INTERNICHE'S C COMPILER AND TOOLS TECHNICAL NOTE

Introduction

This document provides guidelines on how to build the InterNiche sources.

InterNiche software is designed to be compiled on a variety of development systems and for a wide range of target systems using any ANSI C Compiler. Makefiles are also provided which are designed to work with standard make utilities.

Starting from this base, experienced programmers should find it easy to move to their specific development, target and compiler systems. However, when we build a source tree for shipment to a customer, we have to pick one of our many supported configurations for the files we ship.

For this release, we have selected DOS and Windows 95/98 as both development and target systems. Even if your target system is not DOS or Windows 95/98, we recommend you build and test your software under DOS or Windows 95/98 before starting your port. We've selected several popular DOS and Windows 95/98 C compilers (and their associated tool-sets) to be supported "out of the box".

Each compiler is supported by the inclusion of a **cflags** file, which is included in all the subdirectory makefiles and sets the compiler commands, options, etc.

Microsoft	8.0c (VCC 1.x)	**nmake**	**cflags.mc8**	**masm**	**DOS target**
Microsoft	Visual C++ 6.0	**nmake**	**cflags.m32**	**nasm**	**Win 95/98 target**
Borland	4.x	**make**	**cflags.bc4**	**tasm**	**DOS target**

Although the builds for DOS systems can usually be done in a Windows 95 or NT environment, the applications themselves usually require a plain DOS environment when they execute. This is because the "protected" mode feature of Windows makes it difficult for "DOS box" applications to get at hardware devices such as networking cards. A side effect of this is that you should not build these executables with 32 bit versions of the compilers - after all, **DOS is a 16 bit system**.

The builds for Windows 95/98 can be done in Windows 95/98 environment and the application can be run in a Windows 95/98 console.

To build your demo executable:

1. Pick a working directory to be ***\root*** of your demo system source tree.
2. Unzip the distribution file in the ***\root***, being careful to preserve sub-directories.
3. Copy the correct **cflags.xxx** file in the ***\root*** demo directory into **cflags.mak** also in ***\root***.
4. Run the make utility (either Borland **make** or Microsoft **nmake**) in the TARGET directory.

Specific TARGET directories are provided for various platforms (please see the table below).

The **make** should recurse into and build all dependent files. The executable file should be created in the TARGET directory.

Target Directory	Description
ace360	Build for ACE360 based board with SDS Tools (**cflags.sds**)
armarm	Build for ARM PID based board with ARM Tools (**cflags.arm**)
armgh	Build for ARM PID based board with GHS Tools (**cflags.gh2**)
dosmain	Build an executable for DOS (**cflags.mc8** or **cflags.bc4**)
mpc860	Build for MPC860 (MBX) based board with GHS Tools (**cflags.gh1**)
net186	Build for NET186 based board (**cflags.n86**)
vrtxsa86	Build for VRTX SA86 (**cflags.m86**)
cs89712	Build for Cirrus Logic's cs89712 ARM7 development board (**cflags.gh2**)
win32	Build for Microsoft Windows (**cflags.m32**)

The build process for releases starting with 1.8 is different from earlier releases. In prior releases, port dependent files such as **ipport.h** and **webport.h** were named and edited in the TARGET directory as **ipport.h_h** and **webport.h_h**. At build time these files were copied to **..\h_h** directory and renamed as **ipport.h** and **webport.h** etc. In release 1.8 this process is still the same with the one difference that the **..\h_h** directory has now been renamed to **..\h** directory. All the relevant header files are now in this directory. This process of editing the **ipport.h_h** etc files in the TARGET directory and then copying them at build time allows for providing port dependent files for various targets within the same source tree.

Please note that the sources shipped will be configured for the requested TARGET. In order to build for other TARGETs, you will have to do the configuration. This will require setting appropriate options in **ipport.h_h** (enabling/disabling modules) and **makefile** (linking appropriate modules).