Using coLinux to develop under Windows XP

A Linux PC is often required to program embedded Linux systems. This can present unexpected problems for many experienced Windows users. A DIL/NetPC-based embedded Linux application can be developed entirely under Windows XP using the SSV coLinux Cross CD-ROM.

Linux as an embedded operating system has a user community that continues to grow. However, the alternative operating system has not been able to establish itself on PCs until now. Microsoft Windows is the standard here and it does not appear that will change in the foreseeable future.

Software development for an embedded Linux system in most cases requires a free license of GCC (GNU C/C++ compiler) and accessories. Many developers find it disconcerting that the GNU tools are primarily available for Linux-based PCs. Any attempt to make GCC run under Windows in order to program an embedded Linux system has been fraught with problems until now. The popular Cygwin – as a Unix/Linux emulation environment under Windows – is much too slow and in most cases causes significant library and version conflicts. Despite the high costs, commercial attempts by some companies to provide GCC as a Windows application have not lead to any practical solutions. Even here, there are typically significant library problems. As a result, many users have no other alternative than to use a Linux distribution as a second operating system on the Windows PC to be able to use GCC tools. Without some effort, this technically straightforward solution is not feasible for large companies in particular, because an IT department is responsible for the PCs. Users are not able to just install a new operating system to boot instead of Windows.

Another obstacle is often times getting familiar with a new PC operating system. Many questions can arise, such as: How can I change the network settings (e.g. IP

address, etc.)? Where can I find a Telnet client? Which editor should I use? How can I later install other software components? The list could go on ad nauseam. The bottom line is that significant orientation time is required for a knowledgeable Windows user to get familiar with the new environment.

1. A Typical DIL/NetPC Development Environment

Figure 1 lists the most important tools and their use in an embedded Linux development environment. A Linux-based PC is used to develop, edit and compile the C/C++ source code using the GNU tools. The test is performed on the target system, in this case an embedded Linux module from the DIL/NetPC family [1]. There is an Ethernet LAN or RS232 connection between the Linux module and the PC. The LAN link is used primarily for file transfer and user dialog. FTP (File Transfer Protocol) or TFTP (Trivial File Transfer Protocol) can be used, for example, through the embedded Linux. Telnet is used for the dialog with the user interface (shell) of an embedded Linux. A Telnet client is started on the PC, which provides a simple (command line) interface (CLI = Command Linux interface) to the embedded Linux in order to communicate with the software components of the embedded system. The necessary server/client components for FTP, TFTP, Telnet, etc. are available on both a PC-based Linux and in nearly any embedded Linux.
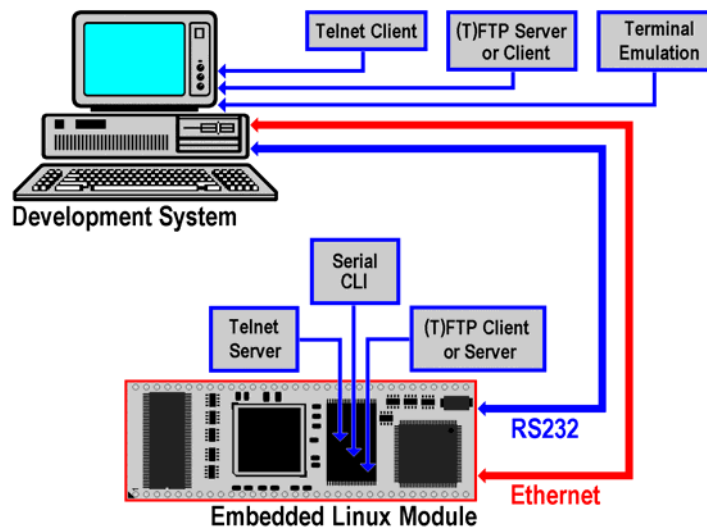

Figure 1: A typical development environment for DIL/NetPCs

In addition to file transfer and the Telnet dialog, the Ethernet LAN connection is also used between both computers for so-called remote debugging. There is no need for a hardware debugger for application-oriented programming of an embedded Linux system. The appropriate components for remote software testing per LAN or RS232 connection are included in the GNU tools with GDB and GDBserver.

With a DIL/NetPC, the RS232 connection between PC and Linux module is also used for CLI-based communication with the boot loader. Such a boot loader (e.g. U boot with the ARM-based DIL/NetPCs or dBUG with a DIL/NetPC with

ColdFire) is available in the Flash of a DIL/NetPC under the embedded Linux operating system.

2. coLinux Replaces the Linux PC

An alternative to using a Linux PC is a free license of Cooperative Linux (coLinux). This Linux project is a new approach to getting the Linux kernel to run under Windows XP [2]. coLinux contains special Windows drivers, which allow it to run under Windows XP as a guest operating system with all privileges. For example, it can directly access the page tables and switch the PC processor between the Windows programs and coLinux. The hardware is virtualized, practically eliminating all access conflicts.
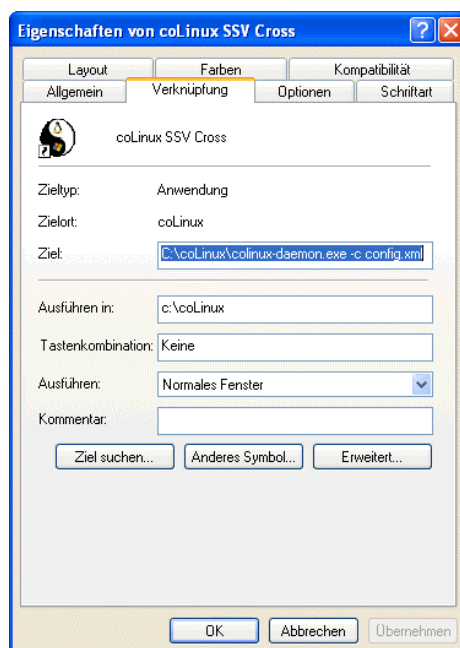
Figure 2: Creating a coLinux icon on the Windows desktop

The **coLinux Cross Tools CD-ROM** has been available from SSV Embedded Systems since June 2005 and makes it possible use coLinux for software development of DIL/NetPC-based embedded Linux applications. The CD-ROM contains a complete preconfigured coLinux for Windows XP PCs in the *SSV-colinux* directory. Older Windows versions are not supported by coLinux.

SSV coLinux installation requires approx. 2 GB of available storage on the Windows XP hard disk drive. The entire installation process is started in a single step: simply launch the (batch) file *SSV-colinux\setupe.bat* in Windows XP (File) Explorer. The default installation directory on the Windows hard disk drive is *c:\Program Files\colinux*. **Change the default to *c:\colinux*.**

After the individual installation steps are automatically executed and the installation program has finished, an icon should be created on the Windows XP

desktop and properly configured for coLinux (Figure 2). The following command line is used to start coLinux:

```
c:\colinux\colinux-daemon.exe –c config.xml
```

The command line should also be saved in the properties of the coLinux icon. However, it can also be used to launch SSV coLinux directly at a Windows command line.

3. coLinux First Impression

The hardware virtualized by coLinux includes a VGA text console (that is displayed in a Windows XP window), network, hard disk drives (as normal files under Windows) and a keyboard. When selecting the virtual hardware, a configuration was consciously selected that is as minimal as possible in order to avoid driver problems in Linux.
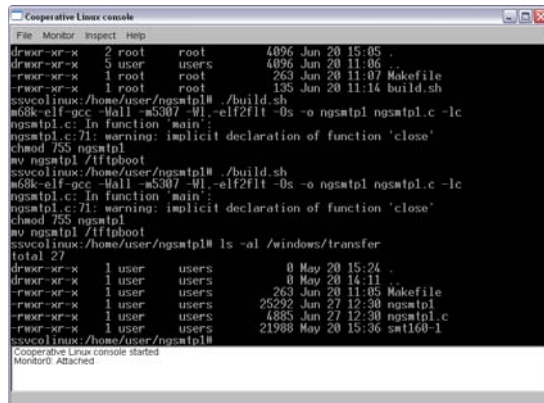

Figure 3: The coLinux VGA text console in Windows XP window

Figure 3 shows the interface of coLinux to be implemented on a Debian Linux under Windows XP. Just a simple mouse click on the previously described icon on the Windows XP desktop is needed to start the configuration. And in just a few seconds, a complete Debian Linux is available in a Windows XP window. The **username** *root* and **password** *root* are used to log in.
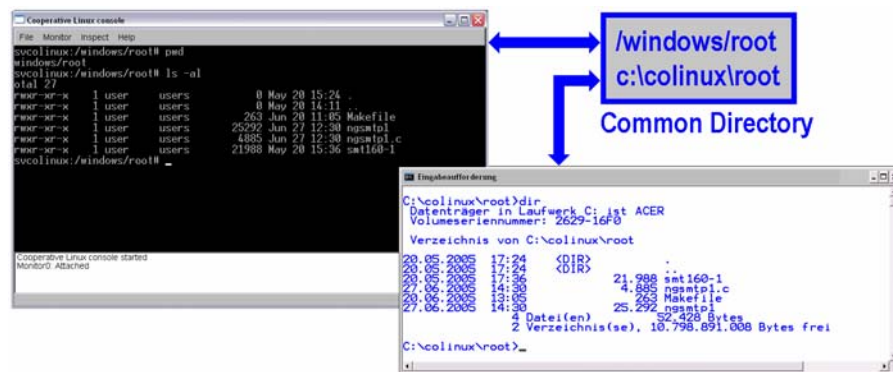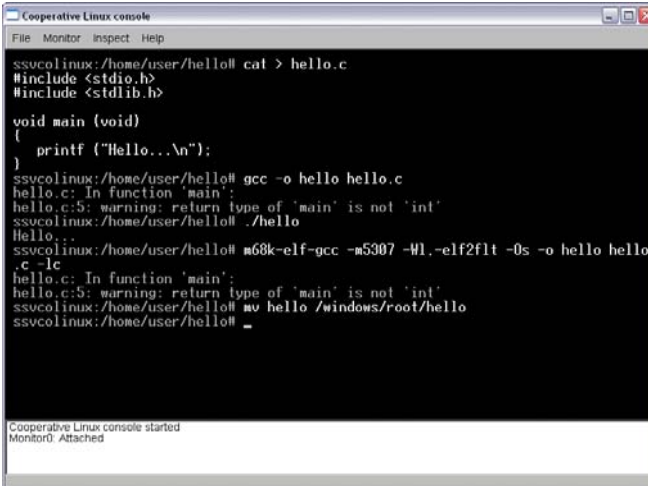

Figure 4: Windows and coLinux use a common directory

Windows and coLinux use a common directory on the hard disk drive (*c:\colinux* from the Windows environment – the same directory is accessible under coLinux under the name */windows*). Any files in the directory are accessible from both Windows and coLinux (Figure 4). However, the common hard disk drive space is managed by Windows exclusively. coLinux does not access the hard disk drive hardware directly. Consequently, there is no need for a separate coLinux partition. A special driver (coFS = coLinux File System) is used by coLinux to access the Windows hard disk drive.

It should be noted, however, that the coFS involves an asynchronous file system. Write operations within the common directory under coLinux or Windows are closed from view of the other operating system only after context is next switched (surrendering control of the CPU to Windows or coLinux). For this reason, no GCC makefile should directly access */windows* under coLinux, if files are used by the makefile that were previously written under Windows.

4. Compiling under coLinux via GCC

A coLinux for embedded Linux development typically contains the original GNU tools for compiling C/C++ programs and corresponding libraries [3]. In addition to the native GCC for x86 code, SSV coLinix contains Cross versions M68K-ELF-GCC (M68K code, output format ELF) and ARM-ELF-GCC (code for various ARM architectures in the ELF output format). The native GCC, for example, can be used to program an ADNP/1520 with an embedded gateway Linux. The M68KELF-GCC lends itself to software development for ColdFire DIL/NetPCs under uClinux (DNP/5280, DNP/5282 and PNP/520). The Linux control DIL/NetPC DNP/7520 is supported by the ARM-ELF-GCC. Other Cross GCC versions can be subsequently added to SSV coLinux, as needed.

Figure 5: Initial steps with the compilers in SSV coLinux

A good first test of the various GNU tools of an SSV coLinux installation is to create and compile a C program in the coLinux window. These steps, for example, can be executed with the following entries:

```
cat > hello.c
#include <stdio.h>
#include <stdlib.h>

void main (void)
{
    printf ("Hello...\n");
}
```
CTRL-D (ends the input of the C source text)
```
gcc –o hello hello.c
```

This input sequence results in a C source code file called *hello.c*. The file is then compiled with the native GCC. The resulting program can be directly executed in the coLinux window. Figure 5 illustrates this example. In this figure, M68K-ELF-GCC is then called up to create an executable for a ColdFire DIL/NetPC from *hello.c*. The uClinux executable *hello* is then moved to the */windows/root* directory. The file can then, for example, be transferred with TFTP to the DNP/5280.

5. Other Auxiliary Tools

There are other tools on the SSV coLinux Cross Tools CD-ROM for DIL/NetPC programming. The opens source editor PN (Programmers Notepad) is of particular importance. The installation program for this auxiliary tool can be found on the CD-ROM in the *ProgrammersNotepad* directory.
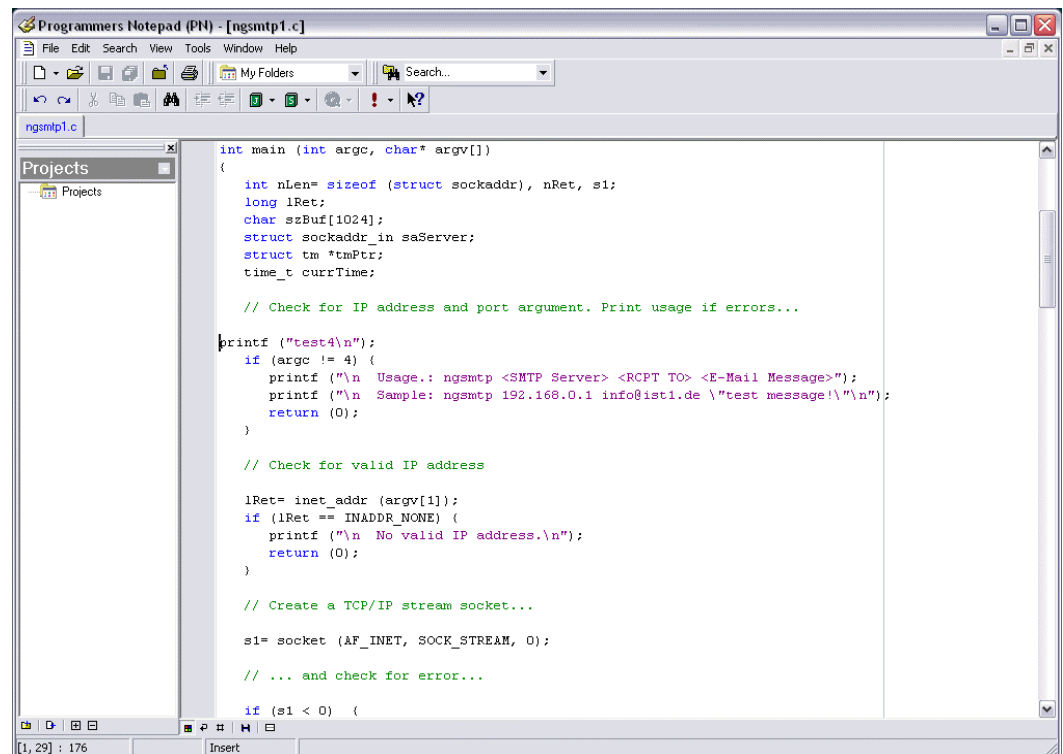

Figure 6: Editing with the open source editor PN (Programmers Notepad)

PN is capable of editing the C/C++ source code for an embedded Linux module in the familiar Windows XP environment. Resulting source code files can be saved in the common Windows directory *c:\colinux*. Additional subdirectories (e.g. *c:\colinux\root* or even *c:\colinux\transfer*) should be created within *c:\colinux* for the individual projects, as needed.

Another important resource on the SSV coLinux Cross Tools CD-ROM is a TFTP server called TFTPD in the *TFTP-Server-Win32* directory. The same server application is also included on the starter kit CD-ROMs for the respective DIL/NetPCs. TFTPD is used for TFTP-based file transfer over an Ethernet LAN connection between PC and embedded Linux module. Figure 7 shows TFTPD in use. It should be noted that this programs requires file transfer from a base directory. The *Browse* button on the TFTP user interface is used to select the directory.

Figure 7: The TFTP server is used for file transfer

All TFTP file transfers to and from DIL/NetPCs are based on the TFTPD base directory. Therefore, *c:\colinux* or a subdirectory within *c:\colinux* must be selected as the base directory to use TFTPD in conjunction with coLinux.

6. The Interaction between Tools

The coLinux window itself, the open source editor PN and a TFTP server TFTP can be used together in such a way that the developer will practically never have to leave the familiar Windows XP environment of the development PC. Figure 8 provides an overview of the interaction between the development tools that are used in the coLinux-based development environment for DIL/NetPCs.

C/C++ source code is edited in Windows using the PN editor. This Windows application organizes the source code files into projects. All source code is saved in the directory *c:\colinux*, or a subdirectory within *c:\colinux*. A script file (build script) is run in the coLinux window to compile the entire project. The script pulls the appropriate source code files from the common directory, activates the GCC using a make file and after successfully running the GCC writes the executable for

the embedded Linux system back to the common directory. The following is an example of such a build script:

```
#!/bin/sh
cp /windows/root/ngsmtp1.c ngsmtp1.c
make all
mv /tftpboot/ngsmtp1 /windows/root/ngsmtp1
mv ngsmtp1.c ngsmtp1.bak
rm ngsmtp1.gdb
```

The second line of the build script copies C source code with the name *ngsmtp1.c* from *c:\colinux\root (/windows/root* in coLinux) to the coLinux directory, in which the build script was launched. The third line activates the GCC make. The GCC make run creates an executable in the coLinux directory */tftpboot*. The fourth line moves the executable to the common directory *c:\colinux\root* (*/windows/root* in coLinux), from where it can then be transferred per TFTP to the DIL/NetPC. Such a build script is executed from the coLinux window.
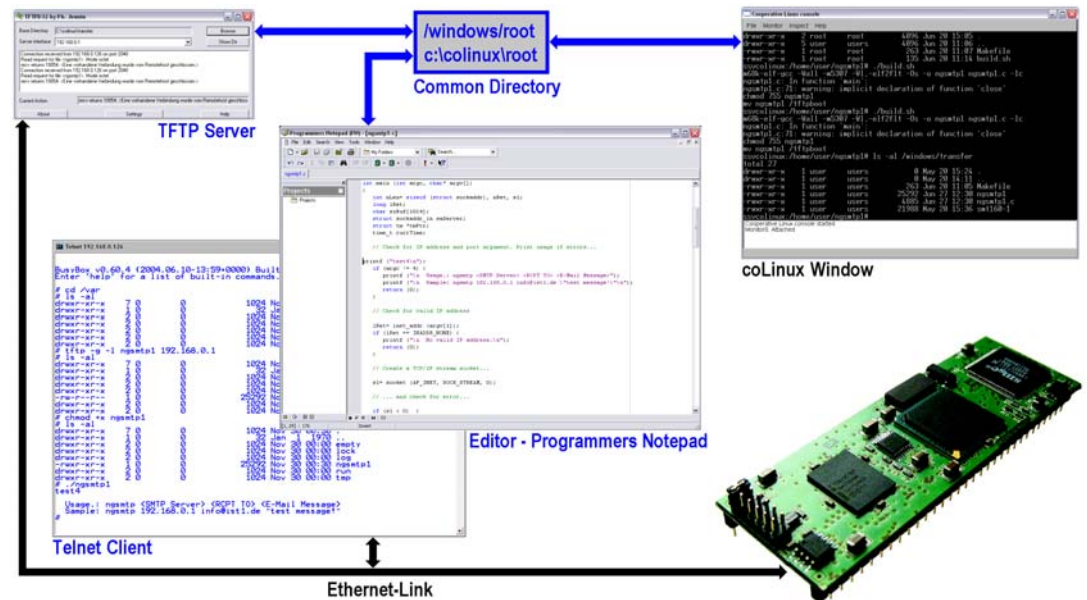

Figure 8: The interaction between development tools

The user only has to launch the script file in the coLinux window. This essentially eliminates any orientation in Linux as a PC operating system. The Windows TFTP server TFTPD and the standard XP Telnet client in Figure 8 are used for file transfer to the embedded Linux system and the shell access.

Figures

Figure 1: A typical development environment for DIL/NetPCs
Figure 2: Creating a coLinux icon on the Windows desktop
Figure 3: The coLinux VGA text console in Windows XP window
Figure 4: Windows and coLinux use a common directory
Figure 5: Initial steps with the compilers in SSV coLinux
Figure 6: Editing with the open source editor PN (Programmers Notepad)
Figure 7: The TFTP server is used for file transfer
Figure 8: The interaction between development tools

References

[1] Web site on DIL/NetPCs: www.dilnetpc.com
[2] Web site on coLinux: http://www.colinux.org/
[3] Walter: Messen, Steuern, Regeln mit ARM-Mikrocontrollern. Franzis´ 2004.

Contact

SSV Embedded Systems
Heisterbergallee 72
D-30453 Hanover, Germany
Tel.: +49-(0)511-40000-0
Fax: +49-(0)511-40000-40
E-mail: sales@ist1.de
Web: www.ssv-embedded.de
Web: www.dilnetpc.com

Notes about This Document (coLinux-APN1.doc)

Revision	Date		Name
1.00	01.07.2005	Initial draft	KDW
1.01	09.09.2005	Some spelling mistakes corrected / engl. translation.	KDW