"TouchScreen" project

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

Product Description V0.1
written by Albert Dorn

This documentation contains the product description of the project "TouchScreen TI6"

V0.1	2006-11-1	Initial Release

Powered by LaTeX
generated at 22:01 on May 10, 2010

Contents

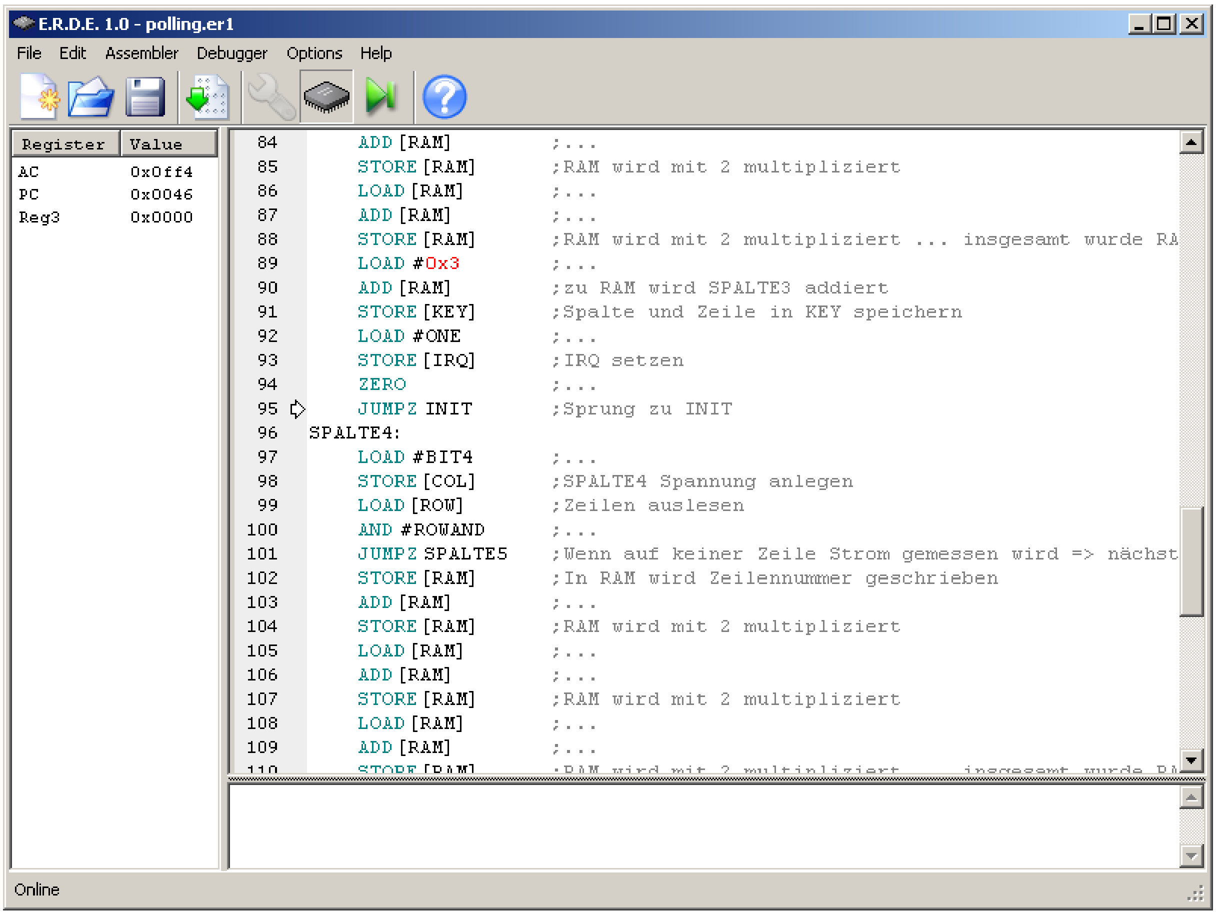
1 Generals

This decument descripes features of E.R.D.E.*Einfachstrechner Development Evironment*, the debugger implementation on the ARM development board and the on Java based compiler.

2 Features

2.1 E.R.D.E.

E.R.D.E. is a all in one solution for creating, debugging and uploading ER1 machine code. Figure 1 shows the E.R.D.E. application. It is an application which is programmed in C#. With ERDE.exe.config which is written in XML, the user can configure the serial port connetion.



Register	Value
AC	0x0ff4
PC	0x0046
Reg3	0x0000

```
        ADD [RAM]           ;...
        STORE [RAM]         ;RAM wird mit 2 multipliziert
        LOAD [RAM]          ;...
        ADD [RAM]           ;...
        STORE [RAM]         ;RAM wird mit 2 multipliziert ... insgesamt wurde RA
        LOAD #0x3           ;...
        ADD [RAM]           ;zu RAM wird SPALTE3 addiert
        STORE [KEY]         ;Spalte und Zeile in KEY speichern
        LOAD #ONE           ;...
        STORE [IRQ]         ;IRQ setzen
        ZERO                ;...
        JUMPZ INIT          ;Sprung zu INIT
    SPALTE4:
        LOAD #BIT4          ;...
        STORE [COL]         ;SPALTE4 Spannung anlegen
        LOAD [ROW]          ;Zeilen auslesen
        AND #ROWAND         ;...
        JUMPZ SPALTE5       ;Wenn auf keiner Zeile Strom gemessen wird => nächst
        STORE [RAM]         ;In RAM wird Zeilennummer geschrieben
        ADD [RAM]           ;...
        STORE [RAM]         ;RAM wird mit 2 multipliziert
        LOAD [RAM]          ;...
        ADD [RAM]           ;...
        STORE [RAM]         ;RAM wird mit 2 multipliziert
        LOAD [RAM]          ;...
        ADD [RAM]           ;...
        STORE [RAM]         ;RAM wird mit 2 multipliziert ... insgesamt wurde RA
```

Figure 1: E.R.D.E. with enabled debug

2.1.1 Debugger

Required: Serial port, Windows XP, Net2.0

Description: With the debugger feature in the E.R.D.E. you can enter any time, even in the runtime of the ER1, into the debugging mode. In this mode you can clock the ER1 manually with a button or a key shortcut. To use the debugging feature correctly the user has to open the same assembler code file which is currently used in the ER1. After you open an assembler code file, a clear text field shows the code. This text field also has a graphical arrow which shows the user exactly the code the ER. is going to execute in the next cycle. Additionally E.R.D.E. shows the most important contents of the ER1 registers and status information like the accumulator or program counter in a list box. Of course the user can leave the debugging mode at any time.

2.1.2 Assembler Editor

Required: Windows XP, Net2.0

Description: E.R.D.E. does not only open already existing assembler code. Users can also use E.R.D.E. to edit, or to develop their own assembler code for the ER1. The syntax highlighting which is specially adapted for the ER1 allows the user to write even bigger code without loosing the overview. It is also possible for users to add highlighting for their own keywords just by changing some values in a xml file. Additionally, E.R.D.E. implements features such as <Undo>, <Save>, <Copy> or <Paste> like every good text editor does.

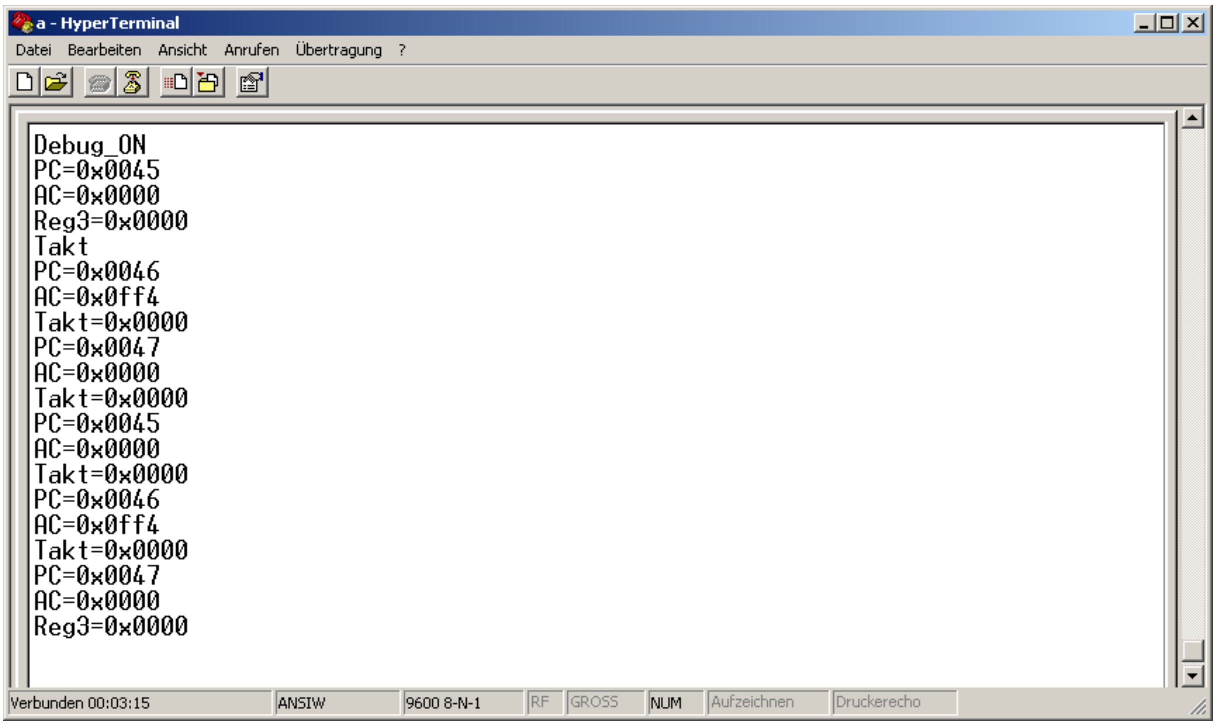
2.1.3 ER1 machine code compiler and uploader

Required: Serial port, Windows XP, Net2.0

Description: With the all-in-one solution E.R.D.E. users can compile and upload assembler code directly to the ER1. over the serial port. It doesn't matter if the assembler code is created in the E.R.D.E. or in another editor. An own designed compiler which is developed in Java, compiles the assembler code into machine code for the ER1. This compiler also throws error messages which are displayed via a message box. This error message box displays a short description and the exact position in the assembler code where the error occured. After the compilation ends successfully, the user can upload the created machine code to the ER1 ROM by using a boot loader routine. Some parts of this feature havent been tested. Tested parts are the communication between the E.R.D.E. and the ARM. We weren't able to write data from the ARM into the ER1 registers which are also used for the bootloader.

2.2 ER1 debugging without E.R.D.E.

A module which is easy to implement in any ARM programm, which doesn't use serial port <UART1>. This module acts like a interface for users who can communicate with it over programms like Hyperterminal as you can see in figure 2.



```
Debug_ON
PC=0x0045
AC=0x0000
Reg3=0x0000
Takt
PC=0x0046
AC=0x0ff4
Takt=0x0000
PC=0x0047
AC=0x0000
Takt=0x0000
PC=0x0045
AC=0x0000
Takt=0x0000
PC=0x0046
AC=0x0ff4
Takt=0x0000
PC=0x0047
AC=0x0000
Reg3=0x0000
```

Verbunden 00:03:15 | ANSIW | 9600 8-N-1 | RF | GROSS | NUM | Aufzeichnen | Druckerecho

Figure 2: Debugging with a Hyperterminal

2.2.1 Debugging with Hyperterminal

Required: Serial port, Hyperterminal

Description A logic which handles interrupts over the serial port makes it possible to debug the ER1 within every program which is able to communicate over the serial port. When ARM is receiving the data, e.g. to start the debug mode, it prints his status informations and also registers from the ER1 back over his serial port. By using this feature you are also able to clock the ER1 and switch its debug mode. A list of output messages are listed in table 1.

3 Examples

For better understanding some of the features are described below in more detail by using some example scenarios.

ARM output	Description
DEBUG_ON	Debugger online
DEBUG_OFF	Debugger offline
Takt	The ER1 PC is inremented by one
AC	The ACC register in the ER1
PC	The PC register in the Er1
Reg3	The status register for the though screen

Table 1: ARM output over the Hyperterminal

3.1 ER1 debugging with E.R.D.E.

Scenario How to enter and exit the debug mode within the E.R.D.E.

1. Connect the serial port COM0 with the serial port <UART1> of the ARM7 evaluation board
2. Connect the ARM7 evaluation board with the power supply
3. Start the E.R.D.E. application
4. Upload the ER1 machine code into the ER1 ROM by using the boot loader
5. Open the same assembler code file which you compiled and uploaded into the ER1
6. Click on the <Start Debugging> button over the menu <Debugger, Start Debugger>
7. Now the ER1 doesn't use his own clock. You can clock the ER1 by clicking on the <Debug Step> button or over the menu <Debugger, Debug Step> or with the very comfortable shortcut <F10>
8. To exit the debugging modus just click on the <Start Debugging> button over the menu <Debugger, Start Debugger> again

3.2 ER1 debugging with Hyperterminal

Scenario How to enter and exit the debug mode within the Hyperterminal

1. Connect the serial port COM0 with the serial port UART1 of the ARM
2. Connect the ARM with the power supply
3. Start the Hyperterminal with 8 databits, no Parity, 1 Stop bit and 9600 baud
4. Make sure a working ER1 is already uploaded
5. To start debugging push the 'g' key to start debugging
6. Now the E.R. doesn't use his own clock. You can clock the E.R. by pushing 'n'
7. To exit the debugging modus just push the 'p' key
8. A overview of the informaion on the hyperterminal you can see in table 1