Using umFish40.DLL and ROBO Interfaces with

MSWLogo

Ulrich Müller

Contents

umFish40.DLL Functions **3**

Common 3
Functions 4

Samples **6**

ROBO Starter Kit : first motor program 6
ROBO Starter Kit : Hand Dryer 8
ROBO Starter Kit : Traffic Lights 9
ROBO Starter Kit : Temperature Control 10
Tower of Hanoi 11

umFish40.DLL Functions

Common

For programming the fischertechnik ROBO Interfaces you need the umFish40.DLL www.ftcomputing.de/zip/umFish41.zip and the english documentation www.ftcomputing.de/pdf/umFish41e.pdf

The functions of umFish40.DLL are capsulated by using the dllcall functions of MSWLogo. No error control is done (but it is possible to do it by controling the return code, if there are none the dllcall must be changed from "v to "l). umFish40.DLL must be positioned in the system directory ..\WinNT\System32 or in the directory of logo.exe from MSWLogo. umFish40.DLL all time must be loaded because to save internal data areas. That means no additional dllcalls can be done if umFish40.DLL is running.

The functions are contained in the file mswFish40 which must be postioned in the directory ..\LogoLib of logo.exe.

Test is done with version 6.5b of MSWLogo. In this version only the first ROBO Interface (or I/O Extension, RF Datalink) found at USB is supported.

For setting the M- and O-Outputs there are variables : :rbLeft, :rbRight, :rbOff, :rbOn

Functions

rbOpenInterface

Loading umFish40.DLL and open the first ROBO Interface found at USB. Setting a rbHandle for internal use (if correctly opened 0, not opened –1, errorneous open –536870911).

rbCloseInterface

Closing the Interface connection, free umFish40.DLL

rbSetLamp :LampNr :OnOff

Set an O-Output to :rbOn or :rbOff, Power is internal set to 7

rbSetLampEx :LampNr :OnOff :Power

Set an O-Output to :rbOn or :rbOff, Power must be set 0 .. 7

rbSetMotor :MotNr :Dir

Set an M-Output to :rbLeft, :rbRight or :rbOff, Speed is always 7

rbSetMotorEx :MotNr :Dir :Speed

Set an M-Output to :rbLeft, :rbRight or :rbOff, Speed to 0 .. 7

rbSetMotors :MotorStatus

Set the state of all M-Outputs or O-Outputs to a direction (2bits : 0 off, 1 left, 2 right), Speed is 7, Normal mode.
For O-Output 1bit 0 off, 1 on.

rbSetMotorsEx :MotorStatus :SpeedStatus :SpeedStatus16

Like rbSetMotors, but setting speed to ((4bit, 0 .. 7)

rbGetMotors

Get the state of all M- (2bit 0 off, 1 left, 2 right) or O-Outputs (1bit, 0 off, 1 on)

rbRobMotor :MotNr : Dir :Speed :ICount

Start a Robmotor (M-Output an fix End- and Impulse-Switches, M1 I1 I2 .. M4 I7 I8). The motor runs until the ICount-Counter (same number as Impulse Switch) comes to 0.

rbRobMotors :MotorStatus :SpeedStatus :SpeedStatus16 :ModeStatus

Set all M-Outputs. RobMotors need to be set the ICount (rbSetCounter(:Impuls) before. MotorStatus 2bit 0 off, 1 left, 2 right. SpeedStatus 4 bit 0 .. 7. ModeStatus 2bit 0 Normal, 1 RobMode.

rbGetModeStatus :MotNr

Get the ModeStatus of an M-Output (0 Normal, 1 RobMode)

rbSetModeStatus :MotNr :Mode

Set the ModeStatus of an M-Output (0 Normal, 1 RobMode)

rbGetInput :InputNr

Get the state of an I-Input ("true "false)

rbGetInputs

Get the state of all I-Inputs (1bit 0 off, 1 on)

rbGetAnalog :AnalogNr

Get the actual value (0 .. 1023) of an A-Input (AX = 1, AY = 2, AX ex = 3 ..

rbGetIRKey :Code :KeyNr

Get the actual value of a key of the IR sender

rbGetVoltage :VoltNr

Get the actual value (Volt/100, 0 .. 999) of an A-Input (A1 = 1, A2 = 2, AV = 3)

rbClearCounters

Clear (= 0) all Counter

rbGetCounter :CounterNr

Get the actual value of an counter

rbSetCounter :CounterNr :ICount

Set a counter to the value of :ICount

rbPause :mSek

Pause a running program for mSek milliseconds

rbShowStatus

Print the most interesting Interface data

Samples

ROBO Starter Kit : first motor program

StarterMotor.LGO : M3 Motor, I1, I5 Switch

Version aaa : all in one

Starts the motor if I1 is pressed an ends if I5 is pressed and released

```
to aaa
  mswFish40
  rbOpenInterface
  if not (:rbHandle < 0) [pr [Gestartet, Taste 1 drücken]]
  while [not (rbGetInput 1)] []
  rbSetMotor 3 1
  pr [Zum Beenden Taste 5 drücken und loslassen]
  while [(rbGetInput 5)] []
  while [not (rbGetInput 5)] []
  rbSetMotor 3 0
  rbCloseInterface
  pr "Beendet
end
```

mswFish40 loads the procedures of mswFish40 in ..\logo\LogoLib

rbOpenInterface loads umFish40.DLL and opens the first ROBO Interface on USB, if :rbHandle is less than 0 an error occurs while opening

Version bbb : with seperate procedures

```
to bbb
  pr [Taste 1 drücken]
  WaitForInput 1
  rbSetMotor 3 :Left
  pr [Zum Beenden Taste 5 drücken und loslassen]
  WaitForHigh 5
  rbSetMotor 3 :Off
  pr "FINIS
end

to Init
  mswFish40
  rbOpenInterface
  ifelse :rbHandle < 0 [pr [Open NICHT erfolgreich]] ~
                       [pr [Open erfolgreich]]
  make "Left 1
  make "Right 2
  make "Off 0
end

to WaitForHigh :InputNr
  while [rbGetInput :InputNr] []
  while [not (rbGetInput :InputNr)] []
end

to WaitForInput :InputNr
  while [not (rbGetInput :InputNr)] []
end
```

Init : to load mswFish40 and OpenInterface first and only once, in this version is no rbCloseInterface.

WaitForInput : Waiting for an Input gets true

WaitForHigh : Waiting an Input gets false and than true

ROBO Starter Kit : Hand Dryer

StarterTrockner.LGO : M1 Motor, M2 Lamp, I1 Light switch, I8 switch

Light barrier on, endless loop, finish by pressing I8, in the loop if light barrier broken : motor runs for 5 secs.

```
to aaa
  pr [Händetrockner gestartet]
  rbSetMotor 2 :rbOn
  rbPause 1000
  while [not Finish] [ ~
    if not rbGetInput 1 [rbSetMotor 1 :rbLeft rbPause 5000 ~
                         rbSetMotor 1 :rbOff]]
  rbSetMotor 2 :rbOff
  pr "FINIS
end

to Finish
  output rbGetInput 8
end

to Init
  mswFish40
  rbOpenInterface
  ifelse :rbHandle < 0 [pr [Open NICHT erfolgreich]] ~
                       [pr [Open erfolgreich]]
end
```

Init : as before

Finish : Controlling I8 pressed to abort

ROBO Starter Kit : Traffic Lights

StarterAmpel.LGO : M1 red, M2 yellow, M3 green lamp, I1 switch

Normal : green light on, if I1 pressed : one cycle yellow – red – red/yellow – green

```
to aaa
  pr [Traffic Lights started]
  while [not Finish] [  ~
    rbSetMotor 3 :rbOn  ~
    WaitForInput 1      ~
    rbPause8 300*:Fak    ~
    rbSetMotor 3 :rbOff ~
    rbSetMotor 2 :rbOn  ~
    rbPause8 400*:Fak    ~
    rbSetMotor 2 :rbOff ~
    rbSetMotor 1 :rbOn  ~
    rbPause8 1000*:Fak   ~
    rbSetMotor 2 :rbOn  ~
    rbPause8 300*:Fak    ~
    rbSetMotor 1 :rbOff ~
    rbSetMotor 2 :rbOff ]
  rbSetMotors 0
  pr "FINIS
end

to Finish
  output rbGetInput 8
end

to Init
  mswFish40
  rbOpenInterface
  ifelse :rbHandle < 0 [pr [Open NOT successfull]] ~
                       [pr [Open successfull]]
  make "Fak 3
end

to rbPause8 :mSek
  localmake "eZeit timemilli + :mSek
  while [timemilli < :eZeit] [if rbGetInput 8 [stop]]
end

to WaitForInput :InputNr
  while [not (rbGetInput :InputNr)] [if rbGetInput 8 [stop]]
end
```

Init, Finish : as above

rbPause : waiting some milliseconds, can be interrupted by I8

rbSetMotors 0 : Clear all M-Outputs

ROBO Starter Kit : Temperature Control

StarterTemp.LGO : M1 Motor, M2 Lamp, AX NTC, I8 Switch

if :AX < 340 cooling
if :AX > 400 heating

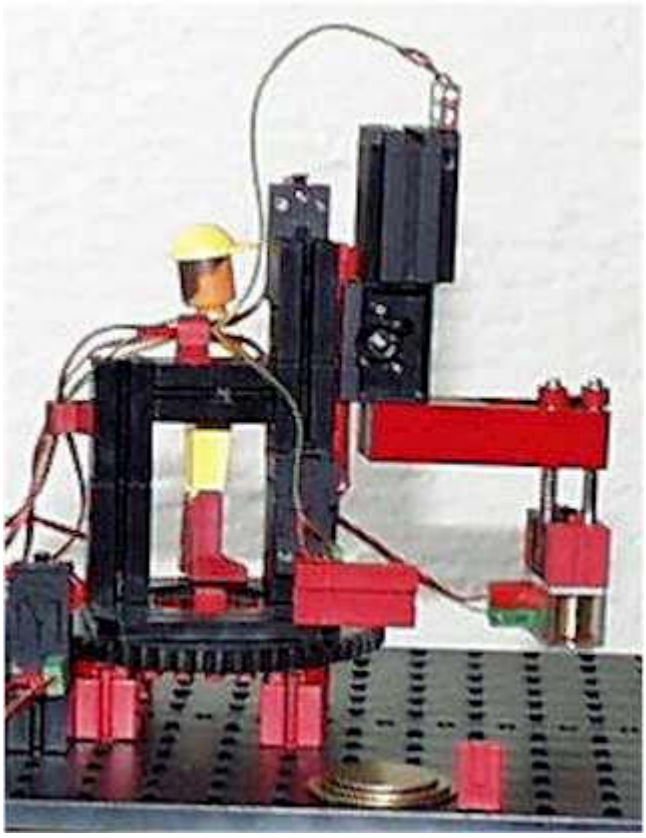
Adjust the values for your commodity

```
to aaa
; --- ROBO Starter Kit : Temperature Control ---
  pr [Temperature started]
  rbSetMotor 2 :rbOn
  while [not Finish] [  ~
    localmake "AX rbGetAnalog 1 ~
    (pr [Temperatur :] :AX)
    if :AX < 340 [rbSetMotor 1 :rbLeft rbSetMotor 2 :rbOff] ~
    if :AX > 400 [rbSetMotor 1 :rbOff  rbSetMotor 2 :rbOn] ~
    rbPause8 1000 ]
  rbSetMotors 0
  pr "FINIS
end

to Finish
  output rbGetInput 8
end

to Init
  mswFish40
  rbOpenInterface
  ifelse :rbHandle < 0 [pr [Open NOT successfull]] [pr[Open
successfull]]
end

to rbPause8 :mSek
  localmake "eZeit timemilli + :mSek
  while [timemilli < :eZeit] [if rbGetInput 8 [stop]]
end
```

Procedures as above

The if's in aaa can be implemented with an ifelse as well.

Tower of Hanoi



Details see www.ftcomputing.de/hanoivbh.htm

Program Hanoi.LGO, start with aaa. M1 Motor with I1 end switch, I2 impulse switch
M2 motor for moving up down I3 Up, I4 Down, M3 magnet

```
to aaa
  pr [Robot wird initialisiert]
  Init
  Hanoi :Scheiben :PosA :PosB :PosC
  rbCloseInterface
  pr [Robot wurde abgeschaltet]
end

to ArmDown
  if rbGetInput :iDown [rbSetMotor :mArm :cOff stop]
  rbSetMotor :mArm :cLeft
  ArmDown
end

to ArmUp
  if rbGetInput :iUp [rbSetMotor :mArm :cOff stop]
  rbSetMotor :mArm :cRight
  ArmUp
end

to Bringe :Pos
  SauleTo :Pos
  ArmDown
  rbSetMotor :mMagnet :cOff
  ArmUp
end
```

```
to Hanoi :n :Anfang :Mitte :Ende
  if :n = 1 [(pr [Scheibe von ] :Anfang [->] :Mitte) Ziehe :Anfang
:Mitte stop]
  Hanoi :n-1 :Anfang :Ende :Mitte
  (pr [Scheibe von ] :Anfang [->] :Mitte)
  Ziehe :Anfang :Mitte
  Hanoi :n-1 :Ende :Mitte :Anfang
end

to Hole :Pos
  SauleTo :Pos
  ArmDown
  rbSetMotor :mMagnet :cLeft
  ArmUp
end

to Init
  make "Scheiben 3
  make "mSaule   1
  make "iImpulse 2
  make "iEnd     1
  make "PosA     20
  make "PosB     50
  make "PosC     80
  make "mArm     2
  make "iDown    4
  make "iUp      3
  make "mMagnet  3
  make "cOff     0
  make "cLeft    1
  make "cRight   2
  make "RobotPos 999

  mswFish40
  rbOpenInterface

  rbSetMotor :mMagnet :cOff
  ArmUp
  SauleTo 0
  SauleTo :PosA
  pr [Auf Grundstellung]
end

to SauleTo :DestPos
  ifelse :RobotPos < :DestPos [rbRobMotor :mSaule :cRight 7
:DestPos-:RobotPos] ~
    [rbRobMotor :mSaule :cLeft 7 :RobotPos-:DestPos]
  while [(rbGetCounter :iImpulse) > 0] []
  make "RobotPos :DestPos
end

to Ziehe :PosFrom :PosTo
  Hole :PosFrom
  Bringe :PosTo
end
```

Init : setting init values, moving robot to home position (end switch true on turning left, magnet up, off)

SauleTo : runnig a RobMotor to position the robot arm with the gear and the arm on it.
RobotPosition < DestinationPosition = right turn
RobotPostion > DestinationPosition = left turn

while waiting for ready then make new position.
PosA, PosB, PosC : the three positions on which the tower plates are postioned (20, 50, 90 impulses from end position (end switch true).