



Product:	CrossTools
Product version:	
Document ID:	UM-CrossTools
Doc revision:	PA2
Written/Aprr.:	NPE /
Date:	03.11.2008

Industrial Control Design AS



CrossTools

How to compile and test Linux applications in Windows

The content of this document is confidential information not to be published without the consent of Industrial Control Design AS.

Industrial Control Design AS, www.icd.no, support@icd.no, forum.icd.no

Contents

1. INTRODUCTION.....	3
1.1. About.....	3
1.2. Prerequisites.....	3
<hr/>	
2. COMPILING FOR LINUX.....	4
2.1. Using the MakefileWizard.....	4
2.1.1. About.....	4
2.1.2. Step by step example.....	4
2.2. Using the CDPToolchain.....	5
2.2.1. About	5
2.2.2. Step by step example.....	5
<hr/>	
3. REMOTE LINUX TESTING.....	8
3.1. Uploading of applications to remote hardware.....	8
3.1.1. About.....	8
3.1.2. How to copy the Application folder to a host named Debian.....	8
3.1.3. How to copy a single file to a host named Debian.....	8
3.1.4. Common mistakes.....	8
3.2. Using putty to execute an application.....	8
3.2.1. About.....	8
3.2.2. Using putty to connect a host named Debian.....	9
3.2.3. Preparing to run the application.....	9
3.2.4. Executing the CDP application.....	10
3.2.5. Final and EXTREMELY important notes.....	10

1. Introduction

1.1. About

This document describes how users can compile and test their applications for Linux in Windows.

1.2. Prerequisites

- The CDP MakefileWizard
- The CDP Toolchain

2. Compiling for Linux

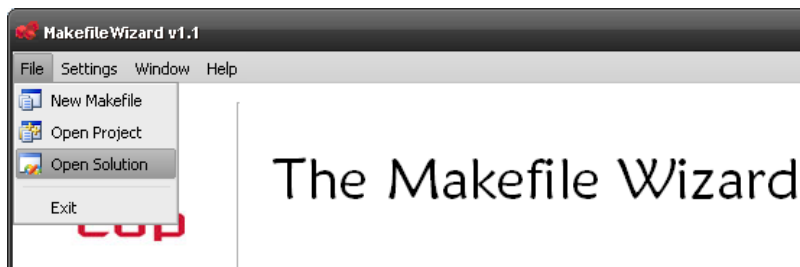
2.1. Using the MakefileWizard

2.1.1. About

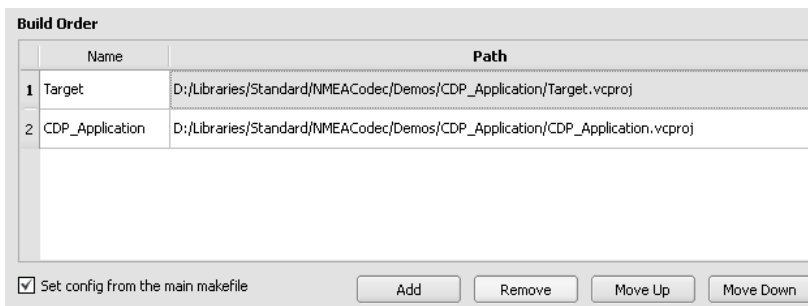
When you have a fully working Windows application, it is time to test that the application also compiles with a Linux compiler. That is, a compiler that actually follows the C++ standards. This section explains how to use the CDP MakefileWizard to convert the visual studio project files into gnu makefiles.

2.1.2. Step by step example

- Start the newest version of the MakefileWizard (currently 1.1).
- Open the project solution file:



- Remove the Target project as it is not required on Linux. This is done by selecting any column in that line and then clicking the button named “Remove”:



- Check that the configuration is correct and hit the “Generate All” button to create the makefiles.
- The MakefileWizard resolves linked libraries by parsing the dependency information provided in the solution file. It handles both the dependency scheme of VS2002 and the one included in the newer versions of VS. The library input, however, is ignored. This is done due to the fact that it includes lots of Windows libraries that would result in linker errors. Hence, users will have to manually add such libraries to CDP_Application.mak. The operation is briefly described in the following section.

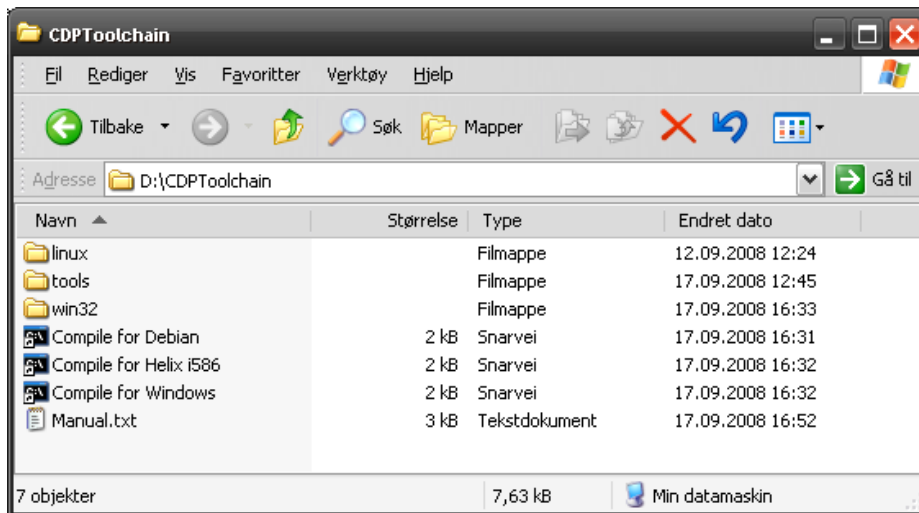
2.2. Using the CDPToolchain

2.2.1. About

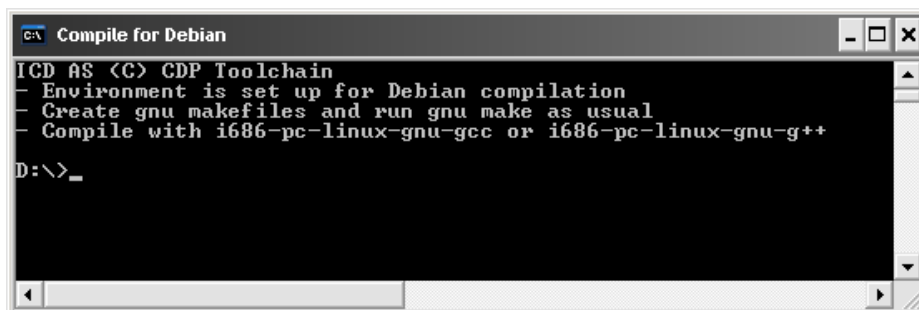
Install the CDPToolchain as described in its Manual.txt. Then see this guide for a step by step example on how to compile makefiles created by the MakefileWizard.

2.2.2. Step by step example

- Browse to the CDPToolchain directory and execute one of the batch scripts depending on what Linux distribution you want to compile for. Usually, that is the batch named “*Compile for Debian*”.



This will open command prompt with the correct environment variables:



- The makefiles that are created with the MakefileWizard are set up with `g++` as default compiler. When using the CDPToolchain you will need to change this depending on the batch file you've chosen. In the above example, `g++` has to be replaced by `i686-pc-linux-gnu-g++`. The same goes for the archiver (`ar`) that has to be prefaced with `i686-pc-linux-gnu-`, similar to the compiler option. Both are changed in the main makefile (the one named Makefile and without .mak extension). Note that it is also possible to set this configuration with environment variables. In such a scenario, the export functions in the main makefile must be removed.

```

#
# ICD AS (C) 2007 - MAIN MAKEFILE CREATED USING CDP MAKEFILE WIZARD
#
export CFG=Release
export COMPILER=i686-pc-linux-gnu-g++
export AR=i686-pc-linux-gnu-ar
export ARFLAGS=rsc
...
    
```

- GNU Make does not like paths with spaces. Thus, if the CDPBase variable of your system is set to *C:\Programfiler\CDP Developer*, change it to "*C:/Programfiler/CDP Developer*" by running *SET CDPBase=%CDPBase%*. This rule also goes for any other environment variable that is used in your project. To make the changes permanent, you might want to add this to the different toolchain batch files that are located in CDPToolchain/win32. The batch file for the Debian toolchain is named *gcc-4.1.1-glibc-2.3.6.bat* while the Helix batch is named *gcc-4.1.1-glibc-2.6.1.bat*.

```
@SET CDPBase="%CDPBase%"
@SET CDPBase_Develop="%CDPBase_Develop%"
...
```

- Some users might experience problems with the above depending on their Windows version, or perhaps their dos prompt configuration. In such badly configured system, GNU Make might interpret the "" around the path in CDPBase as the actual command. Thus, try to avoid environment variables in front of commands, and either hard code the path, or add it to PATH.

```
# Try to avoid this and rather add $(CDPBase)/Tools to PATH
-$(CDPBase)/Tools/validateLicense Debug/CDP

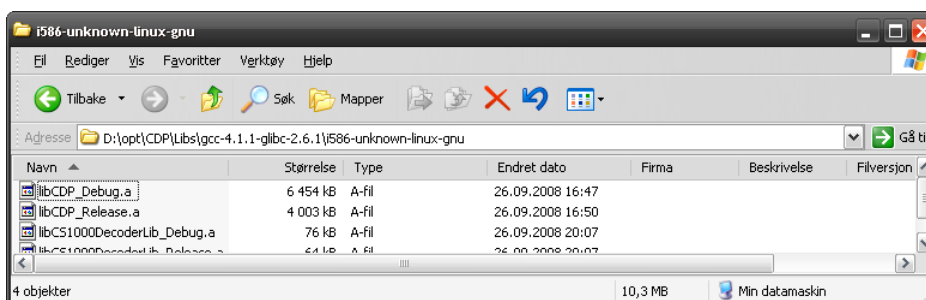
# You are then allowed to write the following instead
-ValidateLicense Debug/CDP
```

- GNU Make does not automatically create missing directories. Thus, if target is set to Debug/CDP you will have to manually create the Debug directory (with *mkdir*) if it doesn't exist. Else, you will get prompted with error messages like: *cannot open output file Debug/CDP: No such file or directory*.

- Recall from the MakefileWizard section that input libraries must be manually included in *CDP_Application.mak*. Such libraries are added in the LIBS section for both debug and release (e.g. *LIBS= -ICDP_Release -ISerialStringLib_Release -lrt -lpthread -lm*). Note that libraries are prefixed with *-l*, and added without *.a* at the end, even though the name of the library actually starts with *lib*.

```
...
LIBS= -ICDP_Debug -ISerialStringLib_Debug -lrt -lpthread -lm
...
```

- Remember that you need to download and unzip the Linux release of both CDP and required addons. In addition, you must make sure to copy the libraries that are correct for the distribution you are compiling for. Recall that *gcc-4.1.1-glibc-2.6.1* is equivalent to Helix, while Debian is represented by *gcc-4.1.1-glibc-2.3.6*. Note that libraries must be placed in *%CDPBase%/Libs*.



- Finally, type *make release* or *make debug* to start compiling:

```

ICD AS (C) CDP Toolchain
- Environment is set up for Debian compilation
- Create gnu makefiles and run gnu make as usual
- Compile with i686-pc-linux-gnu-gcc or i686-pc-linux-gnu-g++

D:\>cd Libraries\Standard\NMEACodec\Demos\CDP_Application

D:\Libraries\Standard\NMEACodec\Demos\CDP_Application>mkdir Release

D:\Libraries\Standard\NMEACodec\Demos\CDP_Application>make release

rm f Release/CDP
rm: cannot remove `f': No such file or directory
rm: cannot remove `Release/CDP': No such file or directory
make: [release] Error 1 (ignored)
make -C . -f CDP_Application.mak CFG=Release
make[1]: Entering directory `D:/Libraries/Standard/NMEACodec/Demos/CDP_Application'
i686-pc-linux-gnu-g++ -W -fexceptions -finline-functions -D_LINUX -DNDEBUG -D_THREAD_SAFE -D_LIB -D_MBCS
-fpack-struct=4 -I"C:\Programfiler\CDP_Devel
oper"/NMEACodec -I"C:\Programfiler\CDP_Developer"/CDP -L"C:\Programfiler\CDP_Developer"/Libs -o CDPMain.o
-c CDPMain.cpp
i686-pc-linux-gnu-g++ -W -fexceptions -finline-functions -D_LINUX -DNDEBUG -D_THREAD_SAFE -D_LIB -D_MBCS
-fpack-struct=4 -I"C:\Programfiler\CDP_Devel
oper"/NMEACodec -I"C:\Programfiler\CDP_Developer"/CDP -L"C:\Programfiler\CDP_Developer"/Libs -o
CDPStarter.o -c CDPStarter.cpp
i686-pc-linux-gnu-g++ -W -fexceptions -finline-functions -D_LINUX -DNDEBUG -D_THREAD_SAFE -D_LIB -D_MBCS
-fpack-struct=4 -I"C:\Programfiler\CDP_Devel
oper"/NMEACodec -I"C:\Programfiler\CDP_Developer"/CDP -L"C:\Programfiler\CDP_Developer"/Libs -o StdAfx.o -c
StdAfx.cpp
i686-pc-linux-gnu-g++ -W -fexceptions -finline-functions -D_LINUX -DNDEBUG -D_THREAD_SAFE -D_LIB -D_MBCS
-fpack-struct=4 -I"C:\Programfiler\CDP_Devel
oper"/NMEACodec -I"C:\Programfiler\CDP_Developer"/CDP -L"C:\Programfiler\CDP_Developer"/Libs CDPMain.o
CDPStarter.o StdAfx.o -o Release/CDP -ICDP
_Release -lSerialStringLib -Release -lrt -lpthread -lm
make[1]: Leaving directory `D:/Libraries/Standard/NMEACodec/Demos/CDP_Application'
"C:\Programfiler\CDP_Developer"/Tools/ValidateLicense/Release/CDP
ValidateLicense.exe(v2.5) : Updating executable 'Release/CDP'
ValidateLicense.exe(v2.5) License: CDP_Developer License key. Do not edit or modify!

ValidateLicense.exe(v2.5) License:
ValidateLicense.exe(v2.5) License: Registration information:
ValidateLicense.exe(v2.5) License:
ValidateLicense.exe(v2.5) License: Developer : Nils-Petter Eftedal
ValidateLicense.exe(v2.5) License: MAC : 00-30-05-CB-18-62-00-00
ValidateLicense.exe(v2.5) License: Company : ICD AS
ValidateLicense.exe(v2.5) License: Branch : Software
ValidateLicense.exe(v2.5) License: Product : CDP , Expiry date (dd.mm.yyyy): 16.08.2009
ValidateLicense.exe(v2.5) License: Product : CDPsim , Expiry date (dd.mm.yyyy): 16.08.2009
ValidateLicense.exe(v2.5) License: Product : RedundancyLib , Expiry date (dd.mm.yyyy): 16.08.2009
ValidateLicense.exe(v2.5) License: Product : CDPLinux , Expiry date (dd.mm.yyyy): 16.08.2009
ValidateLicense.exe(v2.5) License: Product : SNMPManagerLib , Expiry date (dd.mm.yyyy): 16.08.2009
ValidateLicense.exe(v2.5) License: Product : CDPDAQLib , Expiry date (dd.mm.yyyy): 16.08.2009
ValidateLicense.exe(v2.5) License: Product : CS1000DecoderLib , Expiry date (dd.mm.yyyy): 16.08.2009
ValidateLicense.exe(v2.5) License: Product : MSILib , Expiry date (dd.mm.yyyy): 16.08.2009
ValidateLicense.exe(v2.5) License: Product : SerialStringLib , Expiry date (dd.mm.yyyy): 16.08.2009
ValidateLicense.exe(v2.5) License:
ValidateLicense.exe(v2.5) : Executable was updated
cp Release/CDP ../Application/CDP

D:\Libraries\Standard\NMEACodec\Demos\CDP_Application>

```

3. Remote Linux testing

3.1. Uploading of applications to remote hardware

3.1.1. About

The CDPToolchain includes tools from the putty package for transferring data using scp. Type pscp for information about usage and options, and see the following sections for a quick guide.

3.1.2. How to copy the Application folder to a host named Debian

The following line will copy a local directory named Application to directory Test/MyInitials on the host named Debian. The data will get copied to the home directory of user nisse. When executing the command you will get prompted for a password provided by yours truly. Finally, remember to replace MyInitials with the initials of your own name.

```
pscp -scp -r Application nisse@Debian:Test/MyInitials
```

3.1.3. How to copy a single file to a host named Debian

After copying an application folder as described in the above section, it might be useful to copy single files, for example when wanting to test the debug version of an application. Single files are copied just like folders except for the -r option. The following line copies an executable named CDP to the same folder that was copied earlier.

```
pscp -scp CDP nisse@Debian:Test/MyInitials/
```

3.1.4. Common mistakes

There are a few mistakes that are pretty common when copying files and folders in Linux. First of all, it is important to remember to add “/” after a target directory if you don't want to rename the copied folder. In the first example with pscp, we wanted to rename the Application directory to MyInitials. That is why we didn't add “/” at the end of the directory name. In the second example however, the “/” was added. Without the slash at the end, the CDP executable would've been renamed to MyInitials and placed in the Test directory. Thus, be careful when adding the target path.

3.2. Using putty to execute an application

3.2.1. About

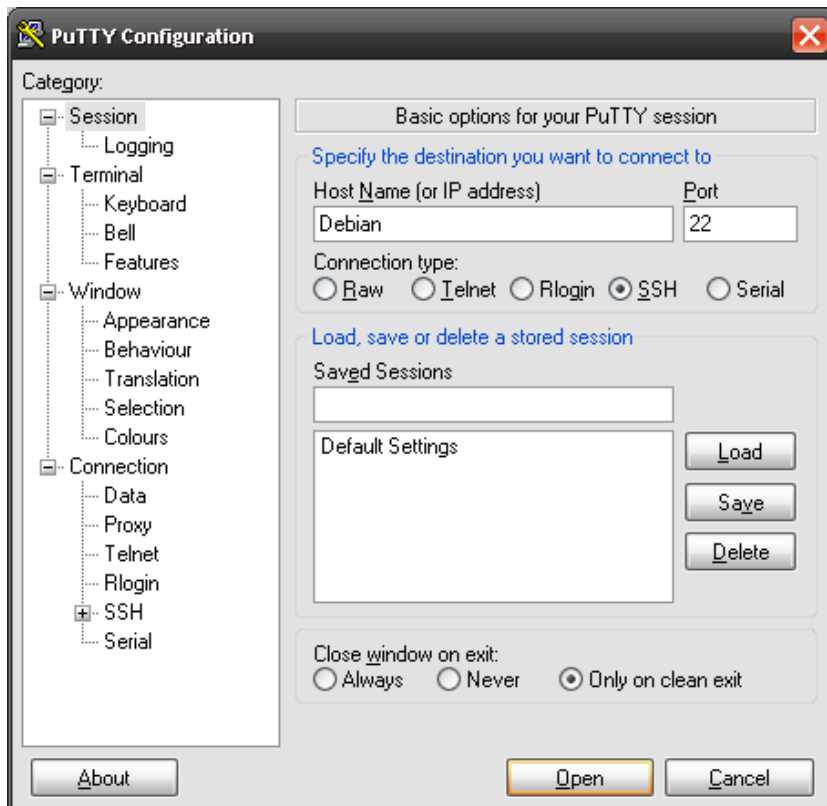
Fire up your favorite terminal with support for ssh and follow the provided instructions. The following sections explains how to use putty for logging into a remote host, but the steps will be just as simple with any other tool.

Get putty from \\icdserver\ICD\Software or download it from putty.org

The program does not require any installation and is started by double-clicking the executable.

3.2.2. Using putty to connect a host named Debian

Enter login information as depicted below, and enter user name and password when prompted.



3.2.3. Preparing to run the application

When logged in, type `cd /Test/MyInitials` to go to the directory where the files were copied. Type `ls` to show files in the current directory. Finally, we need to ensure that the file can be executed. For this purpose we use the `chmod` command: `chmod +x CDP`.

3.2.4. Executing the CDP application

When logged in as described in the previous section, it is time to execute your application. CDP applications are run with a real time scheduling algorithm and thus requires root access (ensured with sudo). As long as the directory has not been added to path you must also specify that the executable you want to run is in the current directory. Hence, execute CDP executables with the following: `sudo ./CDP`.

```

login as: nisse
nisse@Debian's password:
Linux debian 2.6.23 #1 SMP PREEMPT Thu Oct 23 13:46:15 CEST 2008 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Oct 29 12:47:40 2008 from zorro.local
nisse@debian:~$ cd Test/MyInitials
nisse@debian:~/Test/MyInitials$ sudo ./CDP

Binary image build date: Oct 29 2008 13:36:27

CDP Version 2.3.1.0 beta 38.

Libraries included in build:
SerialStringLib build date: Oct 29 2008 13:36:27 Version 1.3 (Built with CDP 2.3 .1.0 beta 38)
CDPLinux build date: Oct 23 2008 13:34:06 Version 2.3.1.0 beta 38

* Wed Oct 29 2008 *
14:33:37 Serial COM1: 4800 8 N 1
14:33:37 NMEACodec_Debian: WorkerTask started.
14:33:37 Messenger is using ip-address 10.0.2.225:2041 (subnetmask 255.255.255 .0).
14:33:37 Connecting App 11: Runestation_CDPBrowserM Ip: 10.0.2.160:2040
14:33:37 Connecting App 13: Icdsveinglam_CDPBrowserM Ip: 10.0.2.227:2041
14:33:37 Connecting App 4: CDPSafe_Lab4 Ip: 10.0.2.33:2040
14:33:37 Connecting App 2: WAGOIPC_APP Ip: 10.0.2.36:2040
14:33:37 Connecting App 6: LabApp6 Ip: 10.0.2.35:2040
14:33:37 Connecting App 1: RDApplication1 Ip: 10.0.2.30:2040
14:33:37 Connecting App 3: LabApp2 Ip: 10.0.2.50:2040
14:33:37 Connecting App 55: RedundantApp Ip: 10.0.2.30:2040
14:33:37 Connecting App 7: Icdsveinglam_TutorController Ip: 10.0.2.227:2040
14:33:38 WebServer: Starting WebServer on 10.0.2.225:80
14:33:39 SerialStringDispatcher_Client: ReceiveTask Started.
14:33:39 SerialStringDispatcher_Client: SendTask Started.
14:33:39 Messenger: Saving new application handle = 5
  
```

3.2.5. Final and EXTREMELY important notes

- Make sure that you are the only one to use the Debian host when running important performance tests.
- Quit any running application before exiting putty (recall that CDP quits when pushing 'q')!!!
- ALWAYS RUN MAKE CLEAN BEFORE BUILDING FOR A DIFFERENT CONFIGURATION!!!
 > *make debug && make clean && make release*