"TouchScreen" project

**Albert Dorn, Eduard Heidt, Jan Helber,
Alexander Irro and Cedric Pilot**

Important ERDE Classes V0.1
written by Eduard Heidt

This documentation contains a small documentation for the ERDE Win32 App
of the project "TouchScreen TI6"

V0.1	2006-12-04	Initial Release
V0.2	2006-12-10	further information

Powered by LaTeX
generated at 10:02 on 11th December 2006

Contents

1 Introduction

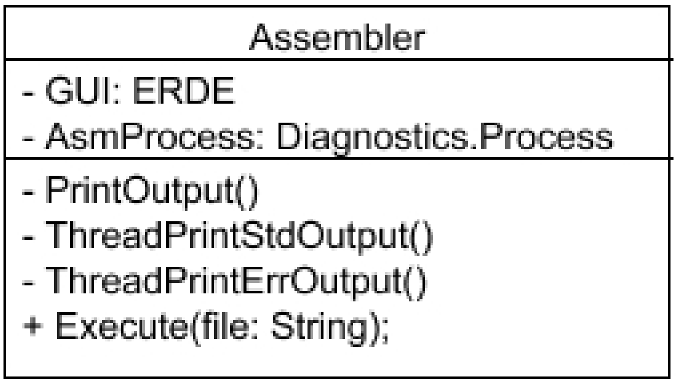
Int this documentation you can see two important Classes that had to be implemented in the ERDE IDE;

The Assembler Class is dealing with an external Process and redirection thats Standard Output/Error to the ERDE IDE.

The Debugger Class is using the Serial Interface to communicate with the Debugger Software on the ARM Board.

2 Assembler

The Assembler Class has 2 Member Classes, the AsmProcess and the ERDE GUI. The AsmProcess is started bei using the Execute Method. The Execute Method is starting the AsmProcess and redirecting thats Standard Output/Error Streams. While the AsmProcess is running there are parallel two Threads that are listening to the redirected Streams for printing the incoming Data to the Erde GUI Textbox. on the ERDE GUI.



2.1 Assembler Source Code

```
/*****************************************************************************
 * Assembler.cs                                                Eduard Heidt WS06
 *
 * Ein fremder Prozess (Assembler.jar) wird gestartet und desses Ausgaben
 * in den Hostprozess (ERDE) umgelenkt..
 * ***************************************************************************/
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;
using System.Windows.Forms;
using System.Diagnostics;

namespace ERDE
{
  class Assembler
  {
    ERDE GUI; // Erde GUI
    private Process AsmProcess = new Process(); // unser Java Assembler

    public Assembler(ERDE erde)
    {
      this.GUI = erde;

      // Prozessparameter initialisieren
      this.AsmProcess.StartInfo.WorkingDirectory = Application.StartupPath;
      this.AsmProcess.StartInfo.Verb = "Open";
      this.AsmProcess.StartInfo.CreateNoWindow = true;
      this.AsmProcess.StartInfo.UseShellExecute = false;
      this.AsmProcess.StartInfo.RedirectStandardOutput = true;
      this.AsmProcess.StartInfo.RedirectStandardError = true;
    }

    public void Execute(string asmfile)
    {
      this.GUI.Output.Text = "";

      // Übergabe parameter vorbereiten...
      string[] cmd = ERDE.Config.AppSettings.Settings["Assembler"].Value.Split(new ch
      this.AsmProcess.StartInfo.FileName = cmd[0];

      if (cmd.Length > 1)
        this.AsmProcess.StartInfo.Arguments = cmd[1] + "  \"" + asmfile + "\"";
      else
        this.AsmProcess.StartInfo.Arguments = asmfile;

      this.AsmProcess.Start();

      Thread std = new Thread(new ThreadStart(ThreadPrintStdOutput));
      Thread err = new Thread(new ThreadStart(ThreadPrintErrOutput));

      std.Start();
      err.Start();
      this.AsmProcess.WaitForExit();
    }

    private void ThreadPrintStdOutput() //StdOut auslesen
    {
      while(!AsmProcess.StandardOutput.EndOfStream)
          PrintOutput(AsmProcess.StandardOutput.ReadLine() + "\r\n");
    }
    private void ThreadPrintErrOutput() //StdErr auslesen
    {
      while(!AsmProcess.StandardError.EndOfStream)
          PrintOutput(AsmProcess.StandardError.ReadLine() + "\r\n");
    }

    delegate void PrintOutputCallback(string msg);
    private void PrintOutput(string msg)
    {
      if (this.GUI.Output.InvokeRequired)
      {
        PrintOutputCallback d = new PrintOutputCallback(PrintOutput);
        if (msg != "")
            this.GUI.Output.Invoke(d, new object[] { msg });
      }
      else
      {
        this.GUI.Output.Text += msg;
        //[> Damit Output nach unten gescrollt wird...
        this.GUI.Output.Parent.Focus();
        this.GUI.Output.Focus();
        this.GUI.Output.ScrollToCaret();
        this.GUI.Output.Select(this.GUI.Output.Text.Length - 1, 1);
        //<]
      }
    }
  }
}
```

3 Debugger

The Debugger Class has 2 Member Classes, the SerialPort and the ERDE GUI; Using the Init Method the SerialPort is initiated with the accordant Settings. By using the Open Method the Debugger Class is going into the Online Mode, that means the SerialPort is now open and the Communication with the Debugger on the ARM Side is initiated. The ER is now clocked by the Debugger.

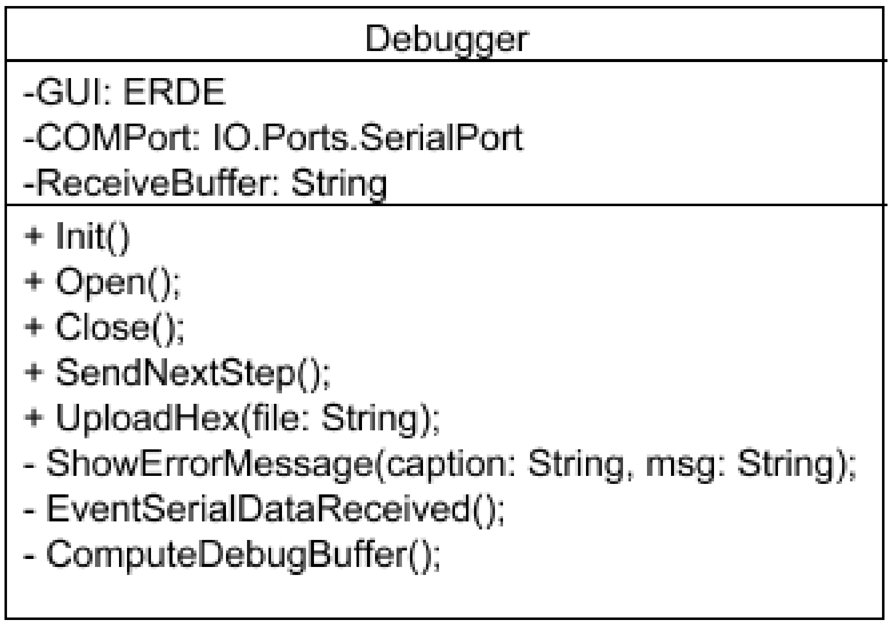
When the Debugger is online you can use the SendNextStep Method for clocking the ER, after that (if some Messages are sent back from Debugger on the ARM Side) the Method EventSerialDataReceived should be started and dumping some data into the ReceiveBuffer.

The ComputeRecieveBuffer Method is parsing ReceiveBuffer and calculating the current Code Line Position (PC) and displaying it in the ERDE GUI.

The UploadHex Method case that the file in the Parameter is beeing send throw the Serial Port tho the Debugger on the ARM Side.

The ShowErrorMessage Method is showing a ErrorMessages in MessageBox.

The Close Method is closing the SerialPort. The ER shoud be clocked by his internal clock now.


Debugger
-GUI: ERDE
-COMPort: IO.Ports.SerialPort
-ReceiveBuffer: String
+ Init()
+ Open();
+ Close();
+ SendNextStep();
+ UploadHex(file: String);
- ShowErrorMessage(caption: String, msg: String);
- EventSerialDataReceived();
- ComputeDebugBuffer();

3.1 Debugger Source Code

```
/******************************************************************************
 * Debugger.cs                                                Eduard Heidt WS06
 *
 * ****************************************************************************/

using System;
using System.Collections.Generic;
using System.Text;
using System.IO.Ports;
using System.Windows.Forms;

namespace ERDE
{
  class Debugger
  {
    ERDE GUI; // Erde GUI
    SerialPort COMPort; // SerialPort

    string ReceiveBuffer = "AC=0\r\nPC=0\r\n";  // RecieveBuffer

    string msgNextStep = ERDE.Config.AppSettings.Settings["DEBUG_STEP"].Value;
    string msgDebugON = ERDE.Config.AppSettings.Settings["DEBUG_ON"].Value;
    string msgDebugOFF = ERDE.Config.AppSettings.Settings["DEBUG_OFF"].Value;

    /// <summary>
    /// Debugger Konsturktor braucht referenz auf die ERDE GUI
    /// </summary>
    /// <param name="RegList"></param>
    /// <param name="SC"></param>
    public Debugger(ERDE erde)
    {
        this.GUI = erde;
        Init();
    }

    /// <summary>
    /// Initialtisiert den Debuger (Setzt Serial Port Konfiguration)
    /// </summary>
    public void Init()
    {
      if (COMPort != null) //falls Config geändert
      {
          if (COMPort.IsOpen)
              COMPort.Close();
      }

      StopBits sbit = StopBits.One;
      Parity par = Parity.None;

      string port = ERDE.Config.AppSettings.Settings["PortName"].Value;
      int baud = Convert.ToInt32(
                          ERDE.Config.AppSettings.Settings["BaudRate"].Value);
      int dbits = Convert.ToInt32(
                     ERDE.Config.AppSettings.Settings["DataBits"].Value);

      switch ((string)ERDE.Config.AppSettings.Settings["Parity"].Value)
      {
          case "Even": par = Parity.Even; break;
          case "Mark": par = Parity.Mark; break;
          case "None": par = Parity.None; break;
          case "Odd": par = Parity.Odd; break;
          case "Space": par = Parity.Space; break;
          default: par = Parity.None; break;
      }

      switch ((string)ERDE.Config.AppSettings.Settings["StopBits"].Value)
      {
          case "None": sbit = StopBits.None; break;
          case "One": sbit = StopBits.One; break;
          case "OnePointFive": sbit = StopBits.OnePointFive; break;
          case "Two": sbit = StopBits.Two; break;
          default: sbit = StopBits.One; break;
      }


      this.COMPort = new SerialPort(port, baud, par, dbits, sbit);
      this.COMPort.ReadTimeout = 500;
      this.COMPort.DataReceived += new SerialDataReceivedEventHandler(
                                              EventSerialDataReceived);
      this.COMPort.Handshake = Handshake.None;
    }

    /// <summary>
    /// Verarbeitet den DebugBuffer
    /// -> setzt ensprechende DubugPosition und Registerwerte
    /// </summary>
    private void ComputeDebugBuffer()
    {

        System.Threading.Thread.Sleep(200);
        string[] lines = this.ReceiveBuffer.Split(new char[] { '\n' });
        //ReceiveBuffer = "";

        foreach(string liner in lines)
        {
          if (liner.Length > 0)
          {
            string line = liner.Substring(0, liner.Length - 1);

            if (line.Contains("="))
            {
              int pos = line.IndexOf('=');

              string reg = line.Substring(0, pos);
              string val = line.Substring(pos + 1);

              this.GUI.SetDebugRegister(reg, val);

              if (reg.Equals("PC"))
              {
                this.GUI.SetDebugPosition(System.Convert.ToInt32(val, 16));
              }
            }
          }
        }

    }

    /// <summary>
    /// Wird aufgerufen wenn Daten am Serial Port ankommen
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void EventSerialDataReceived(object sender, SerialDataReceivedEventArgs e
    {
        try
        {
            this.ReceiveBuffer += this.COMPort.ReadExisting();
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
```

```
/// <summary>
/// Sendet die Step Message an den ARM Debugger
/// </summary>
public void SendNextStep()
{
    try
    {
        this.COMPort.Write(this.msgNextStep);
        this.ComputeDebugBuffer();
    }
    catch (Exception ex)
    {
         this.ShowErrorMessage( "Debugger.NextStep", ex.Message);
    }
}

/// <summary>
/// Sendet den BinärCode
/// </summary>
/// <param name="file"></param>
public void UploadHex(string file)
{
    char []buffer = new char[512];
    try
    {
        System.IO.StreamReader r = new System.IO.StreamReader(file,
                                 System.Text.Encoding.Default);
        r.Read(buffer, 0, 512);
        r.Close();

        byte[] bbuffer = new byte[512];

        for (int i = 0; i < 512; i++)
            bbuffer[i] = (byte)buffer[i];

        this.COMPort.Open();
        this.COMPort.Write(bbuffer, 0, 512);
          this.ComputeDebugBuffer();
        this.COMPort.Close();

    }
    catch (Exception ex)
    {
        this.ShowErrorMessage("Debugger.UploadHex", ex.Message);
    }
}
```

```
    /// <summary>
    /// Öffnet den SerialPort und sendet DebugMode ON Message
    /// </summary>
    public void Open()
    {
        try
        {
            this.COMPort.Open();
            this.COMPort.Write(this.msgDebugON);
            this.ComputeDebugBuffer();
        }
        catch (Exception ex)
        {
            this.ShowErrorMessage("Debugger.Open", ex.Message);
        }
    }
    /// <summary>
    /// Schließt den SerialPort und sendet DebugMode OFF Message
    /// </summary>
    public void Close()
    {
        if (COMPort.IsOpen)
            try
            {
                this.COMPort.Write(msgDebugOFF);
                this.COMPort.Close();
            }
            catch (Exception ex)
            {
                this.ShowErrorMessage("Debugger.Close", ex.Message);
            }
    }
    /// <summary>
    /// Zeigt Error Message
    /// </summary>
    /// <param name="caption"></param>
    /// <param name="msg"></param>
    void ShowErrorMessage(string caption, string msg)
    {
        System.Windows.Forms.MessageBox.Show(msg, caption,
                                MessageBoxButtons.OK, MessageBoxIcon.Error);

    }
  }
}
```

4 Scintilla Editor Component (http://scintilla.org/)

Scintilla is Window Control that is originally written in C++. Its primary programming interface is through Windows messages. ScintillaNET is a .NET 2.0 wrapper around the Scintilla Native control. The current version of the NET Controll is a Alpha version 0.61.

The following Code demonstrates how to add the Control to the Dialog and how to initialize it.

4.1 Init Method

```
private void InitScintilla()
{
  this.sc = new Scintilla.ScintillaControl();
  this.panel.Controls.Add(this.sc);
  // sc
  this.sc.Dock = System.Windows.Forms.DockStyle.Fill;
  this.sc.Location = new System.Drawing.Point(0, 0);
  this.sc.Name = "sc";
  this.sc.TabIndex = 0;
  this.sc.Text = "";

  this.sc.StyleResetDefault();

  this.sc.LexerLanguage("ASM");

  // Configuration...

  Scintilla.Configuration.ConfigurationUtility cu =
        new Scintilla.Configuration.ConfigurationUtility(GetType().Module.Assembly);

  Scintilla.Configuration.Scintilla c = new Scintilla.Configuration.Scintilla();
  Scintilla.Configuration.Scintilla SCConfig
        (Scintilla.Configuration.Scintilla)cu.LoadConfiguration(
        typeof(Scintilla.Configuration.Scintilla), "ScintillaNET.config");

  this.sc.Configuration = SCConfig;


  this.sc.ConfigurationLanguage = "ASM";

  this.sc.CodePage = 65001;
  this.sc.CaretFore = 0xffffff;
  this.sc.CharAdded += new  Scintilla.CharAddedHandler( CharAdded );
  this.sc.CaretLineBack = 0x0;
  this.sc.CaretFore = 0x0;

  this.sc.EmptyUndoBuffer();

    if(System.IO.File.Exists("polling.er1"))
      OpenFile("polling.er1");

    this.sc.IsHScrollBar = false;

    this.sc.SetMarginWidthN(0, 40);

  ////////////////////////////////////

}
private void CharAdded(Scintilla.ScintillaControl pSender ,int ch)
{
  //linenumbers anzeigen
  this.sc.SetMarginWidthN(0,40);
    isdirty = true;
}
```

4.2 Scintilla Configuration (ScintillaNET.config)

The ER Syntax and Color Schema is defined in the ScintillaNET.config File. Here you can see why some Keywords are highlighted and what Font or Color is used for.

```
<?xml version="1.0" encoding="utf-8"?>
<Scintilla xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <globals>
  </globals>

  <keyword-classes>
    <keyword-class name="asm">
      mov load store jumpz zero and nop dc.w add or
  </keyword-class>
  </keyword-classes>
  <languages>

    <language name="ASM">
      <lexer name="asm" key="5" />
      <file-extensions>asm</file-extensions>
      <use-keywords>
        <keyword key="0" class="asm" />
      </use-keywords>
      <use-styles>

    <style name="COMMENTS" key="1"  fore="gray"  font="Courier New" size="10"/>
    <style name="NUMBERS"  key="2"  fore="red"  font="Courier New" size="10"/>
    <style name="TEXT"     key="5"  fore="black"  font="Courier New" size="10"/>
    <style name="KEYWORDS" key="6"  fore="teal"  font="Courier New" size="10"/>

    <style name="STYLE_LINENUMBERS" key="33"  back="0xEEEEEE" />

      </use-styles>
    </language>
  </languages>
</Scintilla>
```

5 The ERDE Configuration file

The Settings used by the ERDE Application for example the Serial Port Parameters are stored in the ERDE.exe.config. NET is providing a Configuration Class that stores the Settings in XML. Getting the Parameter you have to use for example

```
ERDE.Config.AppSettings.Settings["BaudRate"].Value
```

5.1 ERDE Configuration (ERDE.exe.config)

```
<?xml version="1.0"?>

<configuration>
 <appSettings file="">
  <clear />
  <add key="PortName" value="COM1" />
  <add key="BaudRate" value="9600" />
  <add key="Parity" value="None" />
  <add key="DataBits" value="8" />
  <add key="StopBits" value="One" />
  <add key="DEBUG_ON" value="g" />
  <add key="DEBUG_OFF" value="p" />
  <add key="DEBUG_STEP" value="n" />
  <add key="Assembler" value="java -jar Assembler.jar " />
 </appSettings>
</configuration>
```