Pira.cz

MRDSIO.DLL – Programming Guide

What's the MRDSIO.DLL?

The MRDSIO.DLL is a dynamic-link library that enables easy access to the MRDS192 and MRDS1322 based RDS encoders from PC and user application. The user does not need to solve how to access the communication ports in his application. The DLL should work in all Windows versions (95 to 8, incl. 64-bit edition).

The DLL is originally created in 32-bit Delphi 6 but should work with any programming language due to use of *stdcall* convention and simple format of data exchange. Please follow the documentation of your programming language for more details about how to use external functions and procedures from DLL. We have no information about how to use the DLL in other programming languages.

MRDSIO.DLL features:

- Basic and system (Connect, Disconnect, Version, …)
- Send data, Receive data

List of functions and procedures provided by the DLL

```
Type TData = array [0..255] of byte;

function Version: PChar; stdcall;
procedure ConnectionSetup(ConMode, PortNum: integer; Bidirectional, Slow:
boolean); stdcall;
function Connect: boolean; stdcall;
procedure Disconnect; stdcall;
function Send(Len: integer; Data: TData): boolean; stdcall;
function Receive(Adr, Len: Integer): TData; stdcall;
```

Function or procedure	Description
Version	Returns actual DLL version.
ConnectionSetup	Specifies connection parameters. ConMode: Set to 0 for the MRDS192. Set 1 for the MRDS1322. Other values are reserved. PortNum: 1-255: RS232 (COM) port number, >255: parallel port (LPT1 = $378). Bidirectional, Slow: the same use as in the original application.
Connect	Connects to the device using the connection parameters above. Returns true if connected successfully.
Disconnect	Disconnects from the device. Call also before repeated connects.
Send	Sends RDS data to the device. Len: Total length of the RDS data being sent, in bytes, excl. the address. For example for complete PS the length is 8. If writing only PTY, the length is 1, etc. Data: The item with index 0 contains the starting address, for example $02 for PS. Following items contain the RDS data, byte per byte. Unused items are ignored. For complete memory listing see the device datasheet.
Receive	Reads RDS data from the device. Adr: Starting address Len: Total length of the RDS data being read The function returns a TData array where the item with index 0 contains the starting address and the item with index 1 contains the length of the data received. Following items (index 2 and higher) contain the data received. If the Receive operation fails, the length value returned in the array is set to zero.

Notes:

Data structure type useds:
PChar – pointer to a null-terminated string.
Integer – signed 32-bit.
Boolean – true/false variable that occupies one byte of memory.

MRDS1322 communication mode selection:
For this device selecting a parallel port always results in use of the I2C protocol. Selecting the serial COM port always results in use of the RS232 protocol. Byte stuffing and its reverse operation is done automatically within the DLL.

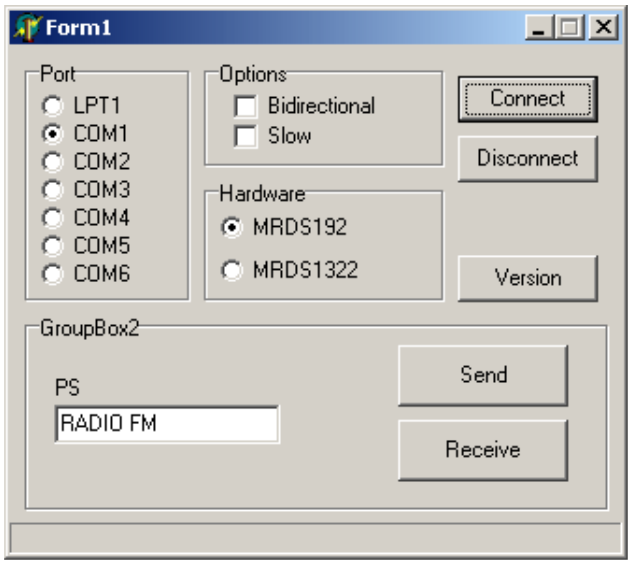
Example of use

Download the example zip file from the website (section RDS Encoders – Software – Development). The example illustrates how to use the DLL in Delphi. Implementation in other programming languages is very similar. Please follow the documentation of your programming language for more details about how to use external functions and procedures from DLL. We have no information about how to use the DLL in other programming languages.

Before running the example make sure the device communicates properly using the original control software.

The example zip file contains:

mrdsio.dll	The DLL file. The DLL must be placed in the application folder and must be distributed with it.
inpout32.dll	A driver required by the main DLL if parallel port is used for the communication.
example.exe	Compiled example. Just double-click to run it.
example.dpr	The example Delphi project file. Open this file in Delphi for edit the example.
example1.pas	The example main unit file.
example1.dfm	The example main form definition.
mrdsio.pdf	This file.



First steps

1. Run the example.exe.
2. Set the connection fields and click Connect. The status bar must indicate ‘OK’.
3. Fill the PS, click Send button.
4. Look at the RDS decoder display.

The example source code listing (example1.pas)

```
unit example1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, StdCtrls, ExtCtrls;

type
  TData = array [0..255] of byte;

type
  TForm1 = class(TForm)
    RadioGroup1: TRadioGroup;
    GroupBox1: TGroupBox;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    Button1: TButton;
    Button2: TButton;
    GroupBox2: TGroupBox;
    Edit1: TEdit;
    Label1: TLabel;
    Button3: TButton;
    StatusBar1: TStatusBar;
    Button4: TButton;
    RadioGroup2: TRadioGroup;
    Button5: TButton;
    procedure Button4Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Button3Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

function Version: PChar; stdcall; external 'mrdsio.dll';
function Connect: boolean; stdcall; external 'mrdsio.dll';
procedure Disconnect; stdcall; external 'mrdsio.dll';
function Send(Len: integer; Data: TData): boolean; stdcall; external
'mrdsio.dll';
function Receive(Adr, Len: integer): TData; stdcall; external 'mrdsio.dll';
procedure ConnectionSetup(ConMode, PortNum: integer; Bidirectional, Slow:
boolean); stdcall; external 'mrdsio.dll';

procedure TForm1.Button4Click(Sender: TObject);
begin
ShowMessage(Version);
end;

procedure TForm1.Button1Click(Sender: TObject);
```

```
var PortNum: integer;
begin
if (RadioGroup1.ItemIndex=0) then PortNum:=$378 else
PortNum:=RadioGroup1.ItemIndex;
ConnectionSetup(RadioGroup2.ItemIndex, PortNum, CheckBox1.Checked,
CheckBox2.Checked);
if (Connect) then StatusBar1.SimpleText:='OK' else
StatusBar1.SimpleText:='Failed';
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
Disconnect;
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
Disconnect;
end;

procedure TForm1.Button3Click(Sender: TObject);
var Data: TData;                    //Data[0] = address, Data[1..] = rds data
    i, Len: integer;                //Len = length of rds data
begin
Data[0]:=$02;                       //PS buffer address
for i:=1 to 8 do Data[i]:=$20;      //fill by spaces at first
for i:=1 to length(Edit1.Text) do
  Data[i]:=Ord(Edit1.Text[i]);      //copy PS to Data
Len:=8;
if (Send(Len,Data)) then StatusBar1.SimpleText:='OK' else
StatusBar1.SimpleText:='Failed';
end;

procedure TForm1.Button5Click(Sender: TObject);
var Data: TData;
    i: integer;
    s: string;
begin
if not (CheckBox1.Checked) then
  begin StatusBar1.SimpleText:='Bidirectional operation not allowed'; exit; end;
for i:=0 to 255 do Data[i]:=0;
Data:=Receive($02,8);
if (Data[1]=0) then
  begin StatusBar1.SimpleText:='Failed'; exit; end;
s:='';
for i:=1 to 8 do
  s:=s+Chr(Data[i+1]);
Edit1.Text:=s;
end;

end.
```

MRDSIO.DLL source code listing

```
library mrdsio;

uses
  SysUtils,
  Classes,
  Forms,
  Unit1 in 'Unit1.pas' {DM1: TDataModule};

{$R *.res}
```

```
function Version: PChar; stdcall;
begin
Result:='1.00';
end;

function Connect: boolean; stdcall;
begin
Result:=DM1.Connect;
end;

procedure Disconnect; stdcall;
begin
DM1.Disconnect;
end;

function Send(Len: integer; Data: Unit1.TData): boolean; stdcall;
begin
Result:=DM1.Send(Len, Data);
end;

function Receive(Adr, Len: Integer): Unit1.TData; stdcall;
begin
Result:=DM1.Receive(Adr, Len);
end;

procedure ConnectionSetup(ConMode, PortNum: integer; Bidirectional, Slow:
boolean); stdcall;
begin
DM1.ConnectionSetup(ConMode, PortNum, Bidirectional, Slow);
end;

Exports
  Version, Connect, Disconnect, ConnectionSetup, Send, Receive;

begin
DM1:=TDM1.Create(Application);
end.
```

Note: Sources from the DM1 data module are not available for public use.

Frequently asked questions

No questions yet.

Conclusion

DLL version: 1.00
Document revision: 21.1.2012

Your feedback is important! Please send us any bug reports.

mail@pira.cz http://pira.cz

(C) 2012 Pira.cz