xHarbour.com

xHarbour Builder

Getting started manual

for Microsoft Windows
(version 1.1, April 2004)

Patrick Mast

Content

1. Preface

1.1. Introduction

Welcome to xHarbour Builder!

This is the “Getting started” manual for xHarbour Builder. It will help you to get started with xHarbour Builder for the Windows platform. If you use xHarbour Builder on a different platform, please consult the appropriate xHarbour Builder “Getting started” manual for your chosen platform.

A more in-depth technical document can be found in our “Programming Guide for xHarbour Builder” which will be a programming guide independent of your operating system.

For those with a CA-Clipper background, this manual will also show how to migrate your legacy applications to 32bit with a minimum of pain, while retaining simultaneously your ability to maintain the same source code as a 16 bit application. This way, you will not have to face the difficulty of maintaining two sets of source code for 16 bit and 32 bit environments.

If you are using CA-Clipper and FiveWin to build your Windows applications, you will see how to easily convert your 16 bit .DLL screen resources into an .RC file that will be automatically integrated into your application by xBuild, xHarbour Builder’s Project builder.

xHarbour originated as an “Open Source - Free Software project”. “Open Source” and “Free Software”, are terms referring to a new trend in the software market, publicized first by software products such as the “Linux Operating System”. The term “Free Software” is not intended to suggest “free of cost”, but rather “free” as in “freedom”. Free to be modified, extended, etc., as well as the freedom to sell. The “Free Software” concept is actually not new, but became more popular in recent years as a direct result of the phenomenal success of Linux, and related products.

xHarbour Builder is xHarbour.com Inc.’s distribution of the free open source xHarbour. So, how did xHarbour Builder actually get started? The xHarbour Project was started by Ron Pinkas in late 2001, as a separate project from the Harbour Project (http://www.harbour-project.org) where he was one of the leading developers. xHarbour was established to provide as a more aggressive alternative to the conservative development style of the Harbour Project.

Shortly thereafter, xHarbour has fulfilled its promise, and introduced numerous new features, performance improvements, and greater levels of reliability compared with the original Harbour compiler.

Soon, It became clear to Ron Pinkas that xHarbour could benefit greatly from commercial distribution and support. Early in December 2002, Ron Pinkas contacted Patrick Mast to discuss that idea and shortly after xHarbour Inc. was established.

xHarbour.com Inc. aims to establish xHarbour as the Development Product of choice, for the extended xBase Developer Community. xHarbour.com Inc. attempts to follow in the footsteps of Red Hat Inc and similar companies, merging the stability and longevity that commercial distribution offers with the freedom and security offered by Open Source - Free Software projects. This powerful combination has already demonstrated its tremendous market appeal and the viability of this business model.

1.2. Acknowledgments

Firstly, I would like to thank ALL of the volunteer workers who have contributed to programming the core xHarbour compiler. The list is way too long to print here. They are great guys from all over the world: Argentina, Brazil, Canada, France, Germany, India, Indonesia, Italy, Nederland, Korea, Poland, Russia, Spain, US …please forgive me for not being able to list all. THANK YOU!

I would also like to thank Ron Pinkas for his great work on the xHarbour Builder. Without him, xHarbour would not be where it is now. The time he devoted to xHarbour is simply amazing. Thanks Ron!

Next I'd like to thank Ross McKenzie for making my so technical English into an enjoyable and fun reading English. Don't underestimate his job! ;-)

My co-workers at the office are working even harder than usual while I'm spending more and more time writing this book. Normally, I'm supposed to be helping them with the day-to-day business at the office. Thank you ALL for the terrific job you are doing. Without your help, I would not have the time and energy to finish this book.

I would also like to thank all of my Internet friends who have helped me in learning and gathering technical information on the subjects covered by this book. They have been helping me for the last 20 years now. I know it's a give and take game, but I feel that I have received more than I will ever be able to give. Let's hope you guys and girls find this book useful and interesting. This book is for YOU!

Lastly, I would like to thank my wife Hilde who is the most important person in my life. She has made writing this book possible. Her superhuman abilities as a mother help me get past the natural guilt that comes from being away from my kids for so long while I struggle along in my office cranking out the xHarbour documentation. Without her support, I'd not be able to complete this book.

1.3. Tell us what you think!

As the reader of this book, you are our most important critic and commentator. I value your opinion and want to know what I'm doing right, what I could do better, what areas you'd like to see covered in a future book, and any other words of wisdom you're willing to pass my way.

I tried to prepare this book carefully; however I'm only human and there is only so much time available for writing and tweaking. So, you may find some errors, inconsistencies, or subjects that could be described better.

You can write to me directly or email to patrick@xHarbour.com to let me know what you did or didn't like about this book—as well as what we can do to improve our future books.

Please note that I cannot personally help you with technical problems related to the topic of this book. You should use our xHarbour's technical newsgroups for direct technical support. Also note that due to the high volume of email I receive, I might not be able to reply to every message personally. I apologize for this in advance.

When you write, please be sure to include this book's title as well as your name. I will carefully review your comments and handle them with the most attention. Thank you.

Patrick Mast
April 2004

2. Installing xHarbour Builder

This chapter provides complete instructions on how to install xHarbour Builder on a Windows based computer.

For installation on another operating system, please consult the appropriate “Getting started” manual for your operating system.

2.1. What’s in the box?

The list of contents for the xHarbour Builder box is available on the xHarbour Builder CD. Look for the file “package.txt” in the root directory of the xHarbour Builder CD.

If you have an “Update and service subscription” plan from xHarbour.com Inc., you can also download your installation program from our web site.

2.2. Download xHarbour Builder from the web

The xHarbour Builder installation program can also be downloaded from the xHarbour.com web site. Go to http://www.xHarbour.com and login to your “My xHarbour” account using your user name and password. If you have forgotten your password, click on the “Forgot password?” link. xHarbour.com will send you an email with your current password.

Once logged on, click on the “My downloads” link in the top menu of your account. Here you will find a list of all currently available xHarbour Builder versions. Click the Windows logo of the xHarbour Builder version you have purchased. The download of the installation program will start. You may save the downloaded installer on your system or you may click “Open” to start the installation.

The installation program will be available for download for a period of 30 days after your purchase date and/or as long as you have a subscription on xHarbour.com’s “Update and Service Subscription”. For more details on this subscription, go to http://www.xHarbour.com/subscription

2.3. Your personal xHarbour Builder Product Key

Before you can install xHarbour Builder you need to get your personal xHarbour Builder Product Key. This key can be generated from the “My downloads” page of the “My xHarbour” page on xHarbour.com website.

See section 2.2 on how to connect to the “My downloads” page at xHarbour.com.

At the bottom of the “My downloads” page, you will find the “Generate key” button. Clicking the button will start the generation of your personal xHarbour Builder Product Key. Please keep your product key in a safe place, as it is your **personal** key. Do **NOT** share this key with anyone else, as doing so will violate your xHarbour Builder License.

Once generated, the key is saved in your profile at “My xHarbour”. You may retrieve this generated key at any time by clicking the “Generate key” button.

Your xHarbour Builder Product Key only changes if you upgrade your xHarbour Builder version. If for example, you upgrade your Personal xHarbour Builder to a Professional version, you will need to generate a new xHarbour Builder Product Key. Once upgraded, your previous xHarbour Builder Product Key will expire and should be discarded.

2.4. Starting the installation program from CD-ROM

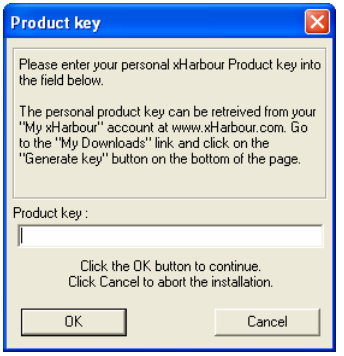
If you want to start the installation program from the CD-ROM, please put the CD-ROM in your CD drive.

The xHarbour Builder CD is an auto start CD and should therefore start automatically with the “Installation menu”. If it does not start automatically, go to “My computer” and select the CD drive holding the xHarbour Builder CD. Open the CD drive and start “Autorun.exe” which is in the root directory of the CD-ROM.

2.5. Installing xHarbour Builder

If you saved the downloaded installation program from xHarbour.com’s web site, please start it from the location where you saved it on your system.

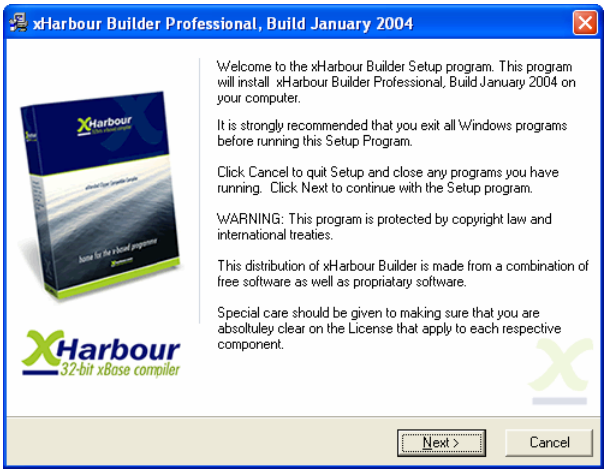
If you are installing xHarbour Builder from the CD-ROM, select your xHarbour Builder version from the given menu. The following “Product key” screen will appear:



Enter your personal xHarbour Builder Product Key in the “Product key” field.

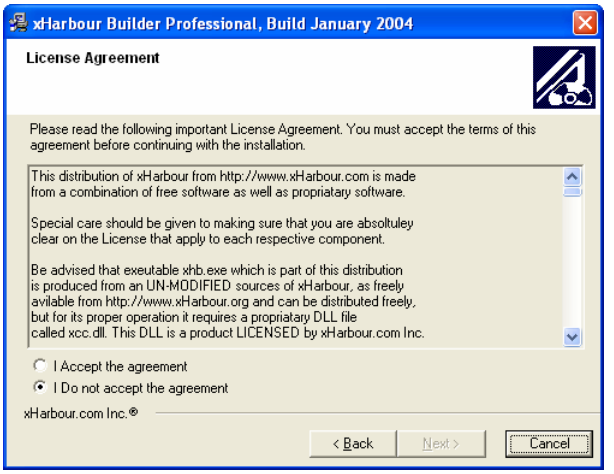
Note:
If you copy/paste the product key from the web site, be very careful not to add any spaces before or after the product key as this will change your product key. The xHarbour Builder Product key should be entered ***exactly*** *as shown on your “My downloads” page. See section 2.3 for more information about the xHarbour Builder Product key.*

Entering a valid product key will start the actual installation. Following screen will appear:



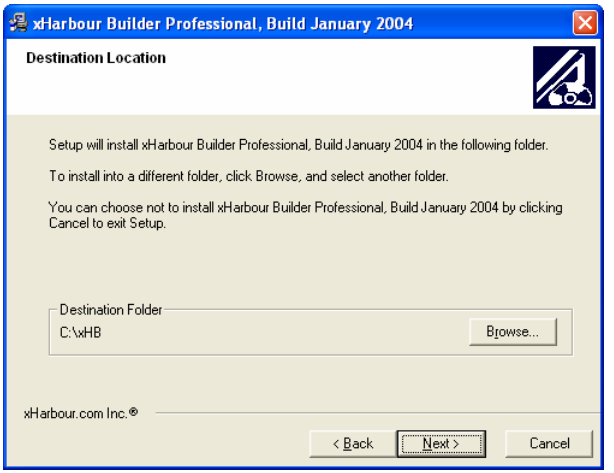
Details of your version appear in the title of the installation program.

Click “Next” to continue to the next page.



Click the “I Accept the agreement” radio button if you have read and accept the license agreement. If you do not accept the license agreement, press the “Cancel” button and you will be offered the option to stop the installation.

If you accept the license agreement, the “Destination Location” screen will appear:



The default destination folder is C:\xHB. An alternative destination drive and or folder can be chosen by using the “Browse” button.

All files will be copied to the destination folder, except the ADS (Advantage Database Server) and ApolloRDD DLL files, which will be copied to the Windows System32 folder.

The ADS files are:

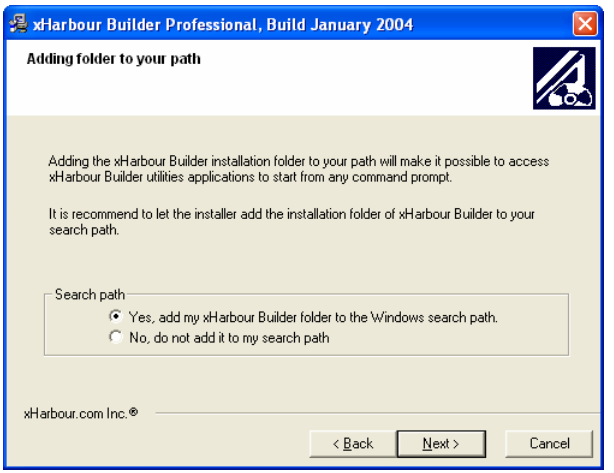
Ace32.dll
ADSloc32.dll
ADSlocal.cfg
Axcws32.dll

The ApolloRDD files are:

FTS32.DLL
SDE61.DLL
SDECDX61.DLL
SDENSX61.DLL
SDENTX61.DLL

Descriptions of these files can be found in the RDD Chapter of the" Programming Guide for xHarbour Builder".

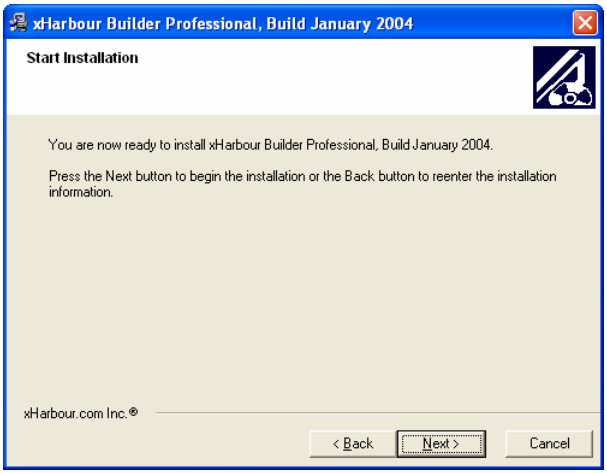
After selecting the destination folder, the installer will ask you if it can add the installation folder to your Windows search path.



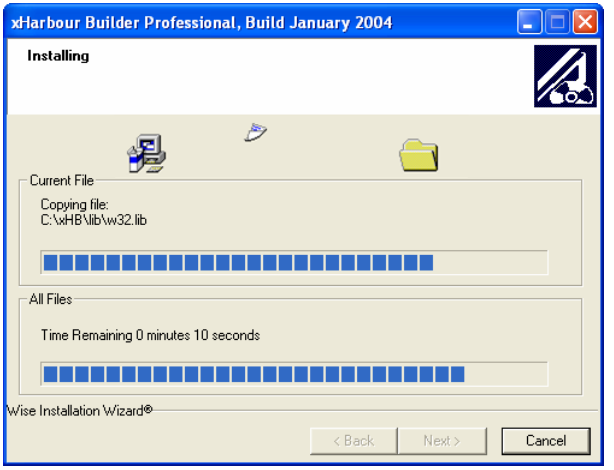
It is recommended that you let the installer add the chosen destination folder to your Windows search path. If you do so, you will be able to access xHarbour Builder's

utilities applications from any command prompt. If you have a .PRG file somewhere to be compiled and linked? Just enter “xBuild.exe YourPrg.PRG” at any command prompt.

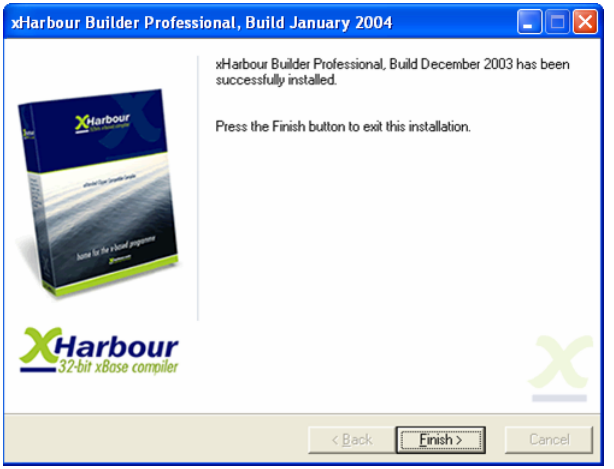
Ok, now the installer has almost everything it needs for the xHarbour Builder be installed on your system. Press “Next” on the “Start installation” dialog to start copying of the files to your computer.



Pressing the “OK” button on the password dialog will start the file copying which can be monitored on the following screen:



After all files are copied, the following screen will indicate successful installation of the xHarbour Builder.



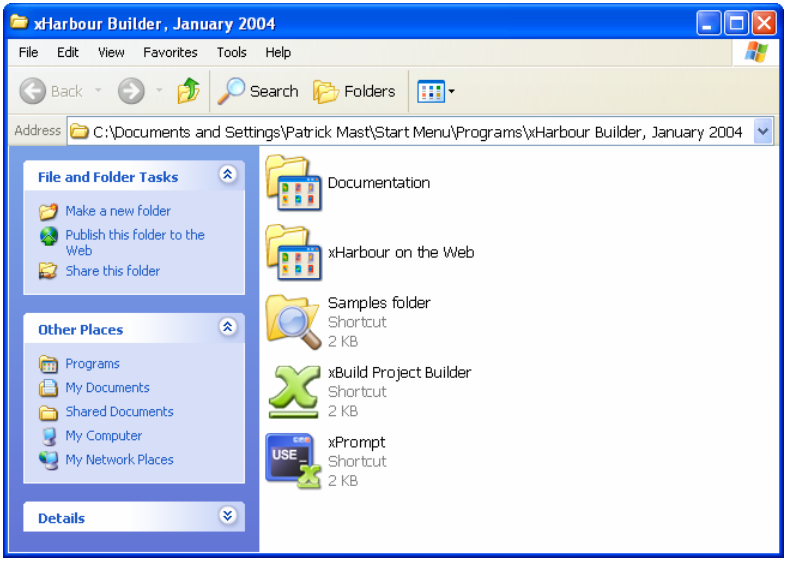
Pressing the “Finish” button will complete the xHarbour Builder installation. The Installation program ends here.

The xHarbour Builder installation program creates a folder in your Windows Programs menu called “xHarbour Builder, January 2004”. The “... January 2004” part in the folder name is the version of your xHarbour Builder installation. If you upgrade your xHarbour Builder later to a newer version, a new folder will be created in your Windows Programs menu and its name will match the version.

Note:
If you upgrade to a newer version of xHarbour Builder, it is recommended that you uninstall your current xHarbour Builder before installing the new xHarbour Builder. See section 2.8 on how to uninstall xHarbour Builder.

2.6. The xHarbour Builder menu folder

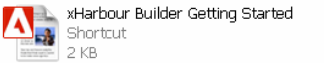
The xHarbour Builder installation program creates and then opens the xHarbour Builder menu folder, which looks like this on a Windows XP operating system:



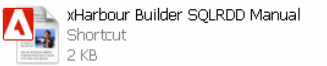
Let's examine all of the available icons in the xHarbour Builder Menu folder.

- **Documentation Folder**

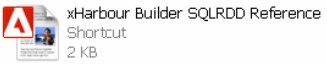
  xHarbour Builder's "Documentation" folder contains the following documents in Adobe's PDF format. You can read and print them with any PDF compatible reader. If you do not have a PDF reader on your system yet, please go to http://www.adobe.com/reader and install Abode's free Acrobat Reader.



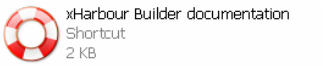
“xHarbour Builder Getting Started” is the manual you are reading now.



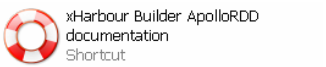
“xHarbour Builder SQLRDD Manual” is the documentation for the SQLRDD removable database driver which provides connectivity with all major SQL database systems such as Microsoft’s SQL server, mySQL, PostgreSQL, etc...



A reference to all of the SQLRDD functions can be found in the “xHarbour Builder SQLRDD Reference”. More SQLRDD details can be found in the “Programming Guide for xHarbour Builder”.



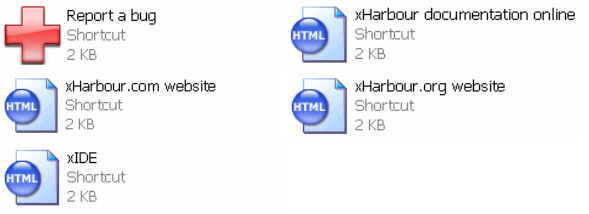
Also in the documentation folder is the Windows help file “xHarbour Builder documentation” which covers the xHarbour functions and commands. The “Programming Guide for xHarbour Builder” manual will replace this soon.



Vista Software’s documentation on the SDE61.DLL API calls can be found in “xHarbour Builder ApolloRDD documentation”, which is a Windows help file and can be opened on any Windows computer. The Vista Software’s SDE61.DLL functions described in this help file apply only to the ApolloRDD for xHarbour Builder.

- **xHarbour on the web**

The “xHarbour on the web” folder contains these shortcuts:



The “Report a bug” icon redirects you to the xHarbour Bugreporter on xHarbour.com. If you encounter a bug in xHarbour Builder, please report it via this facility as it helps xHarbour.com Inc. to correct bugs before the next release of the software.

The “xHarbour online documentation” redirects you to the online documentation web page at xHarbour.info. This web site holds all available documentation gathered from xHarbour.org.

The xHarbour.com and xHarbour.org websites can be reached by opening either of the “xHarbour.com website” or the “xHarbour.org website” icons.

The xIDE web pages at xHarbour.com can be reached via the “xIDE” icon.

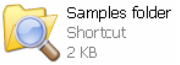
- **xPrompt**

xHarbour.com’s xPrompt (also known as xBaseScript) is an xHarbour Builder application with three personalities:

- 100% Clipper compatible Pre-Processor.
- Dot Prompt console which allows interactive use of most of the xHarbour / Clipper syntax.
- A Clipper / xHarbour / xBase interpreter.

More in-depth information on xPrompt can be found later in this manual.

- **Samples folder**



The “Samples folder” holds xHarbour sample applications and tests to show some of xHarbour’s features. Feel free to modify the source code of the samples to learn and experiment with xHarbour Builder.

The samples folder also holds a “FiveWin” folder. In this folder you will find examples of how to get started with xHarbour Builder and FiveWin. More details on “FiveWin” can be found later in this manual.

Examples of using the SQLRDD can be found in the “sqlrdd” sample folder.

Note:
If you build the SQLRDD samples with the Personal or Professional version of xHarbour Builder, a message will popup saying, "This application was built with a demo of SQLRDD, distributed by xHarbour.com". The full version of SQLRDD is available only in the Enterprise version of xHarbour Builder.

Note:
If you build any samples with the xHarbour Builder Personal version and the ApolloRDD, you will receive a similar popup message. The full version of ApolloRDD is available only in the xHarbour Builder Professional and Enterprise versions.

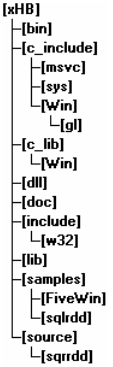
All samples have an xBuild Project file. Just open the xBuild Project file to launch xHarbour Builder’s project builder and start building the samples.

- **xBuild Project Builder**

The “xBuild Project Builder” icon opens the xBuild Wizard with an empty project. See Chapter 4 for more about xBuild.

2.7. xHarbour Builder folder overview

During a default xHarbour Builder installation, the “xHB” folder is created on the “C:” drive. If you chose a different “Destination folder” or drive during the installation, your xHarbour Builder Tree will be adjusted accordingly. The folders of a default xHarbour installation should look like this:



Let’s look at each folder separately.

2.7.1. The xHarbour Builder root folder

The xHarbour Builder root folder, defaulting to “xHB”, holds the xHarbour Builder End-User License Agreement in a Microsoft Word document. Please read the document as it contains important information for you.

2.7.2. The “bin” folder

xHarbour Builder’s “bin” folder holds the following xHarbour Builder compiler, service programs and utilities :

- xcc.exe and xcc.dll
  xHarbour Builder’s C compiler
- xlink.exe
  xHarbour Builder’s linker
- dll2lib.bat
  Creates an Import LIB for a given .DLL
- xPrompt.exe
  xBaseScript - Dot Prompt, Interpreter and Pre-Processor

- xbuild.exe
  Console mode xBuild Project Builder
- xbuildw.exe
  Windows GUI xBuild Project Builder
- xhb.exe
  xHarbour Compiler
- xlib.exe
  xHarbour Librarian
- xrc.exe and xrc.dll
  xHarbour Resource compiler

2.7.3. The “c_include” folder

xHarbour Builder’s “c_include” folder holds the xHarbour Builder include files. This folder will only be used by the xHarbour Builder’s C compiler. Normally, you will not need to use any files from this folder directly.

2.7.4. The “c_lib” folder

The “c_lib” folder too is used by the xHarbour Builder’s C compiler depending upon the functions used in your application. You may need to link libraries from this folder.

For example, if you use any MCI (*media control interface*) function calls from within your application you will need to link in “winmm.lib” which can be found in the “c_lib\win” folder. In chapter 4 we have an example of how to use the MCI functions and how to link in the winmm.lib file.

2.7.5. The “dll” folder

The “dll” folder holds .DLL files as they become needed by xHarbour Builder or by your built application.

xHarbour Core .DLL’s

- xhbdll.dll
  .DLL version of the xHarbour Run-Time support – it will be used automatically by xBuild if you select the “Use DLL” option
- xhbddll.dll
  Used in place of “xhbdll.dll” during debugging
- xhbmtdll.dll
  Used in place of “xhbdll.dll” for Multi Thread xHarbour Builder applications.

- xhbdmtdll.dll
  Used in place of “xhbmtdll.dll” when debugging Multi Thread xHarbour Builder applications.
- xhbcommdll.dll
  .DLL version of the xHarbour Serial Port Communications library
- xhbzipdll.dll
  .DLL version of the xHarbour ZIP library

ADS .DLL’s (Advantage Database Server)

- ace32.dll
  Advantage Client Engine DLL. This DLL contains the core Advantage Windows client functionality.
- adsloc32.dll
  Advantage Local Server DLL. This DLL contains the core local server functionality. This file is needed only when you use the ADS local server.
- accws32.dll
  Advantage remote communication library. This DLL is used when accessing data via the Advantage Database Server. This file is not needed if you only use the ADS local server.
- adslocal.cfg
  Advantage Local Server DLL configuration file. This file must be in the search path of Windows. (e.g. your \windows root directory)

Note:
All of the ADS DLL files have been copyrighted by Extended Systems and are provided with their permission. (http://www.advantagedatabase.com)

ApolloRDD .DLL’s

- sde61.dll
  ApolloRDD’s core DLL.
- sdentx61.dll
  Required if you use Clipper tables and indexes in your application.
- sdecdx61.dll
  Required if you use FoxPro tables and indexes in your application.
- sdensx61.dll
  Required if you use HiPer-SIx tables and indexes in your application.
- fts32.dll
  Required if you use Fast Text Search in your application.

Note:
*All of the ApolloRDD .DLL files have been copyrighted by Vista Software and are provided with their permission. (*http://www.vistasoftware.com*)*

2.7.6. The “doc” folder

xHarbour Builder’s “doc” folder holds all documentation on how to use xHarbour Builder.

xHarbour.chm is a Windows Help file containing all of the xHarbour Builder functions and commands.

Also in the “doc” folder is the “ApolloAPI6.chm”. This is the API documentation for ApolloRDD.

The .TXT files provide more in-depth information on particular topics.

2.7.7. The “include” folder

xHarbour Builder’s Runtime include files are held in the “include” folder.

An xHarbour include file is a separate file that holds xHarbour directives. It is used by your xHarbour Builder .PRG source files. The use of include files is covered later in this manual.

The files in the include folder with “API” or “H” extensions are used by the xHarbour Builder C source files.

2.7.8. The “lib” folder

All LIBRARY files for the xHarbour Builder are located in this “lib” folder. These library files use the file extension “.LIB”. Depending on which functions you will be using in your source code, the appropriate LIB files will need to be linked together with your compiled source code.

The LIB files whose names begin with “xhbd” support debugging activities. If you use xHarbour Builder’s xBuild Project Builder to build your application, linking the debug LIB’s is done automatically for you when you select the “*Include Debug Information*” option.

See later in this manual for more information on LIB files.

2.7.9. The “samples” folder

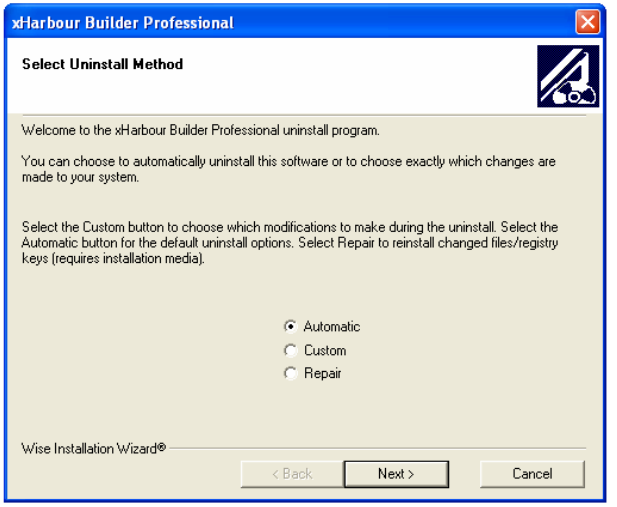
xHarbour Builder’s default installation provides you with some samples to compile and run. This is also a good starting point when learning how to use xHarbour Builder.

Samples on how to use FiveWin in xHarbour Builder can be found in the “FiveWin” samples folder. You need at least FWH (FiveWin for xHarbour) version 2.4 to be able to use FiveWin with xHarbour Builder. The “fwTutor1.prg” sample shows what FWH version you currently have installed. xHarbour’s “Programming Guide for xHarbour Builder” will have a complete chapter on how to use Fivewin with xHarbour Builder.

SQLRDD samples can be found in the “SQLRDD” samples folder. Please read the SQLRDD manual before trying the SQLRDD samples. The SQLRDD documentation can be found in the “doc” folder in your xHarbour Builder installation folder.

2.8. Uninstall xHarbour Builder

If for any reason, you need to uninstall xHarbour Builder, you can do so by going to your Windows “Start” button and select “Control panel”. Open the “Add or Remove Programs” window. Select the xHarbour Builder installation to uninstall and press the “Change/Remove” button. The following dialog will appear:



Use of the “Automatic” radio button is recommended. Pressing the “Next” button will show a final confirmation dialog. Pressing the “Finish” button will start uninstalling xHarbour Builder.
On completion, all files installed by the installation program will be deleted.

Please note that files created by the xHarbour Builder compilers, and source code put in the xHarbour Builder folder by you will NOT be deleted. If you want to delete these files, you will need to delete them manually.

3. Build your first application

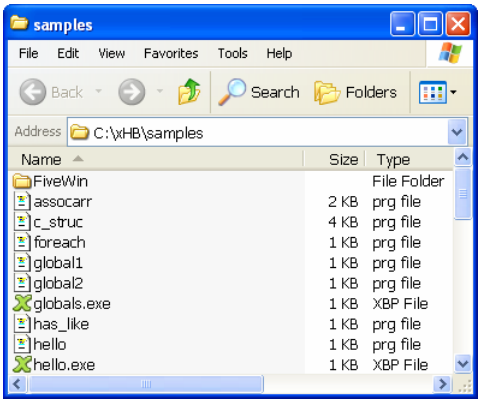
This chapter gives you complete instructions on how to build your first sample application with xHarbour Builder. It will guide you through the process with xHarbour Builder's Project Builder "xBuild".

Before giving an in-depth explanation on how to use xBuild, let's first build some samples from xHarbour Builder's installation samples folder.

Let's first open the samples folder of your xHarbour Builder installation. Go to the Windows "Start" button, and open your "xHarbour Builder, November 2003" folder. Now open the "Samples folder".

Note:
The "January 2004" in the name of xHarbour Builder's start menu folder refers to the release version of your installed xHarbour Builder and may differ from this example.

If you set the viewing mode of this folder to "Details", it will look like this:



Notice that you will see 2 types of files in this folder list; .PRG and .XBP files. What's a .PRG and what's an .XBP file?

3.1. A .PRG file

A .PRG file is an ASCII text file that holds the source code of your application. The complete source of your xHarbour Builder application can comprise several .PRG source files. xHarbour Builder will compile and link these .PRG source files into one executable.

Note:
xHarbour Builder can also link your source files into a .DLL or LIB file. More on this in the "Programming Guide for xHarbour Builder".

First let's see what "hello.prg" looks like. As a .PRG file is just a text file, you can open it with any text editor. In Windows this can be Notepad, or any of your preferred text editors.

The contents of hello.prg looks like this:

```
PROCEDURE Main()

   ? "Hello world"
   wait

RETURN
```

When compiled and linked by xHarbour Builder, this little program will print the words "Hello world" together with "Press any key to continue…" to your console.

More on xHarbour Builder commands and functions in the "Programming Guide for xHarbour Builder".

3.2. A .XBP file

.XBP files are xBuild project files. An xBuild project file holds all settings and information to be able to compile your target file. During the xHarbour Builder's installation, files with the .XBP extension were associated with the program \xhb\bin\xBuildw.exe. In this way you just need to double click your .XBP file to start the xBuild Project Builder with the settings and information to build your target application. In Windows Explorer, an .XBP file looks like this:

An xBuild project file needs to be in this form:

<Application_Name>**.**<Target_Type>**.**XBP

Where "Application_Name" is the name of your application.
"Target_Type" can be:

- "EXE" for Win32 Windows applications;
- "LIB" for xHarbour Builder library files; or

- “DLL” for Win32 Windows DLL

As in our samples, and in most cases, the “Target_Type” will be set to “EXE”.

For more information about how and when to build “LIB” or “DLL” files, please refer to the “Programming Guide for xHarbour Builder”.

3.3. Building hello.exe

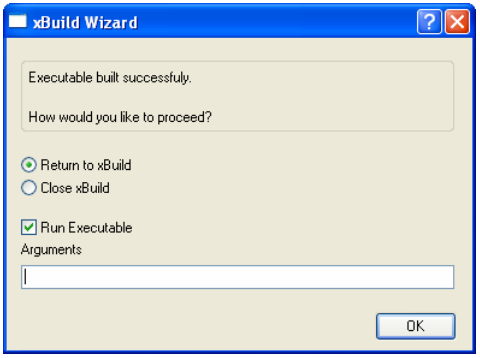
Building “hello.exe” is as simple as opening “hello.exe.xbp” and pressing the “Build now!” button in xBuild.

So, open “hello.exe.xbp” by double clicking on it in Explorer, in the samples folder, OR by clicking on it with your right mouse button and selecting the “Open” menu.

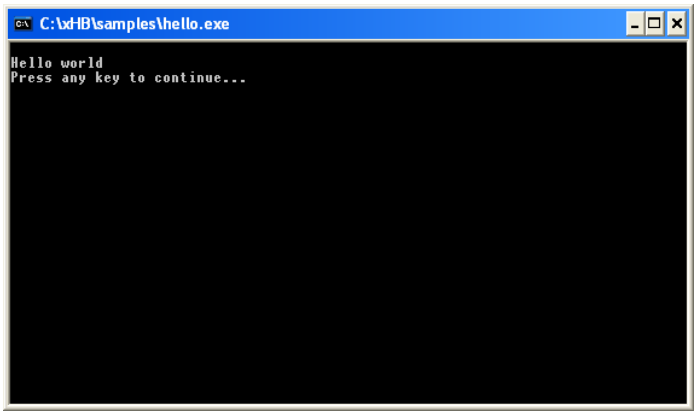
xBuild will open the “hello.exe.xbp” file and load the settings into xBuild, ready for you to use. The xBuild dialog will then look like this:

Press the “Build now!” button to compile and link hello.prg into the target application, “hello.exe”.

If xBuild did not detect any compile or link errors, the following dialog will appear.



xBuild is suggesting to run the new executable application and return to xBuild on its completion. So, press the “OK” button to start hello.exe!



Congratulations! Your first xHarbour Builder Application is ready! It is THAT simple ;-)

The next chapter explains xBuild in more detail.

3.4. Deploy your application

Applications built with xHarbour Builder may be installed on other computer systems under the xHarbour Builder license agreement. The xHarbour Builder license agreement can be found in the root directory of your xHarbour Builder installation folder. Please read this document carefully as it has important information.

Deploying your application to your users is as simple as distributing your .EXE file together with all .DLL, .DBF or any other files needed to be able to run your application.

Depending on how you build your application and/or what RDD you have used, xHarbour Builder will need .DLL files to be available on your user's computer. Page 2-23 has a list of .DLLs available to be deployed together with your .EXE. These DLLs should be installed in the Windows search path of your user's computer. A common place to install .DLL's is the "Windows" or "Windows\System32" folder. In order to avoid "DLL Hell", Microsoft now prefers all associated DLL files to be placed in their application's folder.

Note:
If your application uses one of the native RDDs like DBFNTX or FBCDX, and it doesn't explicitly utilizes some specific DLL, then your application will be ***self contained*** *in a single executable, and you will not need to deploy any DLL[s] when deploying that application.*

4. xBuild Project Builder

In this chapter, we will have a more in-depth look at how to use xBuild Project Builder.

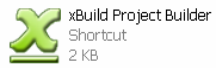
So, what exactly is xBuild? Well, xBuild is a portable comprehensive Project Builder that will make compiling and linking your application as simple and smooth as possible. It will help you with the automation process of compiling and linking. For those with a CA-Clipper programming background, xBuild Project Builder performs the functions of RMake.exe, CA-Clipper's "make" utility. xBuild Project Builder is a .PRG and .C compiler, linker and .RC resource compiler all neatly wrapped together in one package.

To build your application, xBuild Project Builder will search for your project source files with dates or times more recent than your target application. Any source code that is newer than your target application will be rebuilt using the program associated with the extension of that newer source file.

The xBuild Project Builder control panel comes in two versions - one for use in Windows (xBuildW.exe) and another for use in console mode (i.e. in DOS: xBuild.exe). They work in the same way, so only the Windows version will be explained here. xBuild Project Builder's shortcut in the xHarbour Builder menu folder is a shortcut to the Windows version.

xHarbour Builder's installation program added xHarbour Builder's "bin" directory to your Windows search path. So, if you want to use xHarbour Builder's console mode xBuild.exe, just type "xBuild.exe" at any command prompt.

To open xBuild Project Builder in Windows GUI mode, go to the xHarbour Builder Installation folder and click on the "xBuild Project Builder" icon:



You can also open xBuild Project Builder by double clicking on any .XBP file. xHarbour Builder's installation program associated .XBP files with the Windows GUI xBuild Project Builder's application. All settings and parameters saved in an xBuild Project Builder file will be automatically loaded by opening an .XBP file. See also section 3.2 for more on .XBP files.

You will find some samples of .XBP files in the 'samples' folder of your xHarbour Builder's installation directory.

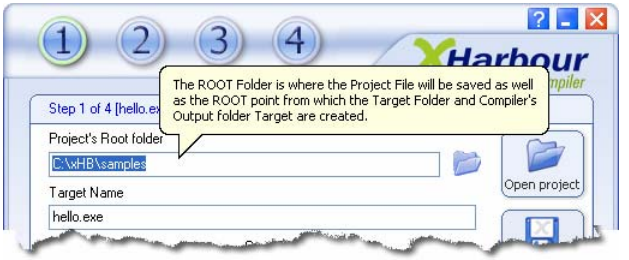
4.1. xBuild Project Builder: the basics

After you have written your code and designed your screen resources if you are building a Windows GUI application, xBuild Project Builder organizes all the pieces of your application into a "project" listing of elements, which it saves in an .XBP file (xBuild Project file). If you come from a CA-Clipper background, the .XBP file is similar in function to the .RMK "make" file combined with a .LNK "link" file.

Let's first have a look at the main window of the Windows GUI based xBuild Project Builder. Start the xBuild Project Builder by going to its shortcut in the xHarbour Builder's installation folder. The xBuild Project Builder will start and look like this:

The xBuild Project Builder uses a four-step approach to build your application. It will build your target application with the minimum of information from you. Moving from step 1 to step 2 is as simple as clicking on the blue "2" button on xBuild Project Builder's top button bar. If you want to go straight to step 4, just click on button "4".

xBuild Project Builder uses tool tips to give you interactive help on most of the fields. Let your mouse cursor rest for two seconds on a field or checkbox and a tool tip will pop up.



On the right side of the main xBuild Project Builder window, there are three buttons that will be accessible and visible every step of the way. This is handy if you only need to change settings in screen 2 and start building right away.

4.1.1. Open project

Clicking “Open project” presents a selection of project files (of type .XBP) saved previously. All active settings in your open xBuild Project Builder will be overwritten by the settings of the newly selected project file.

4.1.2. Save project

Saving your project is done simply by clicking the “Save project” button. It will save all the settings and parameters of your current project. Also, if the “Auto Save Project” checkbox is checked, your project will be saved every time a successful build is made.

Note:
If the build process encounters an error, “Auto Save Project” will NOT save your project. Auto saving your project is ONLY done after a successful build.

4.1.3. Build now

The “Build now!” button will start the build process of the currently open project. If no error occurs during this process, a dialog appears asking if you want to start your application. More information on the build process in section 4.6.

4.2. Step 1

Step 1 of the xBuild Project Builder sequence collects the more general information about your project, like the Project's root folder, your target application name, etc…

Let's go over them one by one.

4.2.1. Project's Root folder

The "Project's Root folder" is the folder where your target application resides on your hard drive and where your .XBP project file will be saved. It is also the top folder under which the target folder and "Compiler's Output Folder" target are created. If you want to change this folder, click on the folder icon on the right of this field. A dialog will appear where you can select the desired folder to use.

4.2.2. Target Name

The "Target Name" field holds, yes you guessed it, the target name. You may also prefix target name with an optional target folder.

Note:
*If specified the target folder prefix will be a **SUB** folder located in the project's Root Folder. If the target folder does not exist, xBuild Project Builder will try to create it for you.*

4.2.3. Target Type

xBuild Project Builder is able to build three types of applications. The “Target type” dropdown list lets you choose between building an “EXE” type of application, a “Lib” or a “Dll”.

4.2.4. Compiler’s Output Folder

Setting this field will tell xBuild Project Builder to redirect all compilers and linker’s output to this subfolder. If the “Compiler’s Output Folder” does not exist, xBuild Project Builder will try to create it for you. If you leave this field empty, all output will go to the project’s root folder. Tipical setting for this field might be “OBJ” or “OBJ32”.

Your target application will always be put in the project’s root folder.

Note:
*The “Compiler’s Output Folder” is a **SUB** folder located in the project’s root folder.*

4.2.5. Force Clean build

Checking the “Force Clean build” checkbox forces xBuild Project Builder to consider all project files to be “out of date”. This means that clicking the “Build now!” button when the “Force Clean Build” checkbox is checked will cause ALL files belonging to the project to be rebuilt using their associated application. Source code saved in .lib files, such as third party libraries, will not be recompiled under these circumstances.

This can be of use if you want or need a fresh, clean build of your application. Open xBuild Project Builder, check the “Force Clean Build” checkbox and click the “Build now!” button.

4.2.6. Include Debug information

If you want to debug your application and need debug information on C-level, check the “Include Debug information” checkbox. xBuild Project Builder will then link the required debug info into your application. All xHarbour Builder libs include C debug info which is used when “Include Debug information” is checked. Source code saved in .lib files, such as third party libraries, will not be recompiled under these circumstances.

Note:
Only check “Include Debug information” if you really want to debug your application at C-level. For example, detecting faulty code when a GPF occurs in your application. Compiling and linking your application with debug set ON, will result in a much bigger

target application. Your application will also NOT run at full speed when it carries all of the debug information with it. So, use this option with caution.

Note:
If you want to debug your PRG code, just add the "–b" switch in the "Flags" field in screen 4 of xBuild Project Builder. Be aware though that xBuild Project Builder adds the "-n -w –q" switches automatically If the "Flags" field is NOT used. So, if you manually set the" –b" switch, don't forget to ***add*** *the "-n -w -q" switches if needed.*

4.2.7. Multi thread support

Note:
The "Multi thread support" checkbox is disabled in the xHarbour Builder Demo and Personal versions. Multi thread support is only available in the xHarbour Builder Professional, Professional MP and Enterprise versions.

xHarbour Builder's applications can be Multi threaded. You need to check the "Multi thread support" to let xBuild Project Builder know that it needs to link your application with multi thread compatible libraries. More on multi-threaded applications can be found in the "Programming Guide for xHarbour Builder".

4.2.8. GUI Application

To build a Windows GUI application, you need to use one of the many GUI libraries available for xHarbour Builder. xBuild Project Builder will automatically detect the two most popular ones, FiveTechSoft's third party library FWH (FiveWin for xHarbour) and What32+WHOO, xHarbour Builder's native Windows GUI library. So, when you use one of these GUI libraries, you do NOT need to check the "GUI Application". You only need to check it when you are building a Windows GUI application NOT using FWH or What32+WHOO.

More on building GUI applications with FWH or What32+WHOO can be found in the "Programming Guide for xHarbour Builder".

4.2.9. Auto Save Project

By checking "Auto Save Project" your project will be saved after a successful build. If the compiling or linking process generates any kind of error, your project file will NOT be saved automatically. If you encountered an error in the compile or link process but still want to save your project, you will need to click the "Save project" button on the right side of the xBuild Project Builder window.

4.2.10. Use DLL

Note:
The "Use DLL" checkbox is disabled in the Demo and Personal versions of xHarbour Builder. Support for DLL enabled applications is only available in the xHarbour Builder Professional, Professional MP and Enterprise versions.

Using this option, the xBuild Project Builder will produce VERY small executables. The "hello.exe" sample, built without this option, is an executable file of 497 KB. Building it with the "Use DLL" option, "hello.exe" shrinks the file to a size of only 9 KB. This is more than 50 times smaller!

To run these small executables the xHarbour Builder's Run-Time engine DLL needs to be accessible. This DLL needs to be in the same folder as your target application or in the search path of your Windows system, such as the "windows" or "windows\system32" folder. By default, the xHarbour Builder installation places these DLLs in the "windows\system32" folder. If you deliver your application to your customers, you will also need to include the required DLLs.

You can find the xHarbour Builder Run-Time engine in your "dll" folder of your xHarbour Builder installation folder. Note that there are four xHarbour Builder Run-Time DLL files in your dll folder:

- xhbdll.dll
  Regular xHarbour Builder Run-Time DLL version
- xhbddll.dll
  Debug version of the xHarbour Builder Run-Time DLL. Use this DLL if you checked the "Include Debug information" together with the "Use DLL" checkbox.
- xhbmtdll.dll
  Multi thread version of the xHarbour Builder Run-Time DLL. Use this DLL if you checked the "Multi thread support" together with the "Use DLL" checkbox.
- xhbdmtdll.dll
  Multi threaded debug version of the xHarbour Builder Run-Time DLL. Use this DLL if you checked the "Multi thread support" together with the "Use DLL" and "Include Debug Information" checkbox.
- xhbcommdll.dll
  .DLL version of the xHarbour Serial Port Communications library
- xhbzipdll.dll
  .DLL version of the xHarbour ZIP library

xHarbour Builder also provides you with the ability to create a very small pcode DLL file from any .PRG source code file. These DLL files can be called from your main application just like you do for standard or user defined functions. You may link your application with many of these DLL files and keep your application as small as possible. Another advantage of using this approach is that you only need to send very small files to your client when sending upgrades of your application to your client. More on this type of DLL file is available in the “Programming Guide for xHarbour Builder”.

4.3. Step 2

In most cases, step 2 needs little intervention from the user’s side. None of the examples in the xHarbour Builder samples folder has any settings changed at step 2. However, Murphy’s Law says YOU will of course need to alter these settings. So, let’s see what the input fields of step 2 can do for you.

4.3.1. Main source file

The “Main source file” field holds the name of the main .PRG or .C file that contains your startup function or procedure. If your application consists of more than one .PRG or .C file, only the .PRG or .C file with the startup function or procedure needs to be filled in here. You can use the folder icon on the right of the input field to select the main source file if needed.

4.3.2. LIB folder[s]

Your application may use your own or third party libraries in addition to those of the xHarbour Builder software. This field records the location of these **extra** library files. xHarbour Builder's standard library folders are automatically added by xBuild Project Builder. FWH's library folder is also NOT needed here as it is set on the "FiveWin" tab in step 4.

If you need to add more than one library folder, you can separate them by a semicolon (;). Use the folder icon on the right side of the field to select one or more library folders as needed.

4.3.3. Include folder[s]

Your application may use your own or third party include files in addition to those of the xHarbour Builder software. This field records the location of these **extra** include files. xHarbour Builder's standard include folders are automatically added by xBuild Project Builder. FWH's include folder is also NOT needed here as it is set on the "FiveWin" tab in step 4.

If you need multiple include folders, you can separate them with a semicolon (**;**). You can use the folder icon on the right side of the field to select one or more include folders as needed.

4.3.4. Define[s]

If you need to set any user-specified defines at compile time, use this field to add them. You can add multiple defines by separating them with a semicolon (**;**). Adding defines here is the same as using the –D compiler option.

4.4. Step 3

The step 3 screen contains the list of all files needed to build your target application, except for the source file defined in the "Main source file" field of step 2.

Use the “Add” button or simply “drag and drop” to add more files to your project. When using drag and drop, xBuild Project Builder will automatically add the file to the relevant tab, depending on the extension of the dropped file.

If you want to remove a file from the list, select it first by highlighting it and then use the “Remove” button or just press the “Delete” key on your keyboard.

The “Properties” button is reserved for future functionality. It has no function for now.

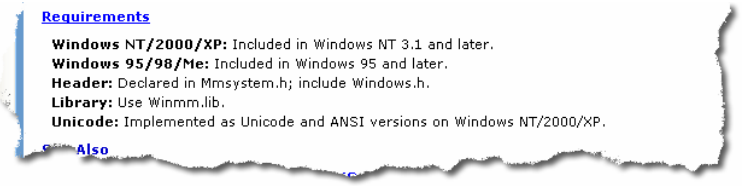
Note:
FiveWin users should pay special attention to the .lib tab. A much-asked question on xHarbour’s technical news group is how to solve an “unresolved external” error from xBuild Project Builder, which results in an xBuild error window message like:

> *xLINK: error: Unresolved external symbol '_HB_FUN_MCISENDSTR'.*
> *xLINK: fatal error: 1 unresolved external(s).*

Such error means that you used the mciSendStr() function somewhere in your source code, but xLink, xHarbour Builder’s linker, could not find it in any of its standard xHarbour Builder libraries or in any library listed in the “.lib” tab in xBuild Project Builder’s step 3.

So, you will need to add the library that holds the mciSendStr() function. But how do you know what library holds this function? One simple solution is to search for the function name on Microsoft’s MSDN web site. Go to *http://msdn.microsoft.com/library* and search for mciSendString (this is the original Microsoft function name). The search engine will give you a list of all pages where it found mciSendString. For this sample,

click on the “Windows Multimedia SDK Changing the Playback State” link. On the bottom of the page, you will see this:



The most important information for us here is “Library” as it will tell us which library holds this function. Now, most Windows specific libraries are in the “c_lib” folder of your xHarbour Builder installation folder. So, click the “Add” button on the step 3 screen and select the Winmm.lib library. Click the “Build now!” button again to see if all unresolved external errors are solved. If new unresolved functions appear, repeat the search process.

A direct link to the mciSendString on Microsoft’s MSDN site is:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/multimed/htm/_win32_mcisendstring.asp

4.5. Step 4

The Step 4 screen shows you where your xHarbour Builder compiler and your C compiler are located. If you use FiveWin for xHarbour, the “FiveWin” tab shows you

where it is located. Most settings on screen four are filled in automatically for you by xBuild Project Builder.

All settings of screen 4 are saved in the “xbuild.ini” file, which is located in the “Root folder” of your project, defined at step 1. When xBuild Project Builder is started again, it will load these settings from the available xbuild.ini file.

4.5.1. xHarbour tab, “Root folder”

The “Root folder” is where your xHarbour Builder compiler is located. It is the same folder where you installed xHarbour Builder. If you need to change it for any reason, click on the folder icon on the right of the “Root folder” field and choose your xHarbour Builder compiler folder. Be aware that if you select a root folder without the “xhb.exe” file present, xBuild Project Builder will not function properly.

4.5.2. xHarbour tab, “Lib folder”

This is where the xHarbour Builder libraries are located. It needs to be set relative to the “Root folder”. So, “lib” actually means “My libraries are in C:\xHB\lib\”.

4.5.3. xHarbour tab, “Flags”

If you want to set some flags for the xHarbour Builder PRG compiler, this is the place. xBuild Project Builder will by default set the –n and –w flags for you. If you want to add extra flags, you will also NEED to add the –n and –w flags by hand. So, if you want to add the -p flag to your xHarbour Builder PRG compiler, specify “-n –w –p” in this field. The effect of all xHarbour Builder PRG compiler flags is explained in “Programming Guide for xHarbour Builder”.

4.5.4. C Compiler tab, “C Compiler”

The “C Compiler” tab is where you tell xBuild Project Builder which C compiler to use, where it is located and what flags to use for this C Compiler. If you use xHarbour Builder, it is preferable to set the “C Compiler” field to xCC as this is the internal C Compiler used by xHarbour Builder. If you want to test your code with another C compiler, use the dropdown “C Compiler” list to choose your preferred C Compiler.

xBuild Project Builder can be used with these C Compilers:

- xCC (xHarbour Builder’s internal C Compiler)
- BCC (Borland C++ Compiler)
- MingW (Minimalistic GNU for Windows)
- MSVC (Microsoft Visual C++ Compiler)

Note:
You can NOT use any C Compiler other than xCC if you use xHarbour.com’s xHarbour Builder. If you want to use another C Compiler you also need to have the xHarbour binaries and libraries compiled for your chosen C Compiler.

4.5.5. C Compiler tab, “Root folder”

The “Root folder” is where your preferred C Compiler is located. If the C Compiler is set to “xCC”, the internal C Compiler used by xHarbour Builder, the “Root folder” is the same folder where you installed xHarbour Builder. If you have chosen a different C Compiler, you need to change the “Root folder” to the root folder of your chosen C

Compiler. Click on the folder icon on the right of the “Root folder” field and choose your C Compiler folder.

Note:
Don’t choose the “bin” folder of your C Compiler. xBuild Project Builder will automatically search your C Compiler in the “bin” dir of your given root folder.

4.5.6. C Compiler tab, “Flags”

Normally you do not need to set any C flags if you use “xCC”, the internal C Compiler used by xHarbour Builder. See “Programming Guide for xHarbour Builder” for more details on C flags.

4.5.7. FiveWin tab, “Root folder”

This tab is only needed if you use “FiveWin for xHarbour” (“FWH” for short). Via this tab you set the root folder for your FWH installation. xHarbour Builder will try to find your FWH installation on your local driver. If it cannot find FWH, you will need to set it manually. Click on the folder icon on the right of the “Root folder” field and choose your FWH folder.

4.5.8. FiveWin tab, “Lib folder”

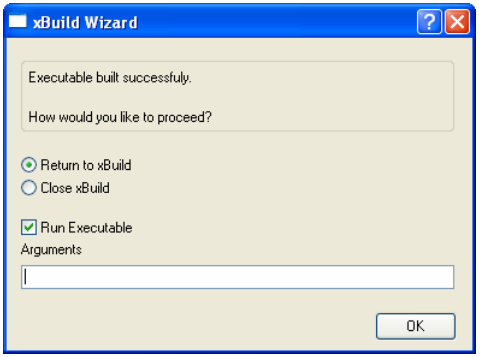
This is where the FWH libraries are located. It needs to be set relative to the “Root folder” of your FWH installation. So, “lib” in the above picture actually means “My FWH libraries are in C:\fwh\lib\”.

Note:
xBuild by default will also search the LibX subfolder of your FWH installation.

4.6. Let's Build!

Ok, all settings are set, so let's build our hello.prg sample and see in more detail what happens next.

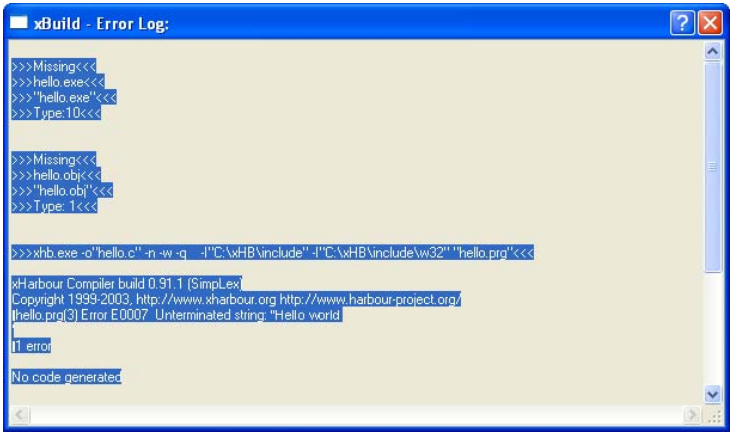
Open "hello.exe.xbp" in xBuild Project Builder as described in section 3.3. Clicking the "Build now!" button will start the build process. xBuild Project Builder will start compiling "hello.prg" into the "hello.obj" file. This "hello.obj" is then linked with the xHarbour Builder libraries to produce the "hello.exe" file. If no error occurs during this process, the following screen appears:



This dialog allows you to run the successfully built hello.exe and then return to xBuild Project Builder. If you need to pass parameters to your application, use the "Arguments" field to tell xBuild Project Builder to start the application with your given parameters.

Selecting the "Return to xBuild" radio button will give focus back to xBuild Project Builder after exiting the newly built application. If you don't need xBuild Project Builder after the successful build, click the "Close xBuild" radio button to close xBuild Project Builder just after starting your application. This is also the moment when xBuild Project Builder saves your project file if you checked the "Auto Save Project" checkbox at step 1.

Should xBuild Project Builder encounter any kind of error while building your target application; it will present its error log to you in an xBuild – Error Log dialog like this:



Note:
Even If you checked the "Auto Save Project" checkbox at step 1, your project file will NOT be saved if an error occurred in the build process.

4.6.1. .LOG file

The error log information shown in the picture above is also saved in a log file in the folder where your application is being built. Look for "<your_application_name>.log". As this is a normal ASCII text file, you can open it in any text editor of your choice. You don't need this file on your distributable files list for your final application.

4.6.2. .C file

The xHarbour Builder creates a file named "<your_application_name>.C". This C file is then compiled and linked by the C Compiler. This compile and linking process is done out of sight by the xBuild Project Builder. The resultant C file is not needed to run your application so you do not need to distribute this file with your end-user application.

4.6.3. .OBJ file

The "<your_application_name>.OBJ" file is the output file of the xHarbour Builder C compiler. This file is linked together with the xHarbour Builder libraries into the final application. This file is not needed by your final application and should therefore not be distributed to your customers.

4.6.4. .MAP file

After a successful project compile and link cycle, there will be a file called “<application_name>.MAP” in your application folder. This file contains a list of segments in the order of their appearance within the corresponding application. This file is only for debugging purposes and does not need to be distributed to your customer.

5. Migrating your FiveWin app

In this chapter, we discuss many of the issues you may encounter in migrating your 16-bit CA-Clipper + FiveWin application to xHarbour Builder + FWH. As hard as the xHarbour developers have tried to make the migration of a CA-Clipper application to xHarbour Builder to be as smooth and effortless as possible, there are still some points that need your attention when migrating your CA-Clipper application.

xHarbour started as a 100% CA-Clipper version 5.2 compatible compiler. Along the way many new functions and extensions where added to the xHarbour compiler. All these extensions mean that the xHarbour compiler is now a modern compiler with all the functionality found in popular modern compilers such as Delphi or Visual Basic.

As CA-Clipper is a 16-bit pcode compiler, and xHarbour a 32-bit pcode compiler, the biggest difference between them is the 16/32 bit architecture. FiveTech Software, the makers of FiveWin, have done a great job in providing a Windows GUI lib that uses the same code base for 16-bit CA-Clipper and 32-bit xHarbour Builder applications.

As always, it is advisable to use the latest versions of your programming tools. At the time of writing this, FWH was at version 2.4. Therefore all related samples will expect FWH version 2.4.

5.1. Environment variables

When you build your application with xBuild Project Builder, there is no need to set any environment variables. xBuild Project Builder will try to find all required files to build your application automatically.

5.2. Replacing .RMK and .LNK files

A CA-Clipper application uses .RMK files to be run by RMake.exe, CA-Clipper's make utility. When all source files are built by their respective builders as defined in the .RMK file, the linker reads the .LNK file and links all parts into one .EXE file.

These two processes are handled by xHarbour Builder's "xBuild Project Builder". So, there is now no need for the .RMK and .LNK files when building your application with xHarbour Builder.

All of the source files required to build your applications (such as .C, .PRG, .RC ...) need to be defined in your .XBP xBuild Project file. See chapter 4 for more information on how to use xBuild Project Builder.

5.3. RESIZE16

FWH has a new RESIZE16 clause to be used with the ACTIVATE DIALOG command. When using this clause, FWH automatically resizes any dialogs built using a .RC resource so that they have the same size as your 16-bit dialogs.

This change is necessary because, when running a 32-bit application, the system automatically applies a three-dimensional appearance to dialog boxes, which results in the dialog box having an unbolded font. Because of this, a dialog box produced by a 16-bit application can look very different when compared with the same dialog box produced by a 32-bit application. There were long discussions about this “feature” in the FiveWin news groups.

A sample of the discussions of that time is still online.

The dialog box produced by a 16-bit CA-Clipper + FiveWin application is available at:
http://www.fivewin.info/16bits-Clipper.gif

The dialog box produced by a 32-bit xHarbour + FWH application is available at:
http://www.fivewin.info/32bits-xHarbour.gif

5.4. Abbreviation of function and variable names

One of CA-Clipper limitations is that only the first ten characters of a function name are used by the CA-Clipper compiler. This means that in CA-Clipper, the following code is correct:

```
PROCEDURE Main()

   ? TestThisFu()

RETURN

FUNCTION TestThisFunction()

RETURN "Test"
```

However, in xHarbour, as the maximum length of a function or variable name is 64 characters, compiling and linking the above sample will give an unresolved external error by xHarbour's linker:

```
xLINK: error: Unresolved external symbol '_HB_FUN_TESTTHISFU'.
```

For CA-Clipper, TestThisFunction() is exactly the same as TestThisFu(). But not for xHarbour. This is a frequently asked question on the FiveWin and xHarbour newsgroups.

Now, let's try to compile and link the sample file "testini2.prg". You will get an unresolved external link error that looks like this:

```
xLINK: error: Unresolved external symbol '_HB_FUN_WRITEPPROS'.
```

This is because the "testini2.prg" sample uses the WritePPros() function, which is a perfectly valid function for your CA-Clipper application as it is the short name for FiveWin's function "WritePProString()". But in xHarbour, WritePPros() is NOT the same as WritePProString(). So, if you used WritePPros() somewhere in your source code, change it to WritePProString() to avoid unresolved external errors when building with xHarbour.

The same is true for variable names. CA-Clipper trims variable names to 10 characters. xHarbour trims them to 64 characters. So, this code is perfectly valid CA-Clipper code. It will run just fine.

```
PROCEDURE Main()

   LOCAL cThisVariable:="Test"

   ? cThisVaria

RETURN
```

However, running this code as an xHarbour compiled application you will produce a runtime error like this:

```
Error BASE/1003  Variable does not exist: CTHISVARIA
```

This is because, for xHarbour, variable "cThisVaria" is NOT the same as variable "cThisVariable"!

5.5. Using ADS (Advantage Database Server)

When your FiveWin application expects to use the Advantage Database Server as its RDD, you will need different ADS files to deploy with your application. The 16-bit CA-Clipper ADS .DLL files cannot be used with a 32-bit xHarbour Builder application. The following files need to be deployed with your final application if you use the ADS LOCAL server:

- ace32.dll
  Advantage Client Engine DLL.
- adsloc32.dll
  Advantage Local Server DLL.
- adslocal.cfg
  Advantage Local Server DLL configuration file.

If your application also uses ADS in its client/server mode, then you also need to deploy “accws32.dll” which is Advantage’s remote communication library.

These DLLs should be installed in the Windows search path of your user’s computer. A common place to install .DLLs is in the “Windows” or “Windows\System32” folder. In order to avoid “DLL Hell”, Microsoft now prefers all associated DLL files to be placed in their application’s folder.

This small sample uses ADS in local mode:

```
#include "FiveWin.ch"
#include "ads.ch"

REQUEST _ADS
EXTERNAL ADSKeyCount, ADSKeyNo

PROCEDURE Main()

   rddRegister( "ADS", 1 )
   rddsetdefault( "ADS" )
   ADSLocking(.f.)

   SET SERVER LOCAL
   SET FILETYPE TO NTX

   USE Test
   GO TOP
   Browse()
   CLOSE
```

```
RETURN
```

Note:
You may NOT use non standard functions in Index Expressions when using ADS.

5.6. Using ApolloRDD replacing SIX lib

If you have used Successware's HiPer-SIx driver as your RDD in your application, you now need to switch to ApolloRDD which uses Vista Software's Apollo database driver. The latest HiPer-SIx version is 3.02 and is only available for CA-Clipper version 5.2x, thus it is not usable with xHarbour Builder.

In July 1999 Vista Software bought the HiPer-SIx technology from Luxent Development Corp (formerly Successware). With that technology, Vista Software developed "Apollo", a line of high-performance, easy-to-use xBase data engines. Apollo is designed for developers building Windows applications with Delphi, VB, VC++ and more. Apollo supports FoxPro (DBF/FPT/CDX/IDX), Clipper (DBF/DBT/NTX/NSX) and HiPer-SIx (DBF/SMT/NSX) file formats.

xHarbour.com Inc. then developed ApolloRDD, the RDD for xHarbour that uses Vista Software's Apollo database engine. With ApolloRDD you can keep using your HiPer-SIx data files because the file format is exactly the same as Successware's HiPer-SIx driver for CA-Clipper 5.2x. All of the functions available in HiPer-SIx for CA-Clipper are also available in ApolloRDD.

Apollo's help file can be found in the xHarbour Builder installation directory under the "doc" folder, or via the xHarbour Builder menu folder. Look for "xHarbour Builder ApolloRDD documentation" under the "Documentation" folder. Not all of the Apollo xBase data engine functions are available in ApolloRDD because some functions do not apply to xHarbour Builder. However, HiPer-SIx functions and commands are all supported.

ApolloRDD only works in LOCAL mode for now. The client/server version will be available as soon as Apollo version 7 is released and found to be stable.

If you use ApolloRDD in your xHarbour Builder applications you will need to deploy the following .DLL files with your application:

- sde61.dll
  ApolloRDD's core DLL.
- sdentx61.dll
  Required if you use CA-Clipper tables and indexes in your application.

- sdecdx61.dll
  Required if you use FoxPro tables and indexes in your application.
- sdensx61.dll
  Required if you use HiPer-SIx tables and indexes in your application.
- fts32.dll
  Required if you use Fast Text Search in your application.

This is a small sample that uses ApolloRDD:

```
#include "FiveWin.ch"
#include "apollo.ch"

REQUEST _SIX

PROCEDURE Main

   rddRegister( "SIX", 1 )
   rddsetdefault( "SIX" )

   SET FILETYPE TO NSX

   USE Test
   GO TOP
   Browse()
   CLOSE

RETURN
```

You will notice from this sample that the HiPer-SIx include files for CA-Clipper 5.2x, "SIXNSX.CH" and "MACHSIX.CH", are not needed for ApolloRDD. Instead, you must use “apollo.ch” as your include file for ApolloRDD.

Note:
You may NOT use non standard functions in Index Expressions when using Apollo.

5.7. ZIP functions

FiveWin for CA-Clipper uses WIZIP16.DLL for its ZIP functionality. As WIZIP16.DLL is for 16-bit applications only, this DLL is not suitable for any applications to be built with xHarbour Builder.

xHarbour Builder has it’s own ZIP and UNZIP functions. To be able to use these, your application must be deployed with the file “xHBZipdll.dll”.,

Following is a list of all ZIP and UNZIP functions available via “xHBZipdll.dll”:

```
Zipcreate()
ZipOpen()
ZipAddFiles()
ZipAddFile()
ZipClose()
ZipExtractFiles()
ZipSetFiles()
ZipSetCompressLevel()      ZipGetCompressLevel()
ZipSetReadOnly()           ZipGetReadOnly()
ZipSetAct()                ZipGetAct()
ZipSetGlobalComment()      ZipGetGlobalComment()
ZipSetFilePath()           ZipGetFilePath()
ZipSetPassword()           ZipGetPassword()
ZipSetBuffer()             ZipGetBuffer()
ZipSetOnDisk()             ZipGetOnDisk()
ZipSetExtractPath()        ZipGetExtractPath()
```

A more in-depth explanation of these functions can be found in the “Programming Guide for xHarbour Builder”.

5.8. Converting your .RC resource files

Resources such as dialogs, bitmaps and cursors saved in .RC files must be converted before they can be used by xHarbour Builder. This is as simple as adding the name of the .RC file to your xBuild project file. This can be done by dragging your .RC file onto the screen of step three in xBuild Project Builder. xBuild will than compile your .RC file and include the compiled .RES in your application file. As the xHarbour Builder Resource compiler is compatible with both Borland and Microsoft products, compiling your .RC file should be simple.

If you store your resources in a separate .DLL file, you have two options.

1) Convert your .DLL resource file to a .RC file.
   For this option, you need to open your .DLL resource file with Borland’s Resource Workshop. Once opened, select the “Save file as…” choice from the “File” menu. Enter the file name of your .RC file and set the “File type” to “RC Resource script”. Press “OK” to save your .DLL resource as a .RC resource file. Once this is done, you can load the newly saved .RC file into your xBuild Project Builder.

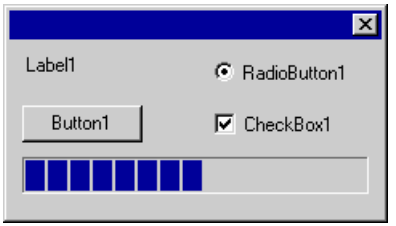
2) Convert your 16-bit .DLL to a 32-bit .DLL resource file.

In the DLL folder of your FWH installation directory, there is a file called “screen32.dll”. Copy the file “screen32.dll” into your project directory where your original 16-bit .DLL resource is located. Rename the “screen32.dll” file to your desired file name. For example, if your application is called “myApp.exe” and you use “myApp.DLL” as your resource holder, rename “screen32.dll” as “myApp32.DLL”.
Now convert your 16-bit .DLL file into an .RC file by following the instructions of option 1 above. Next, open the newly renamed screen32.dll file in Borland’s Resource Workshop. Then select the “Add to project…” choice from the “File” menu. Enter the .RC file name that you have just converted from your original 16-bit .DLL in the “File name” field and press the “OK” button. Borland Workshop will add your .RC file to the newly renamed screen32.dll. Finally, save the newly renamed screen32.dll file by using the “Save project” choice in the “File” menu.

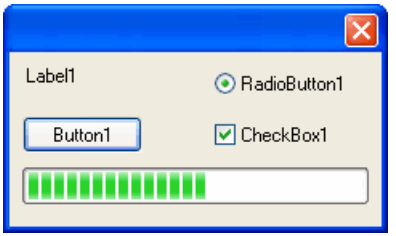
5.9. Windows 32-bit controls

If you want your application to have the same look and feel of a true Windows XP application, you will need to include an application manifest into your application. With this, Windows XP’s theme manager will control the look and feel of all your 32-bit controls.

Your current CA-Clipper + FiveWin application has this look now:



While a true 32-bit Windows XP application looks like this:



So, how do you get this Windows XP look and feel? As stated above, you need to include an application manifest into your application. A manifest file is an XML file that is included in your application as a resource. It can also be deployed as a separate file that resides in the executable file's directory.

If you want the manifest to be included as a resource, add this line to your resource file

```
1 24 "Your.App.Name.XML"
```

Where “Your.App.Name.XML” is the name of the manifest file.

The manifest file contents look like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly
  xmlns="urn:schemas-microsoft-com:asm.v1"
  manifestVersion="1.0">
<assemblyIdentity
    name="Your.App.Name"
    processorArchitecture="x86"
    version="5.1.0.0"
    type="win32"/>
<description>Windows Shell</description>
<dependency>
    <dependentAssembly>
        <assemblyIdentity
            type="win32"
            name="Microsoft.Windows.Common-Controls"
            version="6.0.0.0"
            processorArchitecture="x86"
            publicKeyToken="6595b64144ccf1df"
            language="*"
        />
    </dependentAssembly>
</dependency>
</assembly>
```

This file can also be found in the samples folder of your xHarbour Builder installation directory. Look for “Manifest.xml”.

For this to work, small changes are needed in your resource files. Have a look at the following table. The “16-bit FiveWin” column holds the resource type you were using in your CA-Clipper + Fivewin application. The “32-bit FWH” column holds the resource type you will need to use for your xHarbour Builder + FWH application to produce the Windows XP look and feel.

Win32 Control name	16-bit FiveWin	32 bit FWH
TTabCtrl	TTab	TFolder
TFolder	TFolder	TSysTabClt32
TImageList	*Not available*	*Not available*
TRichEdit	TFGet	TRichEdit
TTrackBar	*Not available*	*Not available*
TProgressBar	TMeter	TMeter
TUpDown	TGet with spinner	TGet with spinner
TSysAnimate	TAnimate	SysAnimate
TDatePicker	*Not available*	TDatePicker
TTreeView	TWBrowse	TTreeView
TListView	*Not available*	*Not available*
TColumnHeader	*Not available*	THeader

5.10. “Cannot create Dialog Box”

In a 16-bit CA-Clipper + FiveWin application, you can define a control in your resource and then NOT define it in your source code. When running the application, the control will not show. However, if you do NOT define a resource that exists in one of your resources you will get this error:

```
Error FiveWin/3 Cannot create Dialog Box: Resource: Main
```

when running the application. Where “Main” in this error message is the function name where the dialog or window was activated.

6. xPrompt

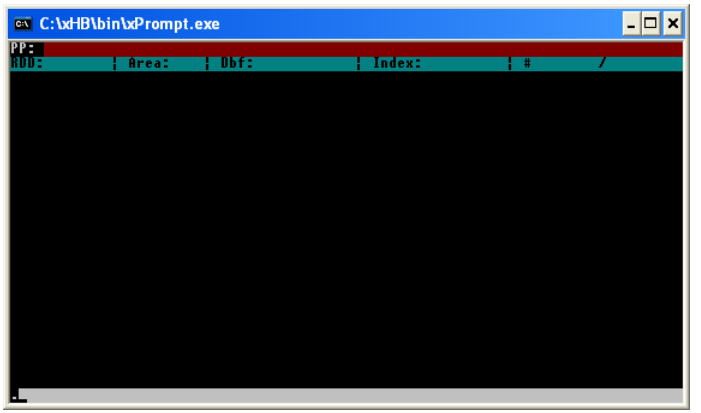
xPrompt is xHarbour Builder's utility for working with your data. xPrompt is an xHarbour application that provides:

- A Dot Prompt console that allows interactive use of most of the xHarbour / Clipper syntax.
- A 100% Clipper compatible Pre-Processor.
- A Clipper / xHarbour / xBase interpreter.

6.1. xPrompt as a DOT prompt

To open xPrompt in DOT prompt mode, go to the xHarbour Builder Installation folder and click on the "xPrompt" icon:

xPrompt will open like this:



In this mode you can execute a single line at a time by typing the line and pressing the ENTER key. Any xHarbour command and/or function are accepted.

To exit DOT prompt mode, type EXIT and press <ENTER>

6.1.1. Opening a database file

As xPrompt can accept all xHarbour commands and functions, it is very simple to open a database file with xPrompt. At the command prompt in xPrompt, type:

```
USE .\samples\test.dbf
```

And press enter.

Something interesting happened here! In the upper row of xPrompt, the pre-processed output of your given command is shown like this:

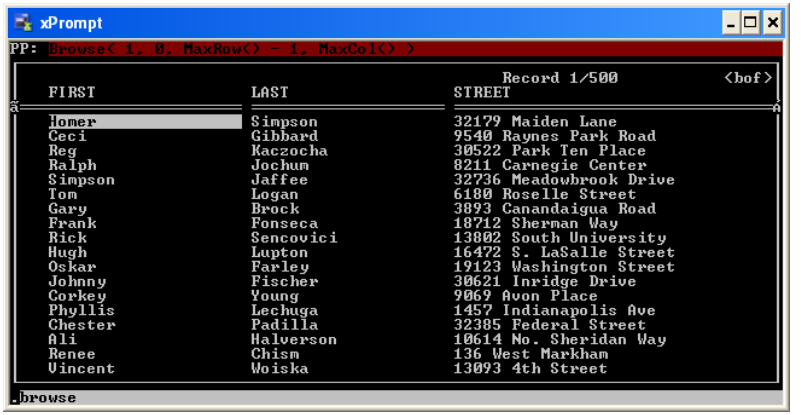
```
dbUseArea( .f., ,".\samples\test.dbf", , if(.F. .OR. .F., !.F.,
NIL), .F.)
```

The 2nd row shows the settings of the opened file.

The RDD used for opening "test.dbf" is "DBFNTX", opened in area 1 with name "TEST" and no index open. The current record is 1 of a total of 500 records.

From here on, any command and / or function can be used to manipulate or view the contents of the database.

Typing "LIST" in the command prompt and pressing <ENTER> shows a list of all fields of all records in the database. If you want to have more control over your list of records, use the BROWSE command. Browsing TEST.DBF looks like this:



Again, notice the pre-processed output for the "BROWSE" command.

6.1.2. Run a external source file

xPrompt can also be used to run external source files. xPrompt will pre-process the given PRG and execute its code. This interpreter mode is subject to a few limitations though:

- Support for LOCAL / STATIC / PRIVATE / PUBLIC variables, but:
  - STATICs are actually implemented as Publics.
  - LOCALs have scoping of locals, but are implemented as privates. This is why you can not have LOCAL and PRIVATE variables with the same name.
- Non-declared variables are auto-created on assignment.
- It does support definition and execution of prg-defined FUNCTIONS and/or PROCEDURES
- It does support ALL control flow structures including BEGIN SEQUENCE [BREAK] [RECOVER] END SEQUENCE and TRY [CATCH [<xCatcher>]] END.

Note:
xPrompt uses pre-burned rules to pre-process PRG code, unless a file named ***rp_dot.ch*** *is present. If rp_dot.ch is present, xPrompt will load rp_dot.ch instead of using the pre-burned rules. If this file doesn't include all the needed rules, the functionality of the Dot Prompt mode may be faulty.*

To run an external PRG source file from the DOT prompt mode, type:

```
DO <filename.prg>
```

Followed by <ENTER>

6.2. xPrompt as a Pre-processor

xPrompt is a 100% compatible Clipper pre-processor with some xHarbour extensions.

To pre-process your source file, go to a command prompt and type

```
xPrompt <filename[.ext]> -P
```

This will instruct xPrompt to read <filename[.ext]> and pre-process it. The pre-processed code is saved in <filename.pp$> which is the exact equivalent of the Clipper <filename.ppo> fie.

Note:
*If no filename extension is used, **.PRG** is used as extension.*

Following optional command line switches can be used in this mode:

-CCH = Generate a .cch file (compiled command header).
-D<id> = #define <id>.
-D:E = Show tracing information into the Expression Scanner.
-D:M = Show tracing information into the Match Engine.
-D:P = Show tracing information into the Output Generator.
-FIX = Do not clone Clipper Pre-Processor bugs.
-I<path> = #include file search path(s) (';' separated).
-U = Use command definitions set in <ch-file> (or none).

6.3. xPrompt as an Interpreter.

xPrompt can be used as a full featured Clipper / xHarbour / xBase Interpreter. Though this interpreter mode is subject to a few limitations:

- Support for LOCAL / STATIC / PRIVATE / PUBLIC variables, but:
  - STATICs are actually implemented as Publics.
  - LOCALs have scoping of locals, but are implemented as privates. This is why you can not have LOCAL and PRIVATE variables with the same name.
- Non-declared variables are auto-created on assignment.
- It does support definition and execution of prg-defined FUNCTIONS and/or PROCEDURES
- It does support ALL control flow structures including BEGIN SEQUENCE [BREAK] [RECOVER] END SEQUENCE and TRY [CATCH [<xCatcher>]] END.
- The compiled module is automatically using -n (No implicit startup procedure) if the script starts with a Procedure/Function definition.

To execute your source file, go to a command prompt and type

```
xPrompt <filename[.ext]> -R
```

This will instruct xPrompt to read <filename[.ext]>, pre-process and run it.
Note:
*If no filename extension is used, **.PRG** is used as extension.*

In this mode these are the optional command line switches.

-CCH = Generate a .cch file (compiled command header).
-D<id> = #define <id>.
-D:E = Show tracing information into the Expression Scanner.
-D:M = Show tracing information into the Match Engine.
-D:P = Show tracing information into the Output Generator.
-FIX = Do not clone Clipper Preprocessor bugs.
-I<path> = #include file search path(s) (';' separated).
-P = Generate .pp$ pre-processed output file.
-U = Use command definitions set in <ch-file> (or none).

Note:
xPrompt uses pre-burned rules to pre-process PRG code, unless a file named ***rp_dot.ch*** *is present. If rp_dot.ch is present, xPrompt will load rp_dot.ch instead of using the pre-burned rules. If this file doesn't include all the needed rules, the functionality of the Dot Prompt mode may be faulty.*

6.3.1. Using pp.prg as scripting engine

If you want to use this xPrompt interpreter functionality in your own code, you need to link pp.prg into your application. Just drag and drop pp.prg into your xBuild Project window.

Run your script with following function calls:

```
PP_InitStd()  // Init rules
PP_LoadRun()  // Init Run rules
PP_Run("<filename[.ext]>")
```

PP_Run() has these parameters:

```
PP_Run( <cFile>, <aParams>, <sPPOExt>, <bBlanks> )
```

<cFile> = The scripting file to run.
<aParams> = An array of the parameters to pass to <cFile>.
<sPPOExt> = The extension of the PPO output file.
<bBlanks> = .T. allows blank lines in the pre processed output.

More on PP_* functions in our “Programming Guide for xHarbour Builder”.