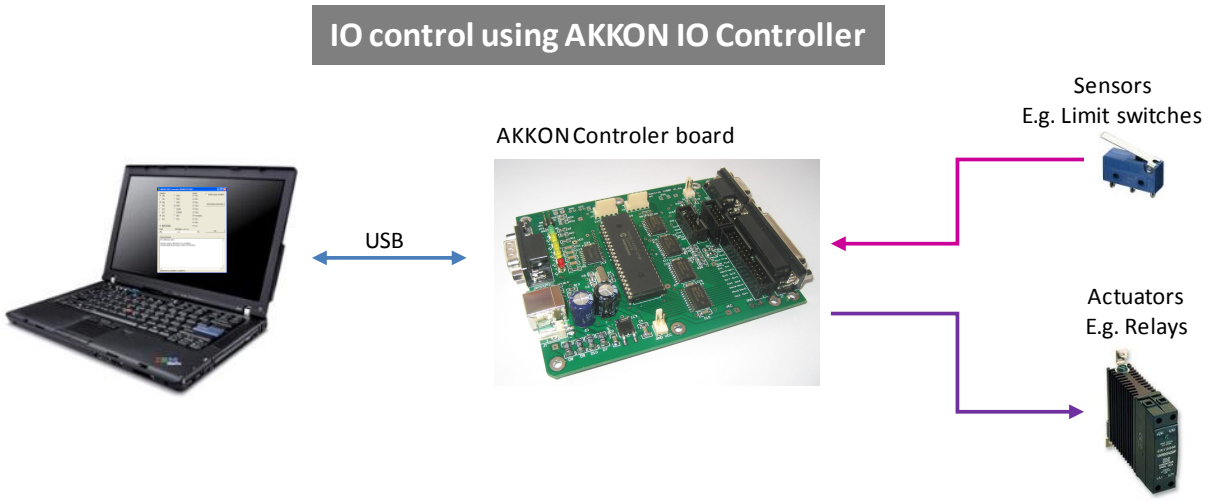
TN026

AKKON IO Control

Operating manual



PRELIMINARY VERSION

Authors: Gerhard Burger
Version: 0.1
Last update: 03.09.2010
File: TN026_AKKON_IO_Control
Attachments: no attachments
Sprache: english

Table of versions

Version	Date	Remarks
0.1	2.09.2010	Preliminary version

©2005-2010 Gerhard Burger

Table of Content

©2005-2010 Gerhard Burger

©2005-2010 Gerhard Burger

1 Introduction

AKKON IO Control is designed for easy access to digital and analogue inputs and outputs over USB. The hardware can be accessed using the AKKON IO control library.

The system can be used for development, controlling and measurement equipment, communicating over USB. The software package includes a Delphi and c# interface library with examples demonstrating the use of the library and hardware. Following picture shows the AKKON controller board that will be supported by the software.

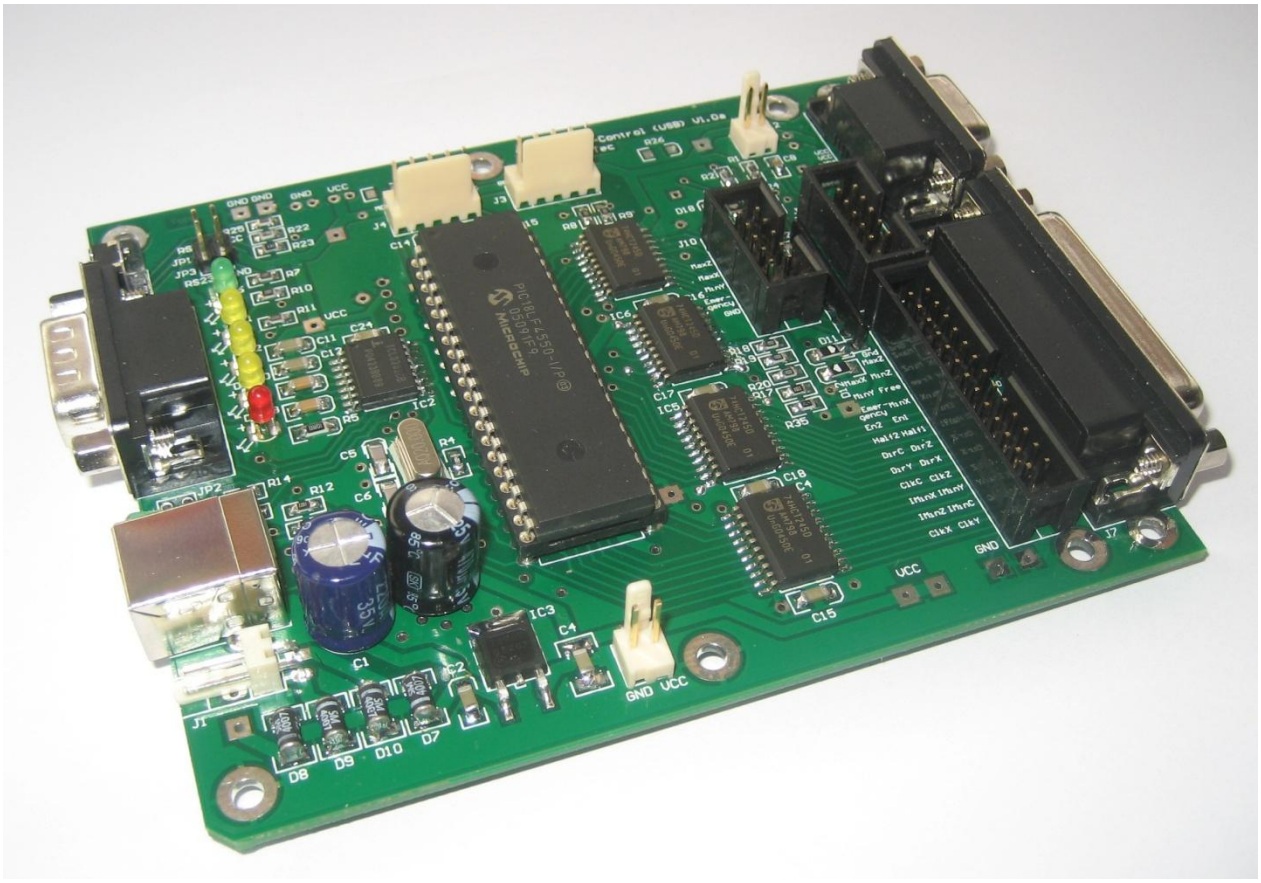


Figure 1: AKKON Controller Board (PIC-Version)

2 Architecture of AKKON IO CONTROLLER Board

Figure 2 shows the architecture of the AKKON IO CONTROLLER board.

The system currently supports following features:

- 16 digital buffered outputs driving 10 mA each, 4 outputs with LED
- 1 PWM output that can also be used as analogue output using a RC-network.
- 10 digital inputs with internal pull-up. Depending on hardware configuration, there are two digital inputs with on board overvoltage / under voltage protection UinMax 24 V
- 2 general purpose digital / analogue inputs depending on the use with interrupt option

©2005-2010 Gerhard Burger

- System timer with millisecond resolution
- Watchdog timer if connection to host PC has been lost
- RS232 interface with optional power source (RS232 interface currently not supported)
- Firmware updates without further hardware using AKKON boot loader or using the In Circuit Serial Programming interface (ICSP)

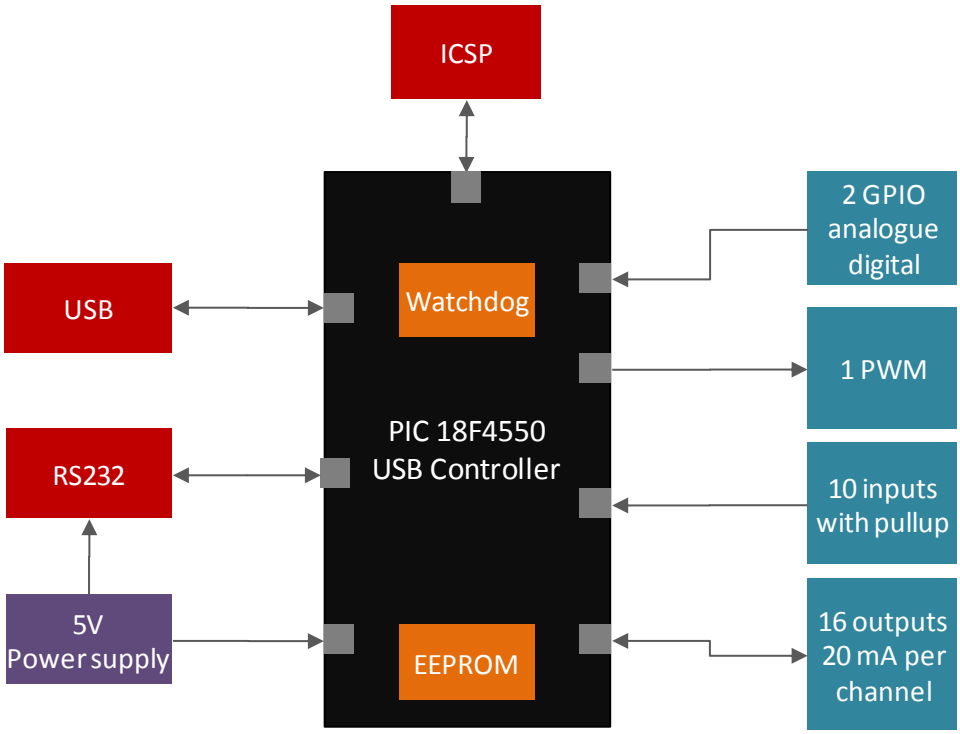


Figure 2: Architecture of AKKON IO Controller

The coloured blocks show the logical units, the Arrows the direction of the information flow.

3 Controller commands and its processing (direct access)

Following description outlines the supported commands by the AKON IO controller. Generally communication is done by commands with parameters that will be sent to the AKKON IO controller. The AKKON IO controller processes the command and responses the processing data as described below.

Please note: Following description outlines the direct access without using the AKKON Controller component (TAkkonController). TAkkonController encapsulate all of the described command and simplifies the access. TAkkonController is described in section 4.

3.1 Write digital outputs

Command: acWRITE_DIGITAL_IO

Output variable: Data: DWORD

Response from AKKON IO controller: --

```
UsbDriver.SendRT_Cmd_DWORD(acWRITE_DIGITAL_IO, Data);
```


©2005-2010 Gerhard Burger

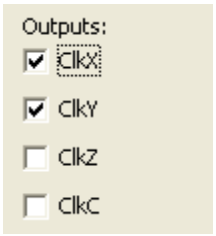
Every bit of variable data relates to one digital output. The upper two bytes are defined for future use.

System constants for digital outputs:

```
pfCLK_X = $00000001;
pfCLK_Y = $00000002;
pfCLK_Z = $00000004;
pfCLK_C = $00000008;
pfDIR_X = $00000010;
pfDIR_Y = $00000020;
pfDIR_Z = $00000040;
pfDIR_C = $00000080;
pfIMIN_X = $00000100;
pfIMIN_Y = $00000200;
pfIMIN_Z = $00000400;
pfIMIN_C = $00000800;
pfENABLE = $00002000;
pfFULL = $00004000;
pfM08 = $00008000;
pfM10 = $00001000;
```

Example:

Following example implements a dialog with four checkboxes. Every checkbox is related to one digital output. Depending on the checkbox state digital outputs ClkX, ClkY, ClkZ and ClkC will be enabled or disabled.



©2005-2010 Gerhard Burger

```
procedure TForm1.WritePort(Data:DWord);
begin
  OutData:=Data; // save current command data
  chkCLKX.Checked:=((Data and pfCLK_X) <> 0);
  chkCLKY.Checked:=((Data and pfCLK_Y) <> 0);
  chkCLKZ.Checked:=((Data and pfCLK_Z) <> 0);
  chkCLKC.Checked:=((Data and pfCLK_C) <> 0);
  if UsbDriver.SendRT_Cmd_DWord(acWRITE_DIGITAL_IO,Data) = 0 then
  Begin
    memo1.Lines.add('Data ' + IntToHex(Data,8) + ' written to AKKON Controller board');
  end
  else
    memo1.Lines.add('No connection to AKKON Controller board');
End;
```

Figure 3: Example source for setting digital outputs on AKKON IO controller board

3.2 Read digital inputs

Command: acREAD_DIGITAL_IO

Output variable: Data: Word;

System constants for digital inputs:

```
pfMIN_X = $0400;
pfMIN_Y = $0800;
pfMIN_Z = $1000;
pfMAX_X = $2000;
pfMAX_Y = $4000;
pfMAX_Z = $8000;
pfEMERGENCY = $0020;
pfRUN   = $0001;
pfSDA   = $0100;
pfSCL = $0200;
```

Response from Controller: State of digital inputs

```
UsbDriver.SendRT_Cmd_GET_WORD(acREAD_DIGITAL_IO, Data);
```

Every bit of data relates to one digital inputs. The upper byte is currently defined for future use.

Example:

Following example outlines a dialog with 10 digital inputs. Method *btnReadClick(Sender :TObject)* reads the digital inputs and depending on the state of dedicated bits the related checkbox will be set to checked or not.

©2005-2010 Gerhard Burger


```
procedure TForm1.btnReadClick(Sender: TObject);
var Data:Word;
begin
  if UsbDriver.SendRT_Cmd_GetWord(acREAD_DIGITAL_IO,Data) = 0 then
  Begin
    chkMinX.Checked:= ((Data and pfMIN_X) <> 0);
    chkMinY.Checked:= ((Data and pfMIN_Y) <> 0);
    chkMinZ.Checked:= ((Data and pfMIN_Z) <> 0);
    chkMaxX.Checked:= ((Data and pfMAX_X) <> 0);
    chkMaxY.Checked:= ((Data and pfMAX_Y) <> 0);
    chkMaxZ.Checked:= ((Data and pfMAX_Z) <> 0);
    chkEmergency.Checked := ((Data and pfEMERGENCY) <> 0);
    chkSDA.Checked := ((Data and pfSDA) <> 0);
    chkSCL.Checked := ((Data and pfSCL) <> 0);
    chkRun.Checked := ((Data and pfRUN) <> 0);
  End
  else
    memo1.Lines.add('No connection to AKKON IO controller board');
end;
```

Figure 4: Example source for reading digital inputs from AKKON IO controller board

3.3 Read controller state

This is a routine of the main AKKON IO Control library. The total state of the controller can be read in one bulk and be processed. In a user application this routine can e.g. be called every 500 ms using a software timer.

Output variable:

Response from AKKON IO controller: ControllerInfo : TControllerInfo;

```
TControllerInfo = packed Record
        OutCtrl:DWORD;
        InCtrl:Word;
        ProcessingState:Word;
        SystemTime:DWORD;
```


©2005-2010 Gerhard Burger

```
        PwmFactor:Word;
        PwmValue:Word;
End;
```

Call of routine: See example source.

3.4 Get system time

Command: acREAD_SYSTEMTIME

Output variable: --

Response from AKKON IO controller: SysTicks: DWORD;

```
UsbDriver.SendRT_Cmd_GET_DWORD(acWRITE_DIGITAL_IO, SysTicks);
```

System time describes time elapsed in milli seconds after startup of the AKKON CONTROLLER board or after last system time reset

3.5 Reset system time

Command: acRESET_SYSTEMTIME

Output variable: --

Response from AKKON IO controller: --

```
UsbDriver.SendRT_Cmd_DWORD(acRESET_SYSTEMTIME);
```

Reset the system timer of the AKKON IO controller board.

3.6 Set PWM duty

Command: acSET_PWM_DUTY

Output variable: Duty 32768

Response from AKKON IO controller: --

```
UsbDriver.SendRT_Cmd_DWORD(acSET_PWM_DUTY);
```

Set PWM duty. The duty in percent is calculated Duty [%] = Duty / 32768 * 100.

3.7 Enable PWM

Command: acPWM_ON

Output variable: --

Response from AKKON IO controller: --

```
UsbDriver.SendRT_Cmd_DWORD(acPWM_ON);
```

Switch PWM controller on

3.8 Disable PWM

Command: acPWM_OFF

©2005-2010 Gerhard Burger

Output variable: --

Response from Controller: --

```
UsbDriver.SendRT_Cmd_BYTE(acPWM_ON);
```

Switch PWM controller off.

3.9 Read hardware information

Command: acGET_VER

Output variable: --

Response from AKKON IO controller: Hardware and firmware version

```
TFirmwareInfo = Record
        Data:Array[0:50] of Byte;
        CompilationDate:Array[050] of Byte;
End;
```

Var Info:TFirmwareInfo:

```
UsbDriver.SendRT_Cmd(acGET_VER,Info.Data);
```

Read current hardware information

Example response from AKKON IO controller:

Aug 30 2010,11:05:01, PIC18F4550

3.10 Read firmware information

Command: acGET_COMPILATION

Output variable: --

Response from AKKON IO controller: Hardware and firmware version

```
TFirmwareInfo = Record
        Data:Array[0:50] of Byte;
        CompilationDate:Array[050] of Byte;
End;
Var Info:TFirmwareInfo;
UsbDriver.SendRT_Cmd(acGET_COMPILATION,Info.CompilationDate);
```

Example response from AKKON IO controller:

AKKON IO V1.0, (c)GB2010

3.11 Store startup parameters

Command: acSTORE_STARTUP_PARAMS

©2005-2010 Gerhard Burger

Output variable:

Response from Controller:

```
UsbDriver.SendRT_Cmd(acSTORE_STARTUP_PARAMS);
```

Save startup parameters to flash resp. EEprom. After system start, all digital outputs as well as the PWM-controller and application timeout will be set to this value.

3.12 Set application timeout

Command: acSET_APPLICATION_TIMEOUT

Output variable: TimeOut: Word; // variable in ms

Response from AKKON IO controller: --

```
UsbDriver.SendRT_Cmd(acSET_APPLICATION_TIMEOUT, TimeOut);
```

AKKON IO controller supports one watchdog timer. If the watchdog timer overflows the application timeout flag will be set and the controller switches to a specified state (See Set timeout parameters). Every time acGET_STATE is called, the watchdog timer will be reset. A watchdog overflow occurs if the connection to the host PC has been lost.

3.13 Set application timeout parameters

Command: acSET_TIMEOUT_PARAMS

Output variable:

```
TTimeOutParams = packed record
        OutCtrlTO: DWord;
        SpeedFactorTO:Word;
End;
```

Response from AKKON IO controller: --

```
UsbDriver.SendRT_Cmd_Buf(acSET_TIMEOUT_PARAMS,TimeOutParams,sizeof(TTimeOutParams));
```

Save system state if a timeout occurs. A timeout can for e.g. occur, if the connection to the PC has been lost. The watchdog timer will be reset after ever call of GET_TOTAL_STATE.

3.14 Read EEPROM

Command: acREAD_EEPROM

Output variable:

```
TEEPROMDATA = packed record
        Adress: Byte; // Start address for reading
        DataLen:Byte; // expected len
End;
```


©2005-2010 Gerhard Burger

Response from Controller:

Function TForm1.ReadEEPROM(Address:Word; Data:Pointer; var len:Integer):Integer;

Read DataLen bytes starting from Address EEPROMData.DataLen from internal EEPROM. In one bulk at maximum 16 data bytes can be read.

Example:

Following example demonstrate reading of 10 bytes from EEPROM.

```
procedure TForm1.ReadEEPROMClick(Sender:TObject);
var Data:Array[0..20] of Byte;
    i, Len:Byte;
    s:String;
Begin
  Len:=10;
  ReadEEPROM(20,@Data,len); // read 10 bytes from EEPROM
  if (Len = 10) then
  Begin
    s:='EEPROM data: ';
    for i:=0 to len-1 do
      s:=s+ IntToHex(Data[i+2],2) + ' ';
    memo1.lines.add(s);
  End
  else
    memo1.lines.add('Error on reading EEPROM')
End;
```

3.15 Write EEPROM

Command: acWRITE_EEPROM

Output variables: EEPROM-Address, length of bytes that have to be written

Input variable: Data:Array[0..n] of Byte;

Response from AKKON IO controller: length data bytes from EEPROM

Function TForm1.WriteEEPROM(Address:Word; Data:Pointer; len:Word):Integer;

Write len data bytes starting from Address to EEPROM.

Note: Address space 192 to 255 is reserved for internal variables like startup state. Address space 0..191 can be used for user variables.

Example:

Following example demonstrate writing of three bytes to EEPROM.

```
procedure TForm1.Button1Click(Sender: TObject);
var EEPROMData : Array[0..2] of Byte;
begin
  EEPROMData[0] := $0A;
  EEPROMData[1] := $0B;
  EEPROMData[2] := $0C;
  WriteEEPROM(20,@EEPROMData,3);
end;
```


©2005-2010 Gerhard Burger

3.16 Write RS232 (not implemented in current version)

Command: acWRITE_RS232

Output variable: Data: Array[0..63] of Byte;

Response from AKKON IO controller: --

```
UsbDriver.SendRT_Cmd_DWORD(acWIRTE_RS232,Buf,len);
```

Write len bytes of buffer data to the internal FIFO of the AKKON CONTROLLER board. The FIFO data will be output to the RS232 of the AKKON Controller board.

3.17 Read RS232 (not implemented in current version)

Command: acREAD_RS232

Output variable: --

Response from AKKON IO controller: Data: Array[063] of Byte;

```
TDataPackage = record
        Len:Byte;
        Data:Array[0..63] of byte;
End;
UsbDriver.SendRT_Cmd_GET_BUF(acREAD_RS232,Buf,len);
```

Read len bytes of buffer data from the internal FIFO of the AKKON CONTROLLER board.

3.18 CLEAR RS232 output FIFO (not implemented in current version)

Command: acCLEAR_RS232_OUT_FIFO

Output variable: --

Response from AKKON IO controller: --

```
UsbDriver.SendRT_Cmd(acCLEAR_RS232_OUT_FIFO);
```

Clear output fifo of RS232 interface

3.19 CLEAR RS232 input FIFO (not implemented in current version)

Command: acCLEAR_RS232_IN_FIFO

Output variable: --

Response from AKKON IO controller: --

```
UsbDriver.SendRT_Cmd(acCLEAR_RS232_IN_FIFO);
```

Clear input fifo of RS232 interface

©2005-2010 Gerhard Burger

4 AKKON IO controller library for Delphi

USB-programming of the AKKON IO controller programming is supported by the AKKON IO controller library. The library includes the AKKON USB driver component (TIOPicUsbDriver) for basic communication and the AKKON controller component (TAkkonController) for performing IO control.

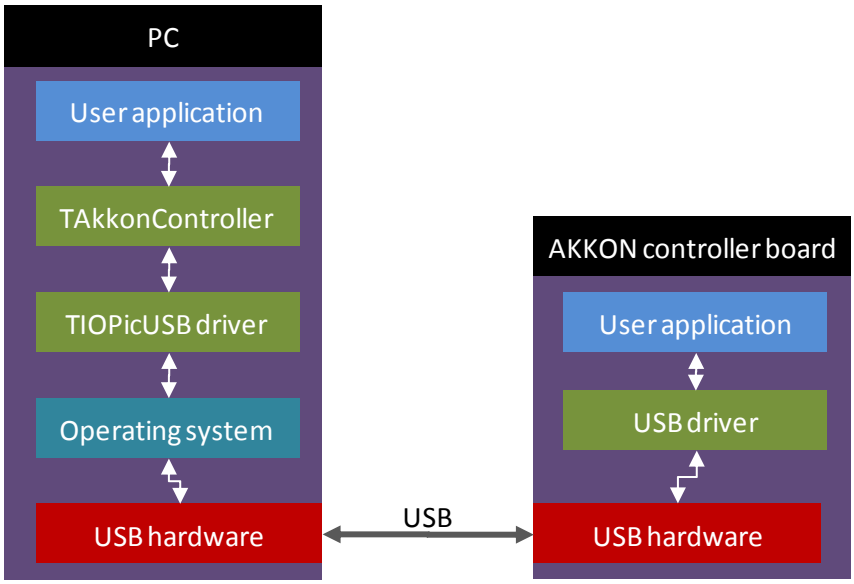


Figure 5: Information flow through layers

In normal case the USB driver component is not accessed directly. Moreover TIoPicUSBDriver creates an interface between the operating system and the functional logic for programming the AKKON controller board. The user application itself accesses the TAkkonController-Component and has access to different methods for steering the functional blocks on the controller.

4.1 Components

4.1.1 AKKON USB driver (TIOPicUsvDriver) component

The AKKON USB driver component implements basic communication from PC to AKKON IO controller board.

USB-driver for AKKON IO controller

The component implements following parameters and events:

©2005-2010 Gerhard Burger

Figure 6: Parameters (left) and events supported by the TIOPicUsbDriver-component

In normal case the USB driver component is not accessed directly.

4.1.2 AKKON Controller (TAkkonController) component

The AKKON Controller component (TAkkonController) implements

AKKON Controller board component

The component implements following parameters and events:

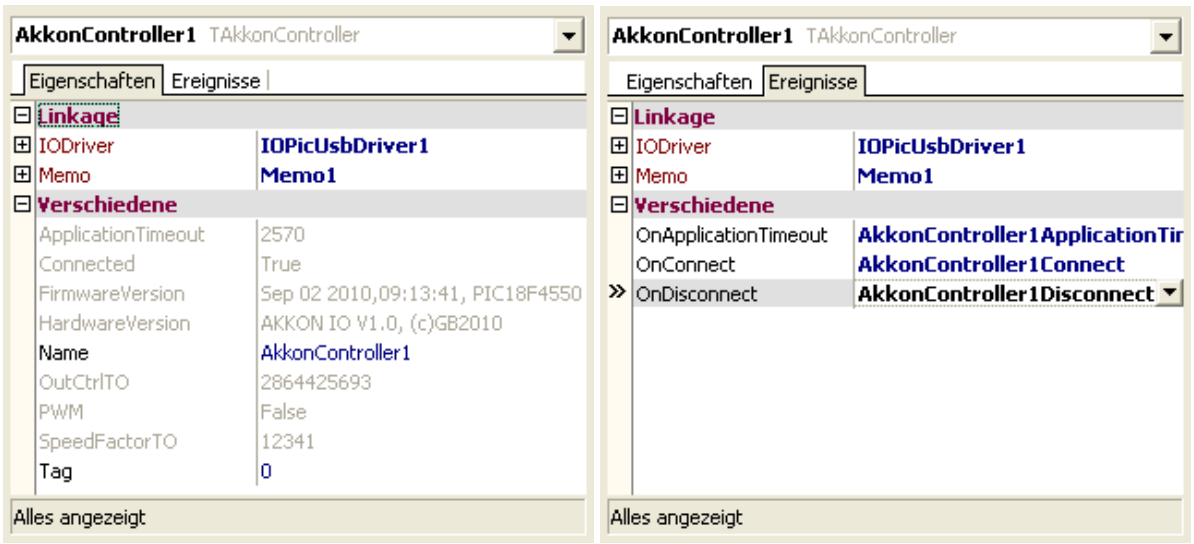


Figure 7: Parameters (left) and events supported by the TAkkonController-component

Parameter IODriver holds a reference to a suitable IO driver. In the case of the AKKON IO controller board it is a reference to the TIoPicUSBdriver component.

©2005-2010 Gerhard Burger

4.2 Use of the components in the Delphi IDE

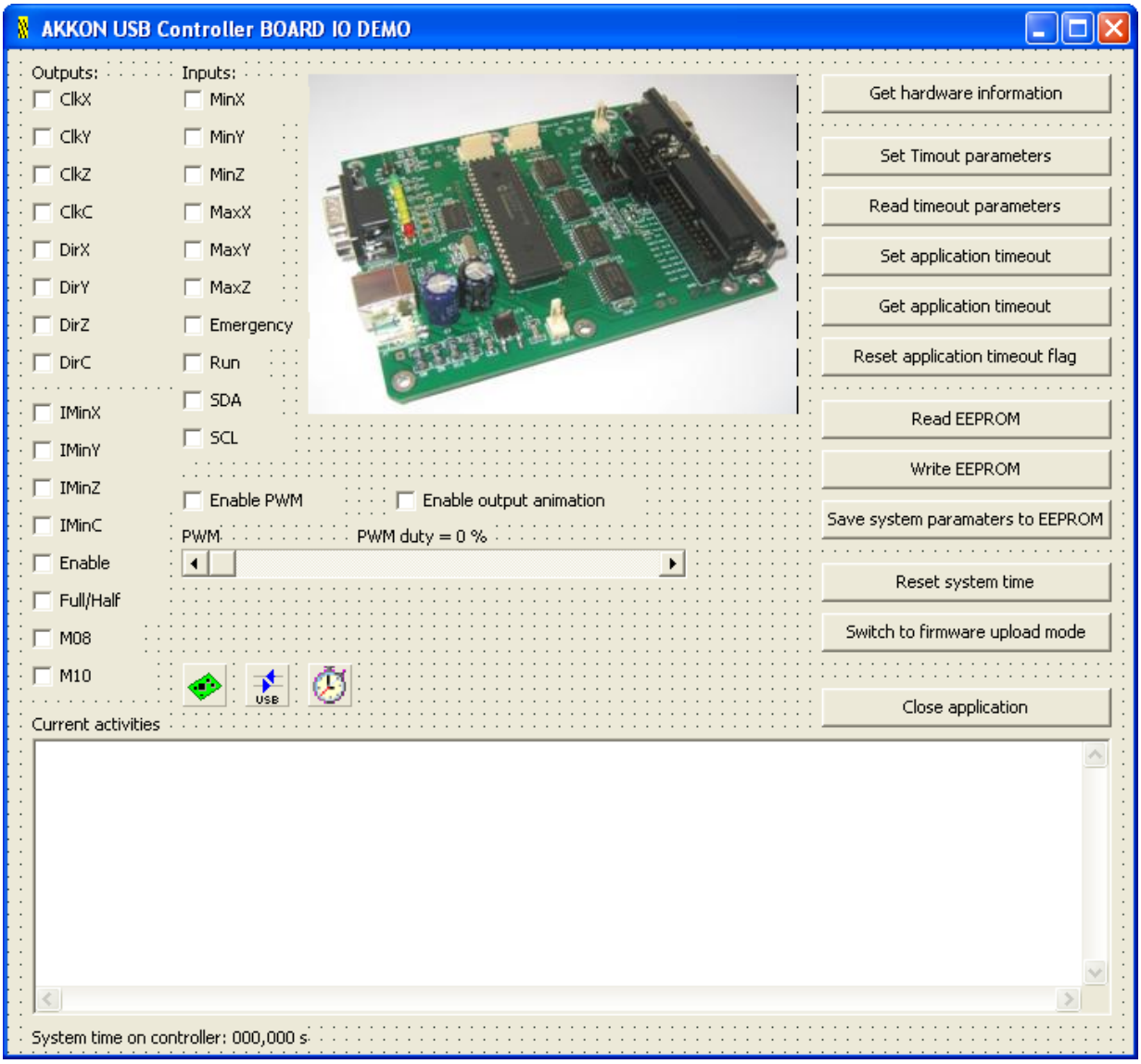


Figure 8: Example application and the use of the AKKON IO library in the Delphi IDE

In the following example application a TAkkonController, a TIoPicUsbDriver and a TTimer component is used. TAkkonController is connected with the TIoPicUsbDrvier. TTimer calls background processing of the current controller state.

5 Demonstration software

The AKKON IO controller Board DEMO program demonstrates the use of the described library functions in this document. The program is written in Borland Delphi 2005.

©2005-2010 Gerhard Burger

5.1 Dialog of AKKON IO controller board DEMO program

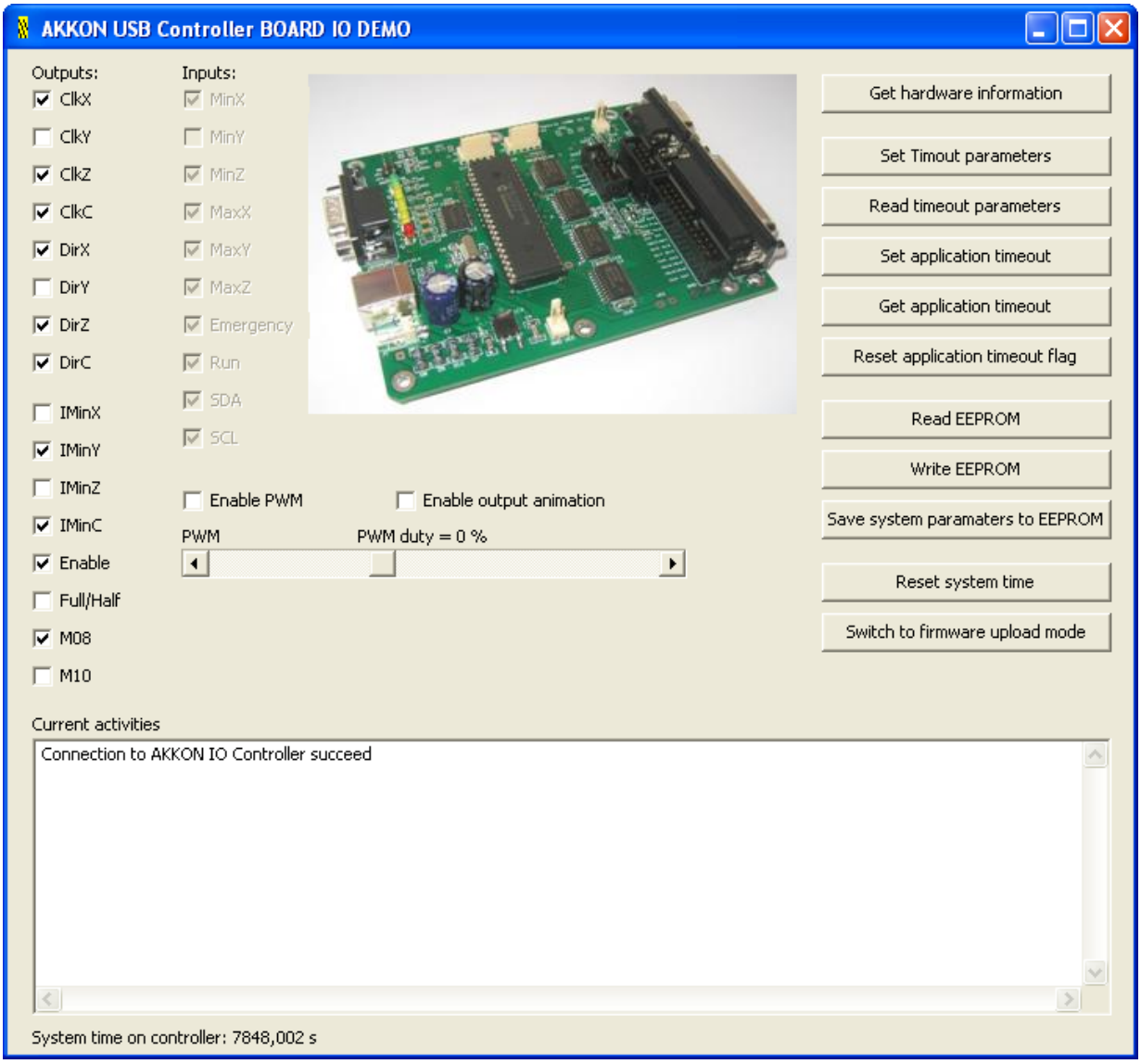


Figure 9: Screenshot of AKKON IO control demo application

5.2 Methods of AKKON USB Controller Board DEMO program

Following description shows main methods of the TAkkonController component and its use in the AKKON IO controller DEMO program.

5.2.1 Setting digital output

```
{
  set bit CLK_X to low or high
}
procedure TForm1.chkClkXClick(Sender: TObject);
begin
  if chkCLKX.Checked then AkkonController1.SetOutput(pfCLK_X,true)
                     else AkkonController1.SetOutput(pfCLK_X,false)
end;
```


©2005-2010 Gerhard Burger

5.2.2 Read digital inputs respectively controller state

Tin the following example a timer is used to read the state of the state of AKKON IO controller continuously. Update time is set to 250 milliseconds.

```
{
  Call ProcessIO and read all information about the controller state
  and display the state in the application window.

  if output animation mode is enabled, then directly set all digital
  outputs at the same time

}
procedure TForm1.Timer1Timer(Sender: TObject);
var NewTime:DWord;
begin
  LedCounter:=LedCounter + 1;    // Ignore overflow
  Begin
    If (TMRUpdate Mod 4 = 0) then  btnGetStateClick(Sender);
    AkkonController1.ProcessIO();
    NewTime:=AkkonController1.ControllerInfo.SystemTime;
    if (NewTime <> Time) then
    begin
      Time:=NewTime;
      Label2.Caption := FormatFloat('##0.###',Time/1000)+ ' s';
    end;
    if chkAnimation.Checked then AkkonController1.WritePort(LedCounter);
    Inc(TMRUpdate);
    if TMRUpdate mod 4 = 0 then  DisplayInputState(); // read limit switches
  end;
end;
```

5.2.3 Read hardware and firmware information

```
procedure TForm1.btnGetInformationClick(Sender: TObject);
Begin
  memo1.lines.add(AkkonController1.FirmwareVersion);
  memo1.lines.add(AkkonController1.HardwareVersion);
end;
```

5.2.4 Set duty of PWM controller

```
procedure TForm1.sbPWMChange(Sender: TObject);
begin
  AkkonController1.SetPwmDuty(sbPWM.Position/1000*100);
end;
```

Example response from AKKON IO Controller:

```
Sep 02 2010,09:13:41, PIC18F4550
AKKON IO V1.0, (c)GB2010
```

5.2.5 Enable / disable PWM controller

```
procedure TForm1.chkPWMClick(Sender: TObject);
begin
  AkkonController1.SetPWM(chkPwm.Checked);
end;
```


©2005-2010 Gerhard Burger

5.2.6 Save system startup parameters to EEPROM

The startup parameters define the state of he digital outputs and PWM after power-up.

```
procedure TForm1.btnSaveToFlashClick(Sender: TObject);
begin
  if AkkonController1.StoreStartupParamters() = 0 then
    memo1.lines.Add('System variables saved to flash');
end;
```

5.2.7 Write user data to EEPROM

```
{
  Write three bytes to location 20 in EEPROM
}
procedure TForm1.btnWriteEEPROMClick(Sender: TObject);
var EEPROMData : Array[0..2] of Byte;
begin
  EEPROMData[0] := $AA;
  EEPROMData[1] := $AB;
  EEPROMData[2] := $AC;
  AkkonController1.WriteEEPROM(20,@EEPROMData,3);
end;
```

Please note: Only address space 0 to 191 can be used for user defined variables. The other 64 bytes from address 192 to 255 are used for saving system variables like. e.g. system state after power-up.

©2005-2010 Gerhard Burger

5.2.8 Read content of EEPROM

```
{
  Read total EEPROM data and display it in memo box
}
procedure TForm1.btnReadEEPROMClick(Sender: TObject);
var EEPROMData : Array[0..100] of Byte;
    i, len:Integer;
    s:String;
    j:Integer;
    Address:Byte;
begin
  Address:=0;
  memo1.lines.add('');
  memo1.lines.add('------------------');
  memo1.lines.add('Content of EEPROM');
  for j:=0 To 15 do
  Begin
    Len:=16;
    AkkonController1.ReadEEPROM(Address,@EEPROMData,len);
    if (len) > 0 then
    Begin
      s:='Adress ' + IntToHex(j*16,2) + ': ';
      for i:=0 to len-1 Do
      Begin
        s:=s + InttoHex(EEPROMData[i+2],2) + ' '
      End;
      memo1.Lines.add(s);
    End;
    inc(Address,16);
  end;
  memo1.lines.add('');
end;
```


©2005-2010 Gerhard Burger

6 Disclaimer

6.1 Limited Warranty and Disclaimer of Warranty

THIS SOFTWARE AND ACCOMPANYING WRITTEN MATERIALS (INCLUDING INSTRUCTIONS FOR USE) ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. FURTHER, the author DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF USE, OF THE SOFTWARE OR WRITTEN MATERIALS IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, CURRENTNESS, OR OTHERWISE. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE SOFTWARE IS ASSUMED BY YOU. IF THE SOFTWARE OR WRITTEN MATERIALS ARE DEFECTIVE YOU, AND NOT the author OR ITS DEALERS, DISTRIBUTORS, AGENTS, OR EMPLOYEES, ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR, OR CORRECTION.

THE ABOVE IS THE ONLY WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, THAT IS MADE BY the author, ON THIS PRODUCT. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY the author, ITS DEALERS, DISTRIBUTORS, AGENTS OR EMPLOYEES SHALL CREATE A WARRANTY OR IN ANY WAY INCREASE THE SCOPE OF THIS WARRANTY AND YOU MAY NOT RELY ON ANY SUCH INFORMATION OR ADVICE.

NEITHER the author NOR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION OR DELIVERY OF THIS PRODUCT SHALL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL OR INCIDENTAL DAMAGES (INCLUDING DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, AND THE LIKE) ARISING OUT OF THE USE OR INABILITY TO USE SUCH PRODUCT EVEN IF the author HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

6.2 ACKNOWLEDGMENT

BY USING THIS PRODUCT YOU ACKNOWLEDGE THAT YOU HAVE READ THIS LIMITED WARRANTY, UNDERSTAND IT, AND AGREE TO BE BOUND BY ITS' TERMS AND CONDITIONS. YOU ALSO AGREE THAT THE LIMITED WARRANTY IS THE COMPLETE AND EXCLUSIVE STATEMENT OF AGREEMENT BETWEEN THE PARTIES AND SUPERSEDE ALL PROPOSALS OR PRIOR AGREEMENTS, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN THE PARTIES RELATING TO THE SUBJECT MATTER OF THE LIMITED WARRANTY.

©2005-2010 Gerhard Burger