits are divided into 2-bit groups: the low bit means left turn (ftiLinks); the high bit means right turn (ftiRechts). Both bits may not be set simultaneously. Both = 0 (ftiAus) means stop. Bit 0/1 : motor 1 (M1), bit 6/7 : motor 4 (M4) (bit 14/15 : motor 8). Special case : Lamps.
• InputStatus	State of all E inputs (E.). Bit 0 = E1, bit 7 = E8 (bit15 = E16).
• Analogs(0..1)	Values of the analog inputs (EX, EY)
• Counters(1..16)	Impuls counters : number of changes of the E inputs since last reset or OpenInterface

The control block is a structure with 32bit words. They are consecutive. The language specific presentation differs slightly.

All umFish20.DLL function have the address of ftDCB as first parameter (not Start/StopDriver, umVersion).

Using Lamps.

Peripherals which only have an on/off state, can be connected only with one pin to a M output (second pin to mass). They can be operated with the function SetLamp. If all "lamps" should be controlled at a time (e.g. traffic lights), it can be done via setting OutputStatus. There for the single M bits must be set, the are switched with next polling cycle.

There are differences between the interfaces :
Parallel interface : state M off the "lamps" are off, front row lamp 1 ..., back row lamp 2 ...
Serial interface : state M off the "lamps" are ON. That means OutputStatus = 0 : all "lamps ar ON. front row lamp 1..., back row lamp 2...

Functions

General : the return code inputstatus contains the actual value of InputStatus, if failed the return code ftiFehler.

umOpenInterface

[inputstatus =] umOpenInterface(ftDCB, PortName)

Open a connection to an interface and setting default values to the control block (values are set explicitely before OpenInterface are saved). PortName (ANSI string) : COM1-8 (serial interface), LPT(parallel interface) or LPT1-3. LPT means using the correponding driver.

umCloseInterface

[inputstatus =] umCloseInterface(ftDCB)

close the interface connection, free the resources.

umVersion

longinteger = umVersion

numercal value for the actual version (e. g. 201 for version 2.01)

umGetInput

bool = umGetInput(ftDCB, InputNr)

State of the specified E input (true means : is on). Beside the standard switches it is possible to operate some more sources.

umSetMotor

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

Set one M output (MotorNr) and Direction (ftiLinks (left), ftiRechts(right) and ftiAus(off)).

umSetLamp

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

Set of a "half" m output. The device (lamp, magnet) is only connected with one pin, other is mass. lamp no 1 – 8(16).

------------ Windows NT/2000 only, parallel interface --------------------

umStartDriver

[res =] StartDriver()

Start the WinRT.SYS driver. This function corresponds to the device control function of Windows. Administrator rights required.

umStopDriver

[res =] StopDriver()

Stop the WinRT.SYS driver.

WinRT.SYS is installed per default with "Allow Conflicts". The ParPort must not be stopped in this case, but not printing should be done.

Equivalent functions

The funcitons umGetInput, umSetMotor und umSetLamp not realy operate on the interface, they are only ask/set the status words of ftDCB, they are pure comfort functions, they can be substituded by asking InputStatus or setting OutputStatus and masking it.

The old umFish functions GetInputs, SeMotors, ClearMotors and GetAnalog can be mapped to the ftiDCB and are nor more suported :

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

Impulse Counter

All impulses to the E inputs (change from state true to false and vice versa) are counted continously : ftDCB.Counters(n). n = 1 correspondeds with E1 ... (with VC++ : n = 0). They can be accessed every time or set to 0 at the beginning of a more complex operation. The are mostely used to control the position of an robot.

Programming techniques

Describes programming techniques for solving typical problems. The descriptions are independent of any particular programming language, but similar to Visual Basic. Look also to the language specific sources delivered with this manual.

Program frame

umFish20.DLL needs for each interface connected a special ftDCB control block. You can operate up to 4 (master) interfaces and additional 4 slave interfaces connected to them. The configuration LPT1 with a parallel interface (master/slave) and COM1 with Intelligent Interface (master/slave) already enables a simultan running of four interface on a normal configured PC.

Initialize the interface with umOpenInterface. Modifications of the ftDCB values should be done before : ftDCB.AnalogScan = 1 (Scanning of EX/EY)
ftDCB.ECount = 16 (Slave connected). In some cases (parallel interface) ftDCB.LPTDelay / LPtAnalog.

Close the interface operating : umCloseInterface.

Operating a model there will be often loops. In that case you should take care to make it interruptable (DoEvents, Application.ProcessMesage) and have in addition an emergency exit.

Wait for a switch

```
Do
  DoEvents                        ' --- interruptable
Loop Until umGetInput(ftDCB, 1) ' --- ends if E1 true
```

Show analog values

```
Do
  DoEvents                      ' --- interruptable
  Display ftDCB.Analogs(0)      ' --- show EX
Loop Until (GetAsyncKeyStatus(VK_ESCAPE) <> 0) ' --- abort
```

Emergency Off

```
umSetMotor(ftDCB, 1, ftiLinks) ' --- start motors
Do
  ....
  DoEvents                      ' --- interruptable
Loop Until umGetInput(ftDCB, 8) ' -- abort
ftDCB.OutputStatus = 0         ' --- all M outputs off
```

Go to end switch

```
umSetMotor(ftDCB, m, ftiLinks)  ' --- start motors
Do
  DoEvents                       ' --- interruptable
Loop Until umGetInput(ftDCB, d) ' --- ends when Ed true
umSetMotor(ftDCB, m, ftiAus)    ' --- motor off
```

Go to a special position

That is a more complicated job. Look to umFish20EX.BAS / PAS for functions like WaitForChange or to FishFa50.OCX(fishfa50.zip) FishFaD4.DCU (umfishu.zip)

Using umFish20.DLL

Visual Basic

Include the code module umFish20.BAS in your project :

```
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-address LPT only
  LPTDelay As Long               '     LPT delay factor
  LPTAnalog As Long              '     LPT analog factor
  ECount As Long                 '     number of E inputs (8 / 16)

  FID As Long                    ' --- MultiMediaTimer Handle
  PollInterval As Long           '     PollInterval MM-Timer
  AnalogScan As Long             '     Scan EX/EY (0/1)

  OutputStatus As Long           ' --- Status M outputs
  InputStatus As Long            '     Status E inputs
  Analogs(0 To 1) As Long        '     values EX/Ey
  Counters(1 To 16) As Long      '     impulse counter on E inputs
End Type

Public Const ftiFehler = &H1FFFF  ' - Error
Public Const ftiAus = 0           '   off
Public Const ftiEin = 1           '   on
Public Const ftiLinks = 1         '   left
Public Const ftiRechts = 2        '   right

Public ftiDCB As ftDCB, ft As ftDCB
```

```
DoEvents
```

Should be included on appropriate locations to make the program interruptable.

True/False values are VisualBasic standard.

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

Should be included on appropriate locations to have abort points in the program.

Visual C++

The Workspace of the project should contain the source umFish20.h and the link file umFish20.lib. umFish20.lib must olse be added to the linker options. Alternatively you can use umFish20Load.h. In this case no umFish20.lib is needed (specially for use with CBuilder), the function LoadFish / FreeFish must be used in addition.

```
typedef struct {
  HANDLE hCom;                 // --- PortHandle / Openflag ( = 0)
  DWORD PortID;                //     IO address LPT only
  DWORD LPTDelay;              //     LPT delay factor
  DWORD LPTAnalog;             //     LPT analog factor
  DWORD ECount;                //     number of E inputs (8/16)
  DWORD FID;                   // --- MultiMediaTimer Handle
  DWORD PollInterval;          //     Pollinterval MM-Timers
  DWORD AnalogScan;            //     Scan EX/EY (0/1)
  DWORD OutputStatus;          // --- Status M outputs
  DWORD InputStatus;           //     Status E inputs
  DWORD Analogs[2];            //     Values EX/EY
  DWORD Counters[16];          //     Impulse counter E inputs
} 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;     //   off
const DWORD ftiEin    = 1;     //   on
const DWORD ftiLinks  = 1;     //   left
const DWORD ftiRechts = 2;     //   right

const DWORD ftiFehler = 0x0001FFFF;  //  error
```

Example : umFish20VC.cpp as a simple Console-Application.

Attention : umFish20.lib has the lib-format of VC++ 6.0, it can not be used with any compiler, in that case use umFish20Load.h and LoadFish / FreeFish without umFish20.lib (tested with VC++5.0, VC++4.2 and CBuilder).

Setup of a new console project :

- new Workspace : Console
- Insert umFish20VC.h, umFish20.lib, umFish20VC.cpp
- Build | Settings | LInk : declare umFish20.lib
- compile (F7)
- copy umFish20.DLL to \Debug or \Release, or alternatively to \Windows\System

Delphi

Insert the following declarations in the interface part (e.g.) of a 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 address LPT only
  LPTDelay:     LongInt;         //     LPT delay factor
  LPTAnalog:    LongInt;         //     LPT analog factor
  ECount:       LongInt;         //     number of E inputs (8/16)

  FID:          LongInt;         // --- MultiMediaTimer : Handle
  PollInterval: LongInt;         //     Pollinterval MM-Timer
  AnalogScan:   LongInt;         //     Scan EX/EY (0/1)

  OutputStatus: LongInt;         // --- Status M outputs
  InputStatus:  LongInt;         //     Status E inputs
  Analogs:      array[0..1] of LongInt; //  values EX/EY
  Counters:     array[1..16] of LongInt; // impulse count E inputs
  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';
```

see also source umFish20.PAS.

```
Application.ProcessMessage;
```

Should be inserted on adequate positions in the code to guarantee that the program ist interruptable.