

# Clarion Magazine

## Clarion News

---

- » [CapeSoft's Clarion Version Support](#)
- » [CapeSoft Email Server Runs As Service](#)
- » [NetTalk 4 Adds SSL, Web Server](#)
- » [VuDeploy 3.0](#)
- » [Beyond Compare 2.4](#)
- » [GoToLunch Batch Compiler Supports Command Line Parameters](#)
- » [Australian DevCon Visa Information](#)
- » [Belarc Advisor Does Free PC Audit](#)
- » [PrintWindow Build 111](#)
- » [RADFusion's New Look](#)
- » [Huenuleufu Back Online](#)
- » [NetTalk 4 Web Server Demo](#)
- » [Dan Pressnell's Better SQL, Better OOP Source](#)
- » [SoftVelocity News Server Moving, DNS Changes May Take Time](#)
- » [EasyResizeAndSplit 2.12](#)
- » [RADFusion Site Under Maintenance](#)
- » [Dictionary Assistant 2.11](#)
- » [Clarion ProScan](#)
- » [NetTalk Web Server](#)

[\[More news\]](#)

## Podcast

---



[\[Track lists, more podcasts\]](#)

## Latest Free Content

---

- o » [Free FTP Client With Source](#)

[\[More free articles\]](#)

## Clarion Sites

---

## Clarion Blogs

---

**Save up to 50% off ebooks.  
Subscription has its rewards.**



## Latest Subscriber Content

---

### [Understanding The Clarion 6 Version Control Interface](#)

Clarion 6's version control interface made it possible to use all kinds of version control systems, but Benjamin Dell discovered the VCS products he tried did not give him nearly the same capabilities that the Clarion 5.5 VCS system had for concurrent developer control. That led him to an investigation of what the cryptic VCS help information in the Clarion Help file actually meant.

Posted Thursday, March 16, 2006

### [Free FTP Client With Source \(free article\)](#)

In 2003 Veronica Chapman wrote a number of highly informative articles for Clarion Magazine on the Windows API, including several on FTP transfers. Veronica's back with a complete FTP client package for Clarion, for all versions from Clarion for Windows 2.0 and up. The package is complete with a reference manual, client application, and source code.

Posted Friday, March 10, 2006

### **[PROP:SQL And Embedded Single Quotes](#)**

If you use MS-SQL with your Clarion programs, and you write your own SQL statements, then you've probably run into problems with quote characters. SQL statements use quote characters to delimit streams, and if you have quotes inside your strings you need to double them up so MS-SQL doesn't treat them as the end of the string. This article by John Griffiths will show you a few quoting tricks, and will also provide you with a handy function - SingleQuoteDoubler()

Posted Friday, March 10, 2006

### **[Writing To A Printer Port: Sending Escape Codes](#)**

The problem: How to send a control code (a.k.a. escape sequence or printer control) to a printer? The Clarion report structure no longer supports sending embedded control codes. There is no Clarion statement that allows sending them, at least not since CDD. Steve Parker explores the mystery of talking directly to Windows printers.

Posted Wednesday, March 08, 2006

### **[The Clarion Challenge Returns!](#)**

The Clarion Challenge is back! We're looking for some lean, mean code to strip link tags from HTML. The winner will receive an exclusive Clarion Magazine/Planet Clarion coffee mug or beer stein.

Posted Thursday, March 02, 2006

### **[Encrypting Data With Number Base Conversion](#)**

Most programmers are familiar with base 2 (binary) numbers, base 10 (decimal) and base 16 (hex). But using arbitrary number bases can also be a useful way to encrypt data for transmission by phone or other voice methods. Dermot Herron shows how he applies this technique to transfer vital information to customers.

Posted Thursday, March 02, 2006

[\[Last 10 articles\]](#) [\[Last 25 articles\]](#) [\[All content\]](#)

## **Printed Books & E-Books**

---

### **[E-Books](#)**

E-books are another great way to get the information you want from Clarion Magazine. Your time is valuable; with our [e-books](#), you spend less time hunting down the information

you need. We're constantly collecting the best Clarion Magazine articles by top developers into themed PDFs, so you'll always have a ready reference for your favorite Clarion development topics.

## [Printed Books](#)

As handy as the Clarion Magazine web site is, sometimes you just want to read articles in print. We've collected some of the best ClarionMag articles into the following print books:



- Clarion 6 Tips & Techniques Volume 1 - ISBN: 0-9689553-8-X
- Clarion 5.x Tips and Techniques, Volume 1 - ISBN: 0-9689553-5-5
- Clarion 5.x Tips and Techniques, Volume 2 - ISBN: 0-9689553-6-3
- Clarion Databases & SQL - ISBN: 0-9689553-3-9

We also publish Russ Eggen's widely-acclaimed [Programming Objects in Clarion](#), an introduction to OOP and ABC.

## **From The Publisher**

---

### [About Clarion Magazine](#)

Clarion Magazine is your premier source for news about, and in-depth articles on Clarion software development. We publish articles by many of the leading developers in the Clarion community, covering subjects from everyday programming tasks to specialized techniques you won't learn anywhere else. Whether you're just getting started with Clarion, or are a seasoned veteran, Clarion Magazine has the information *you* need.

### [Subscriptions](#)

While we do publish some free content, most Clarion Magazine articles are for subscribers only. Your [subscription](#) not only gets you premium content in the form of new articles, it also includes all the back issues. Our [search engine](#) lets you do simple or complex searches on both articles and news items. Subscribers can also post questions and comments directly to articles.

### [Satisfaction Guaranteed](#)

For just pennies per day you can have this wealth of Clarion development information at your fingertips. Your Clarion magazine subscription will more than [pay for itself](#) - you have my personal guarantee.

## Dave Harms

# Clarion Magazine

## Free FTP Client With Source

Published 2006-03-10

In 2003 Veronica Chapman wrote a number of highly informative [articles](#) for Clarion Magazine on the Windows API, including several on FTP transfers. Veronica's back with a complete FTP client package for Clarion, for all versions from Clarion for Windows 2.0 and up. The package is complete with a reference manual, client application, and source code. From the help file's background notes:

This Package 'fell out of' my need to organise the various WebSites with which I am involved (in fighting the New World Order). I required a simple way of constructing WebPages (and other 'artefacts', such as images, setups, zips, etc,) and automatically uploading them to their defined places, on Servers, without having to worry I was choosing the right place each time. The reason I mention this is because the features and facilities afforded by this package enable me to:

- a. Scan any WebSite under my control, and create a full list of every Folder and File thereupon, where it is (i.e. in which Folder), the Size, Last Access Date/Time of each, and the Access Permissions. I can also scan my Local Platform 'mirror' and reconcile each to each.
- b. Arrange to readily discover and adjust any situation whereby my Local and Remote Copies ever become inconsistent.

By this means I was recently able to move a WebSite from one WebHosting to another, firstly by automatically Downloading everything to a 'null' Local Platform 'mirror'. And then by automatically Uploading, from this 'mirror', to a new WebHosting. The entire process ran virtually automatically.

There are several ways to get the source code.

- In a [setup file](#) via Veronica's site. You can also get just the [help file](#) here. This is the recommended way to get the CW 2.x (and forward) compatible source.
- If you have problems downloading the file as above, I have a [copy of the setup file](#) (as of March 10, 2006) on Clarion Magazine. As time goes on this may or may not be the latest version.
- You can also get a [C6 APP and sources](#) from Clarion Magazine. I ported the 2.0 code via TXA, and made some adjustments based on new embed locations and standard equates. You will need the Clarion template chain registered to open this app. A TXA is included in the event you need to back port to an earlier version of Clarion - this may be easier than moving the CW 2.0 TXA forward.

If you're reading this at some later date, you can always check for changes by downloading the [ClarionMag copy](#) of the 2.0 code, and the [latest version](#) of the 2.0 code, and then porting those changes to the [C6 version](#). I'm a big fan of [Beyond Compare](#) for this kind of work - it excels at comparing directories and files.

# Clarion Magazine

## Understanding The Clarion 6 Version Control Interface

by Benjamin Dell

Published 2006-03-16

Why is version control important, and how can it be useful to me?

The term Version Control System (VCS) is defined as a means of tracking various versions of a set of *files*. These may be any files, not necessarily source code, and can include icons, templates, DLLs, etc.

So here you are, writing the next great killer application. Maybe you are by yourself, or you only have one other developer to worry about. Why should you bother with version control software? After all, you are probably used to being able to modify anything you want at any time, and the worst you might have to do is tell the other person that you are going to work on some code now, so don't touch it until done.

Why give up all this freedom?

Well, the most important information available to you in a version control system is the *latest version* of your source code and/or development environment. Another benefit is to know *what files* are in use. You have maybe twenty test files for testing code and only nine files for actual product development. If you *add* your genuine source files to the VCS system, you *know* what files are in use and which is not.

Sometimes even with the best of intentions we decide to do the wrong thing and wish we could go back to having the code as it was before we did the changes. Searching a VCS for a module or procedure is so much easier than going through Zip files or layers of backed up app files trying to find the one that you want.



## The Clarion 6 Version Control System

Clarion 5.5 shipped with a readymade and "Clarionized" VCS. Clarion 6 introduced the concept of "you are in charge of your own versioning management," thereby making us all go learn new skills in the open source and affordable VCS marketplace.

Some of us are too lazy to learn new skills when we have such nice development environment to keep us busy, and simply stopped using version control.

Now I am all for new technology, learning new skills (my wife says it keeps me young) and trying all crazy and weird things, but when I remember what Clarion 5.5 Version Control brought to the table, I just have to salute the people at UTA for their excellent work.

I had been using Clarion 5.5's version control in each and every small and large project I did, be it single developer or multi-developer projects. I really, really wanted the same ease of operation and RAD-ness in Clarion 6, which I enjoyed in Clarion 5.5.

I learned Subversion, GNU-VCS, etc, but I discovered they did not give me nearly the same capabilities that the Clarion 5.5 VCS system had for concurrent developer control. Out of frustration, I decided that I first need to find out what the cryptic VCS help information in the Clarion Help file actually meant, and if there was anything I could do to make my life easier.

This is a diary of that journey. Along the way I will tell you what I learned, what seemed good, what astounded me and what I decided to do about it....and give birth to a new baby called RVCS – [Right Version Control System](#) – because it feels so right for me to use. I should emphasise that you do not need RVCS to do version control with Clarion, or to benefit from this article.

## The in and outs of Clarion 6's VCS system

The VCS support in Clarion 6 is designed to support any VCS system that has a command line interface. The Clarion 6 VCS system also supports Module level control of source files instead of procedure level control found in previous Clarion versions.

Another change in Clarion 6 VCS is that you can only version control files for which the IDE can generate TXA or TXD files. This means in effect that you can not version control your development environment using the built-in VCS system interface, although this is still

possible by making use of the command interface (visually or via batch files) for your VCS system of choice. (See David Harms' article [Putting Clarion 6 Under Version Control With TortoiseSVN](#) for more information on this subject.)

To start off learning the Clarion 6 VCS interface I started by putting the trusty old People example application under version control.

The Clarion 6 VCS interface does the following version control tasks (leaving aside for the moment the issue of creating a repository to hold the versioned files):

1. It takes each and every module for the current application and creates a separate TXA file for the module, and then it also creates a TXA file to contain the application options.
2. It writes these TXA files one at a time to the directory location specified in the setup of the VCS interface. (See Fig 1 below)



**Figure 1. VCS repository database location**

3. Then it contacts the third party VCS system to tell it what file was put in the directory and what to do with it, and it waits for an answer before it continues with the next file.

All this sounds very good, but in a multi developer situation, how does one developer create the VCS database and the other developer know about this database and use the files in the VCS database?

Here's where you need some careful planning.

You see, the Clarion 6 VCS interface can only work on an application or dictionary that is *already loaded* into the IDE. This differs from the Clarion 5.5 method where the VCS files could be extracted (checkout), and worked upon without the application being loaded into the Clarion IDE.

After some thought, it seems the best way to achieve multi developer access to a single VCS database would then be to follow the following simple practice:

- Decide on a location that is accessible to all the developers via a DOS command or path statement. This would typically be a mapped drive on a remote computer.
- Make sure at least one developer has the latest and greatest source of the application that is of interest.
- This developer must compile the application to make sure there are no problems, bugs and other beasts floating around to make trouble later on.
- This developer creates the VCS database and does an *add* to the database for all the modules of the application.
- Repeat the above steps for all the applications and the dictionary that constitute the entire software development project.
- Once all the applications and dictionary are added to the VCS system, each of the concurrent developers gets an exact copy of the first developer's project directory. This is a very important step. If the developers do not copy the first developer's project directory, they will lose the 'project vcs' settings that were created when the project was put into VCS control, and will not be able to check out code correctly. (Their VCS systems will show the project as not checked in yet)
- Each developer sets up their VCS commands the same as the first developer's VCS commands in the Clarion 6 IDE.
- Each developer opens the current copy of their application and starts development. As they want to check something into the VCS database, they need to *check in* the work to the VCS system.

In a normal situation, each developer will then be doing work on the applications and *check in* their work to the VCS system as needed.

Once a full version of the applications needs to be compiled, a developer will have to *check out* all the modules of the application and compile it. When checking out, care needs to be taken that no work not yet checked in to the VCS database is overwritten with old work that is in the database repository. The reason for this checkout is not so much to mark who is working on what inside the third party VCS system, but to get all the code that all the developers have worked on in one central location for compilation.

So what happens if someone is doing a compile locally to test some code without having checked out the complete application(s) and dictionary? I don't know, and I never used the Clarion 6 VCS interface up to that point. I was getting too agitated with the many, many steps and checks and balances that I had to keep in mind to make the VCS system behave anywhere close to how I wanted it to behave in a multi-developer environment. I do know for a fact that no developer can make changes to another developer's code without checking that code out of the VCS system first..

What I want in a multi developer environment is the ability for any of the developers to *check out* the portion of the application they are going to be working on, *and* the rest of the application that someone else is already working on to be able to do a test compile on the latest version of the application.

I also want any developer to be able to check out a previous version of the code for comparison purposes or to get the *lost procedure code* that was so brilliant and went missing in the later versions.

All of these checkouts had to be controlled such that a wrong check in did not overwrite newer correct code with older code.

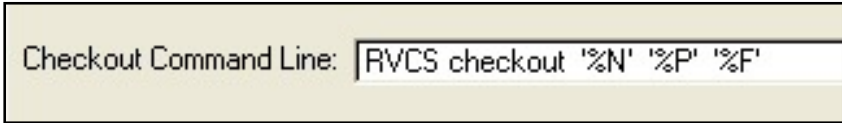
I played around with GNU VCS, Subversion and PVCS, but none really satisfied me as to what I could do without a great deal of trouble, and without leaving the Clarion 6 IDE.

Instead I decided to analyse the operations of the Clarion 6 IDE based VCS system in greater detail so that I can get an idea of how it really works, as the help files seems to cryptic for me and left me too much to my own guesses and conclusions.

## Details of operation

Coming back to the People example application, I decided to see what happens if I just used the Clarion 6 IDE interface for VCS without any third party VCS system attached.

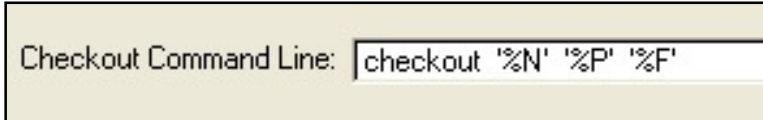
I created my Version Control Database as in Figure 1 above, and then cleared the VCS command names of any VCS application references. The command in Figure 2:



```
Checkout Command Line: RVCS checkout '%N' '%P' '%F'
```

**Figure 2. The default checkout command line**

was changed to:

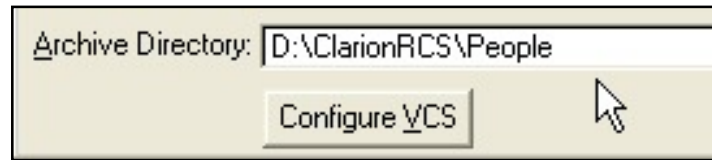


```
Checkout Command Line: checkout '%N' '%P' '%F'
```

**Figure 3. The modified checkout command line**

And behold....The Clarion IDE creates a few files of interest.

First it extracts a TXA file for Each and Every module in the current active Application (app) and also creates a TXA for the Application global settings and puts all these files in the path specified when setting up the version control system (Figure 4).



**Figure 4. The archive directory**

It also creates a file called People.AVC in the home directory of the currently loaded application, and a file called VCStemp.bat.

The People.AVC file is a very interesting file in that it holds a list of commands that was last executed by the VCS system. Below is a sample of the people.avc file.

```

Checkin Directory=D:\ClarionRCS\People
Project=People
Parts=5
Part1=Application Options
Part2=people.clw
Part3=peopl001.clw
Part4=peopl002.clw
Part5=peopl003
[State]
Application Options=No Action
people.clw=No Action
peopl001.clw=Check Out
peopl002.clw=Check Out
peopl003=Check Out
[ __Dont_Touch_Me__ ]
Sectors=0

```

These last actions are important as the VCS system needs to keep track of what it needs to do on the next *Check-Out* or *Check-In* action specified by the developer.

For instance, if the Peopl001.clw file was checked out, obviously it needs to be checked in on the next check in action by the developer, but people.clw will not be checked in as it was never checked out.

This is all very interesting, but brings me back to my multi user environment problem. Say Developer A checks out Peopl001.clw at 10:30, and Developer B checks out Peopl001.clw at 10:31. Developer A works on this module and adds now code. Developer B adds some comments to the current code.

Developer A checks in the code at 10:45 and Developer B check in the code at 10:47. What code is the *latest version*? The one last checked in.

Please note that the code for Developer A will not be lost, but the correct code may not be the latest version, as the old code Developer B checked in was the last code checked in with just new comments.

It would be nice to have some notice of this kind of conflict, and I will get to that a bit later on.

How about the VCStemp.bat file?

Well let me tell you. This is really a little beauty of good old DOS engineering.

The VCStemp.bat file gets called by the IDE VCS system to handle the actual work. This means the IDE VCS system loops through all the modules in the list to be checked in or out and calls the Batch file to actually do the work.

```
@PUSHD .
@RVCS checkout 'People' 'D:\ClarionRCS\People' 'peopl003.APV'
@POPD
@if %ERRORLEVEL% NEQ 0 pause
@if %ERRORLEVEL% NEQ 0 ECHO %ERRORLEVEL% > VCStemp.$$$
EXIT %ERRORLEVEL%
```

Let's examine the batch file commands.

```
@PUSHD .
```

`@PUSHD .` pushes a directory name onto the directory stack. Note the "." after the command. This means that the command pushes the *current working directory* (e.g. c:\clarion6\examples\people) onto the directory stack. `Pushd` always sets Clarion's current working directory to the directory on the top of the stack, so it can be restored later. Without this feature you would have to rely on the CVS to restore the original working directory.

```
@RVCS checkout 'People' 'D:\ClarionRCS\People' 'peopl003.APV'
```

Next, the third party VCS system is called to do its magic with the TXA files supplier. In this batch file, the third party VCS system is called via RVCS.exe. It is thus imperative that your Clarion environment has a path access to the VCS system that you install. If the VCS system you install does not register itself with the windows environment, you may have to do so yourself by setting the name using the PATH command, or registering it via the environment variables in the Windows control panel.

Note that the TXA file is now called APV according to the settings in the VCS setup window, in Figure 5.



**Figure 5. The new application and dictionary extensions**

Then the Clarion 6 IDE VCS system waits for the called third party VCS system to complete its work and return an error code.

If no DOS error code is returned by the third party VCS system, the Clarion6 VCS Interface assumes the file has been version controlled successfully.

If a DOS Error code is returned a typically cryptic VCS error is displayed, as in Figure 6.



**Figure 6. A typical VCS error message**

In this instance no mention is made of the fact that the third party VCS system has declared this an invalid option as no previous check-out has taken place.

**@POPD**

Remove an element from the current directory stack. With no options, POPD removes the top

directory stack element and sets the current working directory to the directory name of the new top element. In short this means that the current working directory is set back to the one stored on the directory stack by the pushd command.

```
@IF %ERRORLEVEL% NEQ 0 pause
```

If the DOS error level returned by the third party VCS system is not 0 (zero), the batch file is paused and a message "Press any key to continue..." is presented to the user in a DOS environment window.

```
@IF %ERRORLEVEL% NEQ 0 ECHO %ERRORLEVEL% > VCStemp. $$$
```

If the DOS error level returned by the third party VCS system is not 0 (zero), the error level is written to the file called VCStemp.\*\*\*. The reason for this may be that the Clarion 6 IDE VCS interface can pick up that there was an error and operate on it.

```
EXIT %ERRORLEVEL%
```

This is to exit the DOS command session and pass the error level, if any, back to the Clarion IDE.

As can be seen from the batch file, moving away from the DDE interfaces can be a good thing, as any third party VCS system can be controlled via this batch file.

## Exporting module TXAs

Another very useful function of the Clarion 6 VCS interface is that it exposes the application to the developer in a module level TXA file, something not available natively via the DDE system, and otherwise requiring a lot of work to do manually via Selective Exporting. This allows any developer to take the txa and txd files generated by the VCS system and use it in any way they see fit.

For example, say you don't have a third party VCS system available, and you want to do some really important coding that you are unsure will work, or which has the chance of producing IDE lockups, crashes or GPFs. Here's what you do:

1. Load your application and define your VCS Repository database.
2. ADD your application using the VCS interface.
3. Make your code changes in the application, compile it and run it.
4. If something goes horribly wrong, roll back your changes by CHECKING OUT



- the application or modules you were working on from the VCS repository.
5. If everything worked according to plan, CHECK IN your changed modules using the VCS interface.

## Summary

The Clarion 6 VCS Interface allows you too check out source code, work on the source code and check back in the new code. It also creates a TXD for the dictionary and a TXA file for each of the modules of the current application.

What it does not do is any error checking on who has the latest version of source code in a multi-developer environment.

VCS systems are a very valuable tool in both a single user and a multi user environment. Version control is as important as having regular backups of your development environment and source code. Take the time to study and use version control; these tools will save you many more hours that it took to learn how to use them.

Good luck to all your development endeavours.

For more information on VCS systems and Clarion, see the [Version Control](#) page in the [topical index](#).

## Reader Comments

[Add a comment](#)

# Clarion Magazine

## PROP:SQL And Embedded Single Quotes

by John L Griffiths

Published 2006-03-10

If you use MS-SQL with your Clarion programs, and you write your own SQL statements, then you've probably run into problems with quote characters. SQL statements use quote characters to delimit streams, and if you have quotes inside your strings you need to double them up so MS-SQL doesn't treat them as the end of the string. This article will show you a few quoting tricks, and will also provide you with a handy function – `SingleQuoteDoubler()`.

SoftVelocity does provide the `QUOTE` function, but `QUOTE` was built to mainly serve the needs of the Clarion compiler. The problem with `QUOTE` is that it isn't that smart about how it doubles up characters, as I'll explain shortly. Read more on `QUOTE` in the Clarion help.

My need showed up when users would enter free text in a text box, usually whilst on the phone with their clients. These users were adding notes which often contained words with single quotes, measurements shown as 5' 11" and amount < \$300 or rarely, curly braces around a word {perhaps}.

I was writing the data to the SQL database using `{prop:sql}` with a long `CSTRING`. Names such as O'Brien and D'Angelo or measurements like 5' 11" produced single embedded quotes within the `{prop:sql}` string. This would cause the insert statement to fail.

Here's the problem. I was building an insert statement string `{using PROP:SQL}` from various fields. The troublesome one was the `MyNote CSTRING` with the free form data.

Say the user entered : 'Mr O'Brien phoned today < 5PM.' I would build up my insert statement in code this way:

```
'SqlStr='INSERT tblNote (sysid,TheNote) values (' & CliSysid & ',<39>'
  & MyNote & '<39>')
```

and the resultant string looked like this:

```
'INSERT tblNote (sysid,TheNote) values (1234,'Mr O'Brien phoned today <
  5PM.')
```

Sending this string to the MS-SQL engine will fail, because of the embedded single quote in Mr O'Briens name. I need to double up any single quotes before sending the string with `{PROP:SQL}`, and for MS-SQL it needs to look like this:

```
'INSERT tblNote (sysid,TheNote)
  values (1234,'Mr O''Brien phoned today < 5PM.')
```

The `QUOTE` function will perform this doubling of the single quote. I can use it this way:

```
SqlStr='INSERT tblNote (sysid,TheNote)
values ( ' & CliSysid & ',<39>' & QUOTE(MyNote,0) & '<39>')
```

The problem now is, that QUOTE will also double up any less-than signs, and any left-curly braces. This would give the following INSERT command:

```
'INSERT tblNote (sysid,TheNote)
values (1234,'Mr O''Brien phoned today << 5PM.')
```

Another problem with QUOTE is that calling it again will double everything again.

Thus I found the need for a function to only double up the single quotes, and to only do it once.

Here is the source code for the function. It is also in the (CW 6.1) sample application file, and also in the attached example .zip as a CLW file.

```
SingleQuoteDoubler    PROCEDURE    (STRING p:str)    !,STRING

ii            LONG,AUTO
lenstr        LONG,AUTO
WorkStr       STRING(8000)
posit         SHORT,dim(8000)
ii2           LONG
OnEVEN        BYTE ,AUTO

CODE
workstr = p:str
lenstr = LEN(CLIP(workstr))
IF lenstr < 1    ! cant fix this one
    RETURN ''
END
LOOP ii = 1 TO lenstr
    IF VAL(workstr[ii]) = 39    ! a single quote here
        posit[ ii] = ii
        ii2 += 1                ! count number found
    END
END
IF ii2 = 0
    RETURN CLIP(p:str)    ! None Found
END
IF ii2 = 1                ! only found the one, so...
    LOOP ii = 1 to LenStr
        IF posit[ ii] > 0
            WorkStr = workstr[1: ii ] & '<39>' & workstr[ ii+1 : len(clip(workstr)) ]
            RETURN clip( workStr )
        END
    END
ELSE                ! Found more than 1
    OnEVEN = 00b    !false
    LOOP ii = (lenstr ) to 1 by -1
        IF posit[ii] > 0
            DO ThisPosition
        ELSE
            OnEven = 01b    !true
        END
    END
```

```

    END
  END
  RETURN clip(workStr)

ThisPosition routine
  OnEven = BXOR(OnEven , 01b ) ! TOGGLE IT
  CASE ii
  OF 2 TO (lenstr -1)
    IF oNeven = 01b
      EXIT
    END
    IF posit[ ii -1 ] = 0
      workstr = workstr[1: ii ] & '<39>' |
        & workstr[ ii+1 : len(clip(workstr)) ]
      EXIT
    END
  OF Lenstr
    ONeven = 00b ! MUST BE FOR FIRST ON AT TAIL
    IF posit[ ii -1 ] > 0 !nextlower is already quote
      EXIT
    END
    workstr = workstr[1: ii ] & '<39>'
  OF 1
    IF OnEven = 01b
      EXIT
    END
    IF posit[ 2 ] > 0
      EXIT
    END
    workstr = '<39>' & workstr[ 1 : LEN(CLIP(workstr)) ]
  END
END

```

The example application uses a single table dictionary to connect to the Customers table in the MS-SQL example database Northwind. The dictionary only includes a few of the fields from within the Customers table.

The application also builds a global scope temporary table in the MS-SQL database. This temporary table is only in existence until you close the example app. You can run the example, and add some notes with single quotes, less-than signs and left-curly braces. You may then view the temporary table (called ##TblNote) with Query Analyzer or Enterprise Manager *while* your program is still running.

The Application also uses dummy temporary files to talk to the database in several places.

Figure 1 shows a note being entered by the user.

CustomerID:

CompanyName:

ContactName:

Free Text area for note to be saved.. Include some single quotes, some '<' signs, and a left brace { or two...

### Figure 1. Entering a note

This note has one each of the problem characters. Figure 2 shows what is stored in the database.

My Note	CW Quote Note
1 < sign, 1 left brace {tst} and one quote O'Brien	1 << sign, 1 left brace {{tst}} and one quote O'Brien

Figure 2. The stored data

The text in the left hand column uses my function. The one on the right uses the Clarion QUOTE function.

## Summary

If you use {PROP:SQL} in your applications to send strings to the database, then you need to watch out for strings that may contain embedded single quotes. The Clarion QUOTE function may do the job for you, or you may find out, as I did, that a little custom code does the job better.

One final word of warning. You should not blindly accept free form text input from users when accessing a SQL database. There is the danger known as an "[SQL Insertion Attack](#)" (thanks to Russ Eggen for the link) whereby a disgruntled employee/user with a little background in SQL may attempt to alter or delete data.

[Download the source](#)

[John Griffiths](#), an Australian, has been developing with Clarion since the DOS days. John has two summers each year, spending six months in Australia for the Southern summer and six months in Texas for the Northern summer. He works as a contractor for a Texas company and has developed several business/financial programs which he sells worldwide. John has a B.Bus degree with a major in Informations Systems.

## Reader Comments

[Add a comment](#)

- [» What timing... I have been using quote and it has worked...](#)

# Clarion Magazine

## Writing To A Printer Port: Sending Escape Codes

by Steven Parker

Published 2006-03-08

The problem: How do I send a control code (a.k.a. escape sequence or printer control) to a printer? The Clarion report structure no longer supports sending embedded control codes. There is no Clarion statement that allows sending them, at least not since CDD.

In fact this problem is a problem because of Windows. Clarion certainly contributes to the problem. But, it is first and foremost caused by Windows.

How does Clarion contribute? In making the transition to Windows, TopSpeed changed the `PRINT( )` statement. In Clarion for DOS (all iterations, as far as I can recall), the `PRINT( )` statement immediately sent its argument directly to the printer. So, for example, embedding

```
Print('Hi there!')
```

caused the string "Hi there!" to go immediately to the printer. Similarly,

```
Print('<12>')
```

immediately sent a form feed ("printed" a form feed). Complete programmer control was as easy as:

```
If NDX = Records(ListQueue)
  Print('<12>')
End
```

In fact, a CDD developer was rarely, if ever, required to embed escape sequences in a report. Any control code, *any*, could be selected from the report formatter menu. Control codes or printers not supplied by TopSpeed could easily be added to a printer control file (which *had* to be distributed). This was the subject of my article "Printer Control in 3.0" (Clarion Tech Journal, 6, 2; March/April 1994), 12 years ago. Granted, Clarion for Windows does allow setting font attributes for one or more report controls, landscape/portrait orientation, etc., right in the report formatter, without resorting to control codes.

With some work, and remembering to call `SetTarget`, fonts can be affected dynamically at runtime. A good number of report properties can be affected with `SetTarget` at run time. With more work, a degree of program control of form feeds is possible (see, for example, [Reports: OOP, ABC and Ignoring Templates \(Part 3\)](#)). But what Clarion now allows me to do seems limited compared to the complete programmatic control I had in DOS. For example, try embedding the `Print ( '<12>' )` code snippet in a Windows report. It won't compile.

Microsoft itself, however, is primarily responsible for these difficulties. When Windows became the mediator between the program and the computer and between the computer and the printer, when Microsoft decided to isolate hardware and make direct communication with it all but impossible, strange things began to happen.

Hardware manufacturers eventually came to love this (it did take a while). Under Windows, hardware no longer needs to work quite so hard – printers can be "dumber" since the hard work of formatting documents is done in software. A vendor's hardware line can be more generic under the skin, reducing production costs.

The page to be printed is created by Windows, not the printer, not the user's application. Then it is dumped to the printer, again by Windows. Windows is specifically designed to isolate end user applications from hardware (and quite a lot else; in fact, I am beginning to think that the transition from line printing to page printing is the real conceptual mistake underlying the current conundrum). The printer "only" needs a way to translate Windows GDI commands into its own proprietary formats.

Enter the "driver."

Driver software, of course, is much cheaper to produce and distribute than precision hardware. On the other hand, producing stable and accurate drivers has turned out to be a somewhat more ambitious and elusive goal. Add to that, any bugs that may or may not

exist in the printer driver sit on top of any bugs that may exist in Windows. This wonderful symbiosis has evolved over the years so that it now seems that what ought to be an updated driver is accompanied by a new model printer. This new model, of course, is just the previous printer with some slight change (sometimes only the driver). The change makes the new printer incompatible with the previous model; the difference may be a slight as a renumbered printer cartridge. The HP Laser Jet III driver can be used on Laser 4s and the full family of 5s, for example, but this is a thing of the past now. An Epson C88 will not respond to the C86 driver even though they are essentially the same machine.

The fact is that developers simply cannot rely on two printers of the same make and model producing identical output. Two machines manufactured 20 seconds apart may render fonts slightly differently or a given stroke noticeably differently. These same two machines may have different non-printable areas. This last is a major and often reported problem.

I have seen non-printable areas on department class lasers (duty cycles of 5000 or more pages per month) vary from 1/8 inch to almost 1/2 inch. The spec for the machines is 1/4 inch. In DOS, a 1/4 inch unprintable margin meant exactly that, 1/4 inch. In Windows, it means "a smallish distance."

Trying to produce precision output, as required by labels or government or insurance forms can become quite a challenge to the hair line. And, these are cases where minor adjustments of left or right margins ("[nudging](#)") just won't do; moving the output slightly one way causes even more output to be cut off on the other side.

There are printer commands to "reset" the margins, to ignore the non-printable area in these kinds of printers ("Windows printers" do not, so far as I know, advertise their control codes; higher end machines do make this information available – though sometimes you have to search, vigorously, for it). Thus the current exercise: How to get these commands to the printer?

## Forward to the past

So, how in this Windows world do I do a DOS-ism? In fact, there are several ways. They fall into two basic categories: easy and hard. The hard way is to use the Windows API (there is another hard way; it involves the Spooler API but I think that is best left alone for a while). The easy ways, there are two that come immediately to mind, are wrappers for the hard way.



With a little help from my friends, however, using the API is not so terribly hard. The easy ways, therefore, are all that much easier. The friend, in this case, is Paul Attryde and his [The Clarion Insider](#) web site. Paul earns his daily bread using the API to enable applications to talk to hardware. So his advice and site are a most valuable resource.

Two of his Clarion Insider articles are particularly important to the task at hand: [16-bit Serial RS232 Communications](#) and [32-bit Serial RS232 Communications](#) (several of Paul's Clarion Insider articles are available in the Clarion Online archives here at Clarion Mag; these two are not among them as they never had a chance to appear in Clarion Online).

These articles are so important that I am going to assume you have read them both. *Both*. The [16-bit Serial RS232 Communications](#) article provides not only the foundation required for understanding [32-bit Serial RS232 Communications](#), it contains information that is still necessary in 32 bit Windows for accessing devices on serial ports (if serial ports interest you).

"Why 'good Doctor', are you referring us to articles on serial communication when our printers are parallel devices?" Good question. Easy answer. Accessing, outputting to and releasing a port is the same for both serial and parallel ports. Some aspects of serial ports setup, for example baud, stop bits and parity, are not required for parallel ports. In other words, those bits are irrelevant. But everything else is the same for both.

The key to writing directly to a port, Paul tells us, is that "for the 32-bit API Microsoft removed the port-specific APIs ... and made the file API calls of `CreateFile()`, `ReadFile()`, `WriteFile()` and `CloseHandle()` work with both files and ports."

So, the sequence to drop a control code directly to a printer is

```
CreateFile  
WriteFile
```

and

```
CloseHandle
```

Time to look at these three calls.

## CreateFile

Windows is a lot like Clarion. Did you know that?

In Clarion, everything has an FEQ (**F**ield **E**quate – look up "Field Equate Labels" in the on-line help; this is a seminal concept in Clarion).

In Windows, everything has a *handle*.

An FEQ is a number, the number of a control. A handle is a number, the number of a Windows object.

To use an API to address a Windows device, you need to know its handle (one rather significant difference is that Clarion not only creates FEQs but automatically makes them available as Field Equate Labels; in Windows, you have to go get them). In the case of ports, [CreateFile](#) is the API to get that handle: "The **CreateFile** function creates or opens a file, file stream, directory, physical disk, volume, console buffer, tape drive, communications resource, mailslot, or named pipe. The function returns a handle that can be used to access an object."

CreateFile (CreateFileA in ANSI, CreateFileW in Unicode) is one of those calls with a typically lengthy Microsoft prototype:

```
CreateFileA( *CSTRING lpFileName,          |
             ulong dwDesiredAccess,        |
             ulong dwShareMode,            |
             ulong lpSecurityAttributes,    |
             ulong dwCreationDisposition,  |
             ulong dwFlagsAndAttributes,   |
             ulong hTemplateFile),long, raw, pascal
```

**lpFileName** is a CString, passed by address. It is the variable containing the name of the port whose handle you want.

**dwDesiredAccess**: a DWord (ULong but a Long will do) indicating whether you want to read, write or do both. `Generic_Write (04000000H)` and `Generic_Read (08000000H)` and `Generic_Write + Generic_Read (0C000000H)` are the options. To simply output to a printer, `04000000H` will do.

**dwShareMode**: "The sharing mode of an object, which can be read, write, both, or

none." Printing is not a shared operation. So this parameter is "0" (zero).

**lpSecurityAttributes**: security attributes are not relevant. Zero. The same is true for **dwFlagsAndAttributes** and **hTemplateFile** (in fact, it must be zero for **hTemplateFile** when getting a handle to a port).

**dwCreationDisposition**: "An action to take on files that exist and do not exist." However, for ports, only `OPEN_EXISTING` (3) is permitted. So, "3" it is. (You can create a file programmatically but a new port requires a card and screw driver.)

So, a call to `CreateFile` looks like this:

```
PortName = 'LPT1:'
PortID = CreateFileA( PortName, 040000000H, 0, 0, 3, 0, 0 )
```

Yes, network printers can be accessed this way. From the DOS prompt, or in a batch file, do a `Net Use` to assign the network printer to a port name. Name that port in the `PortName` assignment. That's it.

Error checking? In this case, quite easy. If `CreateFile` fails, it returns -1. Otherwise, it returns the desired handle:

If the function succeeds, the return value is an open handle to a specified file. If a specified file exists before the function call and *dwCreationDisposition* is `CREATE_ALWAYS` or `OPEN_ALWAYS`, a call to **GetLastError** returns `ERROR_ALREADY_EXISTS`, even when the function succeeds. If a file does not exist before the call, **GetLastError** returns 0 (zero).

If the function fails, the return value is `INVALID_HANDLE_VALUE`. To get extended error information, call **GetLastError**.

So, a fully qualified call looks like this:

```
PortName = 'LPT1:'
PortID = CreateFileA( PortName, 040000000H, 0, 0, 3, 0, 0 )
If PortID = -1
    Message('Attempt to open ' & Clip(PortName) &|
           ' failed.', 'Error', ICON:Hand)
Return
```

End

## WriteFile

"The **WriteFile** function writes data to a file at the position specified by the file pointer. This function is designed for both synchronous and asynchronous operation." This is the call to do the actual writing.

[WriteFile](#), therefore, does the actually "printing" and, again, has a very involved prototype:

```
WriteFile(long hFile,          |
          *cstring lpbuffer,  |
          long nNumberOfButesToWrite, |
          *long lpNumberOfBytesWritten, |
          ulong lpOverlapped), short, raw, pascal
```

**hFile:** "Handle to the file. The file handle must have been created with the `GENERIC_WRITE` access right." In other words, this is the handle to the port gotten with `CreateFile`.

**lpBuffer:** "Pointer to the buffer containing the data to be written to the file." This, then, is a `CString`, passed by address. It is the variable containing the string to be printed. No directly passing a string; if you do, you get a compiler error. A `String` is not a `CString` and the compiler knows it.

**nNumberOfBytesToWrite:** "Number of bytes to be written to the file." I don't quite understand this one. The string to print is a `CString` so the length should be easily computed by the function. But, it is required.

**lpNumberOfBytesWritten:** "Pointer to the variable that receives the number of bytes written." I don't find this useful myself but it is required. It *could* be used to check that the number of characters printed matches what you think you sent in the previous parameter.

**lpOverlapped:** I had a link to a discussion of this on MSDN. I'm not sure the author understood this fully (since he seemed to contradict himself on the subject several times). He certainly couldn't explain it so that I understood it. Short take: don't bother, send zero.

**Return Value:** If the function succeeds, the return value is nonzero.

The full call, then, is:

```
StringToPrint = 'yada yada'           ! prime output
PrintLen = Len(Clip(StringToPrint))   ! get length
x = WriteFile(PortID,StringToPrint,|   ! write it
    PrintLen,Written,0)
```

## CloseHandle

As soon as possible after "printing," close the handle. If you don't, nothing else can use the device. Failing to close the handle is like LOCKing a file and not UNLOCKing it. Bad. Reboot required.

This is prototyped very straightforwardly:

```
CloseHandle( ulong hObject ), short, raw, pascal
```

taking the handle from `CreateFile` as its only parameter.

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call [GetLastError](#).

So the call to `CloseHandle` is:

```
x = CloseHandle(PortID)
```

Or,

```
If ~CloseHandle( PortID )
    Message( 'Whoops!' )
End
```

## The Sample App

As one might expect, the prototypes for the three API calls are in the Global Map embed. One datum is global, `PortID`. There is no particular reason I made this global. It could have and probably should have been local. Lazy me.

The one procedure in the `.APP` calls a window. You can enter a string to print. I made this string 40 characters long. 40? I presented this to the Chicago Area User Group using a receipt printer to demonstrate. A receipt printer has a line width of 40.

Everything occurs in the `Accepted` embed for the string. The code there is precisely what has been outlined above.

The only caveat is that this procedure is completely device-independent. That is, no formatting is done, the string is just written to the printer as is. Of course, that is the point.

Suppose that the escape sequence for some formatting that is needed, say to set a left margin, is `ESC 19`. This must be sent to the printer, "printed," before sending the text:

```
PortName = 'LPT1:'
PortID = CreateFileA( PortName, 04000000H, 0, 0, 3, 0, 0 )
Sequence = '<27>19'
X = WriteFile(PortID, Sequence, 3, Written, 0)
PrintLen = Len(Clip(StringToPrint))
x = WriteFile(PortID,StringToPrint,PrintLen,Written,0)
x = CloseHandle(PortID)
```

It would be a good idea to follow this with "printing" the control code to turn off the formatting. Just like `CDD`.

## Summary

The reader is urged to examine the resources that went into this article in detail:

- [16-bit Serial RS232 Communications](#) and [32-bit Serial RS232 Communications](#): Paul Attryde's guide to port communications, geared to serial ports.
- [32-bit Serial RS232 Communications](#): Paul's guide to the 32 bit API.
- `CreateFile`: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/fs/createfile.asp>

- WriteFile: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/fs/writefile.asp>
- Closehandle: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/sysinfo/base/closehandle.asp>

Sending escape sequences is hard to get your head around. These resources make it much, much easier.

Looking over the steps to "write directly to the printer," I see a real need for a wrapper procedure. I see a need for a single call, like PRINT ( ) in DOS, which takes what I want to print and where I want it to go.

Stay tuned, I think I will come back to that next. Afterwards I will look at the easy ways of talking directly to a printer.

[Download the source](#)

---

[Steve Parker](#) started his professional life as a Philosopher but now tries to imitate a Clarion developer. He has been attempting to subdue Clarion since 2007 (DOS, that is). He reports that, so far, Clarion is winning. Steve has been writing about Clarion since 1993.



## Reader Comments

[Add a comment](#)

- » [Simplest way: use the dos file driver Program ...](#)
- » [If you do not need to send a control code in the middle of...](#)
- » [Works with com ports too, just change the file name to...](#)
- » [Steve, The following comment intrigued me: "If you do...](#)
- » [I should certainly think so. I use the API to write to...](#)
- » [I should certainly think so. I use the API to write to...](#)
- » [This approach 'seems' to work great when writing to LPT...](#)
- » [This approach 'seems' to work great when writing to LPT...](#)
- » [USB? No, you can't handle output to USB printers using...](#)





# Clarion Magazine

## The Clarion Challenge Returns!

by Dave Harms

Published 2006-03-02

After a too-long absence, the Clarion Challenge is back! Sharpen your keyboards and put on your tin foil hats.

I recently rote some RSS aggregator code, which basically means I retrieve RSS feed items, dump them into a database, and then semi-automatically assign selected items to a new feed. Among other things, I decided I wanted to be able to automatically strip link tags out of the RSS item descriptions.

In its simplest form, an HTML link tag, more accurately called an Anchor tag, looks like something like this:

```
<a href="http://www.clarionmag.com/blog/">The ClarionMag Blog</a>
```

Within an HTML page, the above code will appear something like the following (depending on which styles are in use):

[The ClarionMag Blog](http://www.clarionmag.com/blog/)

Now an A tag can have a bunch of attributes in addition to href, including class, name, onclick, style, etc., but basically the first part of the A tag, before the text to be linked, begins with <a and ends with >. The closing part of the tag is simply </a>.

To strip A tags out of a block of HTML text, then, you simply remove everything from <a to > as well as the string </a>.

And that's the challenge. Write a function, or a class, to strip A tags. Points will be awarded for speed and elegance.

Here are few things to keep in mind:

- In HTML, tags may be upper case or lower case. Lower case is recommended, and in XHTML, all tag names *must* be lower case. But your code must be able to handle <a as well as <A.
- Pay attention to boundary conditions - test with tags at the start and end of your test string.
- Preserve spaces - don't remove anything other than the tags.

[Email your entries](#) to me by March 10. The winner will receive a Clarion Magazine/Planet Clarion [coffee mug or beer stein](#).

---

[David Harms](#) is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of *Developing Clarion for Windows Applications*, published by SAMS (1995), and has written or co-written several Java books. David produces the [Planet Clarion](#) podcast, which he co-hosts with Andrew Guidroz II.

## Reader Comments

[Add a comment](#)

- [» A couple of questions... are you expecting the original...](#)
- [» Geoff, The original string does not need to be...](#)
- [» Is this html on the web or is it a local text file? IOW,...](#)
- [» Chris, The function receives the HTML as a string; all...](#)
- [» Hi Dave Your comment that: "my own coding of this...](#)
- [» Dave, I'm sorry, but I don't quite understand your...](#)
- [» Geoff, I'm really not trying to trick you. It is quite a...](#)

- [» Charles, I meant to discard the tags themselves, so...](#)
- [» Dave, I think you want:](#)
- [» Russ, Not quite - I want](#)

-

-

# Clarion Magazine

## Encrypting Data With Number Base Conversion

by Dermot Herron

Published 2006-03-02

I sell software to send automatic SMSs (Short Message Service or "text" messages) to a cell-phone (mobile) to remind people to come for appointments. Each SMS sent deducts its cost from a pre-paid amount of money as it is sent. The software also expires after a pre-set date. To administer this I needed a simple system to verbally transfer semi-encrypted LONGs from a client, to me, and back again, to update the dates and money. Large number bases (i.e. a number base larger than the usual 10) are very compact when you need to transfer numeric information over the phone, or by other voice communication means. Others in the Johannesburg Clarion User Group have used this system to enable software after download.

It works like this: The software is originally supplied with a sum of money in its "buffer". In use the software deducts each SMS cost from this money. When the software starts to run out of this pre-paid money the user pays money into my bank account, tells me how much, and then opens a window which shows him a string of alphanumeric characters which he quotes to me. This string is calculated from his current money balance, serial number and date of expiry, all stored as LONGs. I type these into a small program which allows me to add the amount he deposited to his credit and specify a new expiry date. I quote back to him the newly calculated sequence and he types this in. Decoded, this updates his credit and expiry date. I normally include the current date in the encrypted data, to make the code expire automatically so it cannot be applied more than once.

Over the years I have developed a sequence of alphabetic characters and numbers that can be read by humans with a minimum of verbal confusion. This consists of the 10 numbers 0-9, and 21 letters. The five letters left out (with the potentially confused character in brackets) are O(0) I(1) U(V) Q(0) and G(C). The total comes to 31 characters, which allows me to have a base31 number (which is also a prime number, a helpful feature, as I'll explain later).

In the "accept" code of the program's entry field, I automatically make the obvious substitutions, 0 for O, 1 for I, etc. so the users and I don't have to worry about what we are reading out. (One could extend the "base" with some of the symbol characters, such as @ and &, but a lot of my users don't know the names of the symbols (like ~, which is *tilde*) so I have found it safer not to use them.

### How to do a general base conversion.

The steps in a base conversion are:

- Take a digit number that is higher than the highest expected number of digits, and divide by the Divisor = Base \* (No-of-digits - 1). (If the first division yields a number greater than the base, you

- have too few digits and must increase the start digit number. I haven't done this in the example)
- Take the integer of the result.
  - If the integer is 0, reduce the number of digits by one and try again.
  - When the first non-zero result does come in, set a flag to remember there is a result (in case of a subsequent zero).
  - Put this "number" into the corresponding string-position and reduce the digit number.
  - Calculate (Remainder \* Divisor) to find what is left
  - Repeat until last digit

An example converting base10 to base 3...

A 4-digit base3 number has the following structure:

$$n_4 \ n_3 \ n_2 \ n_1$$

which means it has the value:

$$n_4 * 3^3 + n_3 * 3^2 + n_2 * 3^1 + n_1 * 3^0$$

which is

$$n_4 * 27 + n_3 * 9 + n_2 * 3 + n_1 * 1$$

( note that it goes from  $3^3$  to  $3^0$ , *not*  $3^4$  to  $3^1$ , since numbers are zero-based)

A real example:

Start with a decimal (base 10) value of 52 and convert it to base 3 (I have reduced the shown "precision" for the sake of clarity, but the application will calculate to 16+ significant figures which is generally sufficient.)

#### 4<sup>th</sup> digit

```
3 ^ 3 = 27          ! note the " ^_3 " for the 4th digit
52 / 27 = 1.925925
INT(1.925925) = 1   ! this "1" is immediately used to
                    ! return the string character - see below
0.925925 * 27 = 25
```

#### 3<sup>rd</sup> digit (n<sub>3</sub>)

```
3 ^ 2 = 9
25 / 9 = 2.777778
INT(2.777778) = 2
0.77778 * 9 = 7
```

#### 2<sup>nd</sup> digit (n<sub>2</sub>)

```
3 ^ 1 = 3
7 / 3 = 2.33333
```

$$\text{INT}(2.33333) = 2$$

$$0.33333 * 3 = 1$$

**1<sup>st</sup> digit ( $n_1$ )**

$$8 \wedge 0 = 1$$

$$1 / 1 = 1$$

$$\text{Thus } 52_{10} = 1221_3$$

To convert from a number to a corresponding character (which is not necessary in the case of base3, but is necessary in any case of base11 and above) I use the position in a string of the available "digits" in the number system.

Take the hex string (16 characters long) as an example:

```
DigitStr = '0123456789ABCDEF'
```

In the case of hex, the result of the INT( ) step (TmpShort in the code) is always a short of 0 – 15. So to find the alpha character corresponding to TmpShort I use string-slicing, thus:

```
TmpChr = DigitStr[ TmpShort + 1 ]
```

(The + 1 is because the first character which corresponds to TmpShort=0 is character No.1 in DigitStr.)

And here is where the "encrypting" comes in. If you scramble DigitStr but use the same string on both sides, it becomes a little difficult to unscramble from the given user-seen characters. And if you add a checksum (see later) which has to be numerically correct before the characters are accepted, it becomes even more difficult for a normal user to defeat the encryption.

The code that converts from base10 to any other base looks like this:

```
LongToStr          PROCEDURE (pLong, pLen)
ResStr             STRING(20)  ! the result string - allow lots of space
Cntr               SHORT      ! loop counter
Divisor           REAL        ! what we are going to divide by at Cntr
TmpReal           REAL        ! intermediate results
TmpLong           ULONG       ! use this LONG to work on
TmpShort          SHORT       ! the temp integer value of that position
PosOut            SHORT       ! which character we are working on in ResStr
LenRes            SHORT       ! Length of ResStr
DigitStr          STRING(50)  ! The encoding string
Base              SHORT       ! base of the number (LEN(DigitStr))
```

```
CODE
! LongToString CODE
! prototype (LONG pLong, ), STRING
! where pLong is the LONG to convert to the number-base
! pLen is an optional LEN parameter if you want zero-padding
! passed pLong
! convert a LONG into a baseN No. and return the string
```

```

! This is normally declared more globally than this to be available
! in one common place for the decode as well as the encode
! - see the example
DigitStr = '0123456789ABCDEFHJKLMNPRSTVWXYZ'      ! base 31
! the "base of the number is the length of DigitStr
Base      = LEN(CLIP(DigitStr))
! this allows you to put '0's as prefix to the number
! The "0" is the first character of DigitStr
IF pLen > 0
    ResStr = ALL(DigitStr[1], pLen)
ELSE
    ResStr = ''
END
! the value of Cntr may need increasing if your base is very short
! e.g. base 8 because the first division may result in more
! than expected. A programmers warning may be needed here to
! prevent supprises.
Cntr      = 8      ! this is the number of times it divides by the base
TmpLong   = pLong ! have to modify TmpLong
PosOut    = 1      ! was 0
LOOP
    ! start from the most significant
    Cntr -= 1
    !are we done?
    IF Cntr < 0 THEN BREAK.
    ! Divisor is the "value" of the "Cntr" position in the number
    Divisor = Base ^ Cntr
    ! This is the "value" of Cntr position including the rest of the number
    ! as decimal places
    TmpReal = TmpLong / Divisor
    ! the actual "value" of this position
    TmpShort = INT(TmpReal)
    ! are we still looking for the starting result?
    IF PosOut = 1 AND TmpShort = 0 THEN CYCLE.
    ! We always start at position 2 to signal that we HAVE started
    PosOut += 1
    ! if TmpShort is 0 it must return the 1st DigitStr, so it is TmpShort+1
    ResStr[PosOut] = DigitStr[TmpShort+1]
    ! put only the "rest" of the value back into TmpLong
    TmpLong = ROUND((TmpReal - TmpShort) * Divisor, 1)
END

! get rid of "extra" zero-digits in ResStr
IF pLen > 0
    LenRes = LEN(CLIP(ResStr))
    ResStr = ResStr[LenRes - pLen + 1 : LenRes]
END
RETURN(CLIP(LEFT(ResStr)))

```

Getting back to the LONG from Base31 is much simpler. You calculate the value of each position times the number in there and add it all together.

```
! StrToLong DATA
```

```

tStr      STRING(50)      ! Temp string to work from
Base      SHORT          ! base of the number
LenTstr   SHORT          ! Length of tStr
tPos      SHORT          ! Position in tStr
Expnt     SHORT          ! exponent at this position
DigitStr  STRING(50)     ! The encoding string
RetVal    LONG           ! the resultant LONG
CODE

! prototype (STRING pStr), LONG

! it is up to you to make sure the LONG does not overflow
! ( you could use a DECIMAL or a REAL perhaps? )

! This is normally declared more globally than this to be
! available in one common place for the decode as well
! as the encode - see the example
DigitStr = '0123456789ABCDEFHJKLMNOPSTVWXYZ' ! base 31
! the number base
Base      = LEN(CLIP(DigitStr))
! put the passed string into a local string and make
! sure it is LEFT
tStr      = LEFT(pStr)
LenTstr   = LEN(CLIP(tStr))

tPos      = 0 ! pre-incremented in the LOOP
RetVal    = 0 ! result starts at 0
LOOP
  tPos += 1
  IF tPos > LenTstr THEN BREAK.
  ! this is the "multiplier" of the value of that position
  ! note the -1 to make it zero-based - the position in
  ! DigitStr (-1) is the "value"
  Expnt = INSTRING(tStr[tPos], DigitStr, 1) - 1
  ! (Base^(LenTstr-tPos)) is the value of that position.
  ! (Remember the 1st character in the string is the
  ! MOST SIGNIFICANT DIGIT in the string - numbers are
  ! "counted" from the right)
  RetVal += Expnt * (Base^(LenTstr-tPos))
END

RETURN(RetVal)

```

If you need more than one LONG to store your data, convert each one separately and concatenate the result. This is the real significance of pLen in the LongToStString function; it fixes the size of the output string, padding it with zeros.

## Checksums

If you need to validate a number after transmission through an imperfect medium (like voice), the most economical technique is to use some calculation to produce one or two characters that can be sent with the original message. Then the receiving end can do the same calculation and see if it gets the same character(s). If not, a re-transmission



can be requested.

There are many different checksum/hash algorithms, all aimed at guaranteeing to find errors with near-zero chance of compensating errors. CRC16 is one standard for this and is very good, but unnecessarily complicated for this purpose. I use a literal checksum – i.e. I add up the ASCII character values of all the characters and find the remainder after division by the base. (This is where a base of a prime number helps to improve the checksum. A non-prime number might return the same result if one of its factors was used instead of the complete number, making it more likely to get the same checksum for any given alteration of the code)

The checksum will have a value between 0 and base-1. Adding 1 to the checksum allows a character to be identified in `DigitStr`, which can be concatenated to the original string. This is not a very solid checksum but is easy to implement and understand and is adequate for this simple purpose. (See the example `ChkSum` function in the downloadable source.)

## Conclusion

Changing the base of a number can be quite confusing until you suddenly "see" how it works. Remember that the *position* in a string of "numbers" determines the "value" of that position and the *digit* *in* that position fixes how many of that "value". So 315 as a decimal number contains 3 "Hundreds", 1 "ten" and 5 "one" units; from these facts I developed the above functions.

Base conversion using a string allows conversion from decimal into any other base and back again. This makes it very useful in lots of places like converting from decimal to hex or octal. If you scramble the base-string, you have elementary, but useful, encryption. This is mainly useful for things like registration-codes or the "sending" of money in rented software that the user must not be able to easily reproduce. It is not a strong encryption method, but good enough to deter most normal users.

[Download the source](#)

---

[Dermot Herron](#) grew up and went to school in Rhodesia, and earned an electrical-engineering degree at Capetown University. In the 1970s he worked for the Rhodesian Post Office, which was then also the telephone company. He was given the use of an HP9100, which was the very first programmable desktop computer, to help with the design of party-line telephones. He totally fell in love with computers then and there. Leaving the Post Office, he traveled for four years, worked in Canada (as a cabinet maker) and in England (as a microprocessor engineer) and then immigrated to New Zealand. But Dermot learned that once Africa gets into your blood you are doomed! Now he runs his own company in Johannesburg, writing software to send messages to mobile phones. He lives on a 19-acre plot with a river, and spends his off-time fixing things.

## Reader Comments

[Add a comment](#)

- [» Very interesting use of base encoding. In the...](#)
- [» Thanks for that - It was the one bit missing that would...](#)

- [» Thanks for that - It was the one bit missing that would...](#)
- [» Back in prehistoric days \(or at least pre-Clarion\), I came...](#)
- [» Or in other words how do you solve the problem  \$B^Y=X\$  for...](#)

# Clarion Magazine

## Clarion News

[Search the news archive](#)

### [CapeSoft's Clarion Version Support](#)

CapeSoft's general policy on supported Clarion versions for our Accessories has always been more or less the last 3 major versions of Clarion. In other words, before Clarion 6, CapeSoft supported Clarion 4, Clarion 5, and Clarion 5.5. Since Clarion 6 releases, there have been many patch releases, and point releases, and many programmers have found the version that suits their programs best, and not necessarily upgraded to the latest version of C6. With these new versions, the runtime library has changed fairly frequently as well, requiring DLL based products to be recompiled for them. To make this process easier, CapeSoft now ships one install file containing builds for Clarion 5, Clarion 5.5, and the latest version of Clarion 6 (currently 6.3 9050). If you are using a previous version of Clarion 6 however, and using a DLL based product, then you simply download another patch install for the relevant build.

Posted Thursday, March 09, 2006

### [CapeSoft Email Server Runs As Service](#)

CapeSoft Email Server is a compact yet full featured Email Server for small business and home users. It features free upgrades and unlimited mailboxes. Now CapeSoft Email Server runs as a fully compatible Windows Service. Right now the price is \$30; as of March 15, 2006 the price will be \$39. Free 60 day trial. Source code available.

Posted Thursday, March 09, 2006

### [NetTalk 4 Adds SSL, Web Server](#)

NetTalk 4 includes 2 major new features, namely Secure Socket Layer (SSL), and full

featured, completely template driven, customisable Web Server functionality. One of the primary purposes behind the NetTalk 4 Web technology is being able to utilize both your existing Clarion skills, and your existing Clarion code. By building a web server component into your existing application, your web app can call all your existing processing functions live, just as your win32 app would. The template comes packed with a choice of Stylesheets, which you can edit to your hearts content, or use your own CSS file. The template automatically generates browses, forms, lookups, menus, tabs, and even graphs (Insight Graphing required). The interface generated utilizes today's latest AJAX and Web 2.0 technologies offering you the smoothest most responsive web interface possible. During the beta you can get the NetTalk 4 upgrade for \$149. If you bought NetTalk after Feb 1, 2005, the upgrade is free. (Please place "Free Upgrade" in the comments field, and select the Cheque payment option on the order form. CapeSoft will then verify whether you qualify, and send you the registration details.) New licenses are \$399.

Posted Thursday, March 09, 2006

### [VuDeploy 3.0](#)

vuDeploy version 3.0 is available for download. This release is a complete redesign, with a new graphical user interface. New features include: Add, Edit, and Delete projects and display the projects in a list; Select files for inclusion, from anywhere and display them in a list; Automatically run multiple files (EXEs can be run either consecutively or concurrently); Change the order of the files (for consecutive execution); Automatically detect and add (via user confirmation) any additional files found in the extracted folder and roll them back up into the EXE; Automatically detect and delete from the EXE (via user confirmation) any files that were deleted while the files are extracted; Minimize all other programs during execution and automatically restore the file status on completion. vuDeploy 3.0 is offered as a free upgrade to all current users.

Posted Thursday, March 09, 2006

### [Beyond Compare 2.4](#)

Scooter Software has released Beyond Compare version 2.4. Although not a Clarion third party product, BC is highly recommended by a number of Clarion developers. Version 2.4 adds: Support for checking in and out of most version control systems; Support for Explorer context menu in folder viewer; Support for ignoring file extensions when comparing folders; New FTP library under Windows NT/2000/XP; Explorer menu add-on now works under Windows XP/x64; Numerous fixes and tweaks.

Posted Thursday, March 09, 2006

### **[GoToLunch Batch Compiler Supports Command Line Parameters](#)**

The GTL62 batch compiler, available for download from the PAR2 Download Center, supports command line parameters. For any .APP tagged on the "Applications" tab, MAKE generates and compiles, TXA exports .APPs to TXA, and DCT exports to TXD. This allows running GTL62 on a scheduler (with the appropriate parameter) for unattended backups (if Clarion is not running, GTL will start it).

Posted Thursday, March 09, 2006

### **[Australian DevCon Visa Information](#)**

There may be some people out there who want to come to the Australian DevCon in April, who have not yet registered, and who live outside of Australia. If you live in a country that is not eligible for a simple online visa (ETA), then you need to make a decision very soon, as visas can take a while to process. The Australian DevCon is now listed as an international event with the immigration department, so the conference organizers can help you fast track your visa application.

Posted Thursday, March 09, 2006

### **[Belarc Advisor Does Free PC Audit](#)**

Greg Berthume likes this free reporting tool that tells you what's on your PC.

Posted Thursday, March 09, 2006

### **[PrintWindow Build 111](#)**

PrintWindow Build 111 is now available. Changes include: Fix to printing problems when excluding tabs; Fix to shadows problem when setting vertical centered alignment for the report; Fix to shadows not printing if the entry field was empty; Fix to text box printing at incorrect height.

Posted Thursday, March 09, 2006

### **[RADFusion's New Look](#)**

The RADFusion site has been upgraded with a new look.

Posted Thursday, March 09, 2006

### **[Huenueleufu Back Online](#)**

After a somewhat traumatic move to Bariloche (how can moving to Patagonia be traumatic?), Huenuelufu is once again connected to the Internet (now wireless) and working at full capacity.

Posted Thursday, March 09, 2006

### **[NetTalk 4 Web Server Demo](#)**

A NetTalk 4 web server demo is now available online. Log in using demo / demo. Check out the two browse styles, and the graph under the config menu.

Posted Thursday, March 09, 2006

### **[Dan Pressnell's Better SQL, Better OOP Source](#)**

Dan Pressnell's source code for his Better SQL and Better OOP series are now available at IceTips.

Posted Monday, March 06, 2006

### **[SoftVelocity News Server Moving, DNS Changes May Take Time](#)**

The SoftVelocity news server is being moved onto a new T1 circuit on March 4. The move means the server is getting a new IP address. The assignment of the new IP address means that until DNS entry propogates to the DNS server you are using, your news client will not be able to "see" the server (resolve the IP address.) Most DNS propogation depends on the ISP and how often they update their DNS servers. Some are really slow and take several days even though the correct domain information will usually show up in WHOIS within 24 hours. So depending on your DNS server you may immediately see the server, or it may take some time. Most of the registrars state 24-48 hours, a few update in 12, then there are those that can take as long as 3-5 days.

Posted Sunday, March 05, 2006

### **[EasyResizeAndSplit 2.12](#)**

EasyResizeAndSplit 2.12 is now available. This release fixes compatibility issues with CapeSoft's AnyFont and Power XP Theme.

Posted Wednesday, March 01, 2006

### **[RADFusion Site Under Maintenance](#)**

The RADFusion site is undergoing a make-over in many areas. While this rebuilding is in

progress, the site will be unavailable except for the home page which states what is currently happening (including products which may still be ordered).

Posted Wednesday, March 01, 2006

### **[Dictionary Assistant 2.11](#)**

Dictionary Assistant 2.11 is free of charge to all registered customers of Dictionary Assistant 2. The significant enhancement in this release is the new Dictionary Assistant Viewer, a redistributable component for viewing and printing your Dictionary Assistant database.

Posted Wednesday, March 01, 2006

### **[Clarion ProScan](#)**

Charles Edmonds ( LANSRAD ) has opened the Clarion ProSeries website and has released Clarion ProScan, the first of a series of new Clarion Development tools. Building on ImageEx technology and Clarion 6, ProScan is a single or multipage document scanning solution that can be added to your application in less than 10 minutes. Adding ProScan to your program involves adding a global template, importing a TXA, and dropping the ProScan link button on your parent form. ProScan also comes with a 5600+ word help file that was written for you to distribute to your end users (or integrate into your own help system). Demo available. Clarion ProScan normally sells for \$199.95 but until March 1, 2006 you can buy it for \$159.95. There is also a bundle option to get Clarion ProScan along with ProImage (a photo processing tool, coming soon) at a combined introductory price of \$299.95. All ProSeries products come with a 30 day no-questions asked refund policy. Clarion ProScan and ProImage are source products - no black box DLLs.

Posted Wednesday, March 01, 2006

### **[NetTalk Web Server](#)**

CapeSoft's new NetTalk Web Server integrates a number of readily-available scripts into its own web server framework. Web Server includes Ajax support. It's still early days for this product - look for more news out of CapeSoft as development proceeds.

Posted Wednesday, March 01, 2006

### **[Australian DevCon Early Bird Special Ends Feb 28](#)**

The Early Bird special rates for the Australian DevCon end on February 28th. Sign up

now!

Posted Monday, February 27, 2006

### **[Evolution Report Export 1.09](#)**

Evolution Report Export 1.09 exports to Excel, HTML, ASCII, CSV, Report and XML from the standard Clarion Report Preview. Clarion 5, Clarion 5.5, Clarion 6. ABC or Clarion (Legacy) Templates.

Posted Friday, February 24, 2006

### **[Evolution Edit in Form 1.04](#)**

The Evolution Edit in Form 1.04 template transforms your standard Forms into an Edit in Place. You can use the form, as always, and the template resizes the form window on top of the browse row, simulating the Edit in Place behaviour. You can use lookups, dropdowns, calendars, tooltips, etc. and all the form embed points. For Clarion 5.5 and Clarion 6.x. Both ABC and Legacy.

Posted Friday, February 24, 2006

### **[Clarion FreeImage Project 3.8.0.3](#)**

The Clarion FreeImage Project 3.8.0.3 is now available for download. This update includes an installer built with SetupBuilder. New examples show how to use the File Dialog and Clipboard classes without the FreeImage class or Image control. The Clarion FreeImage Project is an open source project built around the FreeImage raster processing library.

Posted Friday, February 24, 2006

### **[18,960 Ace Icons For \\$49.95](#)**

You now get 18,960 icons in either the Ace Brights or Ace Lights icon sets, for \$49.95 each. In the Roma set you get 7020 icons. Both the Ace and Roma icon sets are being expanded; the Roma set will soon have over 10,000 icons. Anyone who purchases (or has purchased in 2006) the Ace or Roma icon sets will receive the new images for free, when they are released.

Posted Friday, February 24, 2006

### **[EasyExcel 4.00](#)**



EasyExcel 4.00 is now available. New methods include: SetPageHeaderFont - sets page header font; SetPageFooterFont - sets page footer font; FreezePanels - freezes the split panes. Changed methods include: GetSelection - added new parameter and method now returns errorcode; List2Excel - method was re-written to support the date and time columns and was fixed a bug of incorrect columns format setting. Changed templates include: PageSetup code template - now includes the ability to change page header/footer font. The upgrades from 1.0 to 2.0/3.0 and from 2.0/3.0 to 4.0 are available at ClarionShop for only US\$39. Registered users that purchased version 2.0/3.0 after September 1st, 2005, will receive this upgrade free of charge.

Posted Friday, February 24, 2006

### **[xPowerManager 1.02 Freeware](#)**

xPowerManager 1.02 is a free tool to automatically shutdown your PC on a timer. You can use a countdown timer or an alarm clock timer.

Posted Friday, February 24, 2006

### **[xNotes 1.7](#)**

xNotes 1.7 is now available, and is compatible with PowerXP-Theme 2. xNotes supports SingleExe, MultiDll (Local Mode, Standalone Mode), 32-bit. Compatible with Clarion 6.x (6.3, 6.2, 6.1), Clarion 5.5 and Clarion 5.

Posted Friday, February 24, 2006

### **[xAppWallpaper Template 1.9](#)**

xAppWallpaper Template 1.9 is now available. New in this version: Modification in Template for setting xAppWallpaper items in frame menu; You can set/change font, color, height, width, offset etc; Compatible with Clarion 6.x (6.3, 6.2, 6.1); New installation kits prepared with SetupBuilder 5.3; updated Demonstration programs and Installation Kit for Clarion 6.x, Clarion 5.5 and Clarion 5. xAppWallpaper Template is \$19.

Posted Friday, February 24, 2006

### **[SetupBuilder 5.3 Build 1414](#)**

SetupBuilder 5.3 Build 1414 is now available. As always you may use the "Check for Updates" option within SetupBuilder to auto-update your current SB5 version.

Posted Friday, February 24, 2006

## **Insight Graphing Review**

A short review of Capesoft's Insight Graphing, by Ennis Baggott, is now available on [DeveloperReviewed.com](http://DeveloperReviewed.com).

Posted Friday, February 24, 2006

# Clarion Magazine

## First Look: CapeSoft Profiler

by Dave Harms

Published 2006-02-27

[CapeSoft's Profiler](#) is just what you'd expect – a tool that examines your running application to determine how much time is being taken by each block of code. What is particularly fascinating about Profiler, at least to me, is the fine granularity that's available. I'm accustomed to using Clarion's built-in [profiler hooks](#) to get a procedure-level look at what my code is doing; CapeSoft's Profiler takes a different, and much more powerful, approach; it runs your application as a child process, and makes use of the application's own debug information to discover what's happening and how long it takes to execute each line of code.

### Installing

As with other CapeSoft products, the [Profiler download](#) is wrapped inside a CapeSoft [SafeReader](#) file. SAF files are created with [SafeWriter](#), which encrypts files using either the 168-bit Triple DES or the variable key-length Blowfish algorithm. You can buy SafeWriter for \$39; SafeReader is a free download.

When I opened the Profiler SAF file, SafeReader asked if I wanted to install an update (from 2.18 to 2.19 – Profiler requires at least 2.14). I agreed, and the update process was painless enough, except that the install program told me I had to restart my computer for the update to take effect. I decided to do so later, since I was well beyond the required 2.14 version, but after the Profiler install I ran SafeReader to check the version number. It was 2.19. Ah well, no harm done.

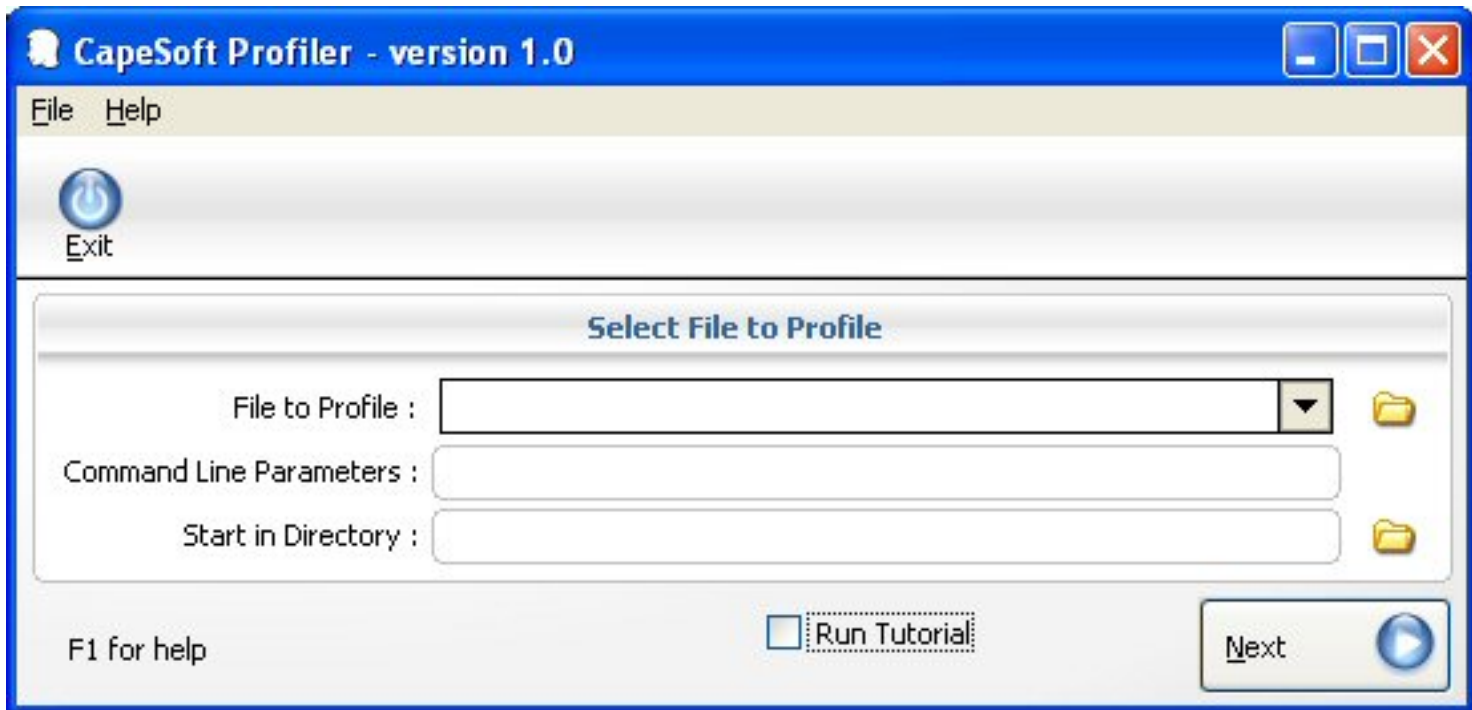
In any case I pasted my unlock key into SafeReader and ran the Profiler.exe install program. I was sternly warned part way through to shut down Clarion, which I did. And when the install program completed, it told me where I might find Profiler. But I closed that window out of habit, before I actually got a chance to read what it said. So I didn't know where to find Profiler, and for some reason I couldn't see it in my Start Menu (Bruce Johnson assures me it should be there, but I have since built a new machine and didn't have to re-run the installer, so I still don't have the Start Menu entry). No matter - I reviewed the [online documentation](#) and discovered that Profiler was installed in the Clarion IDE, under the Accessories menu. That seems pretty obvious now.

## Running the profiler

Before you run the profiler you'll need to ensure that the application you want to profile has been compiled with Debug Information set to Full. Without this information, the Profiler has no way to discover what your application is doing.

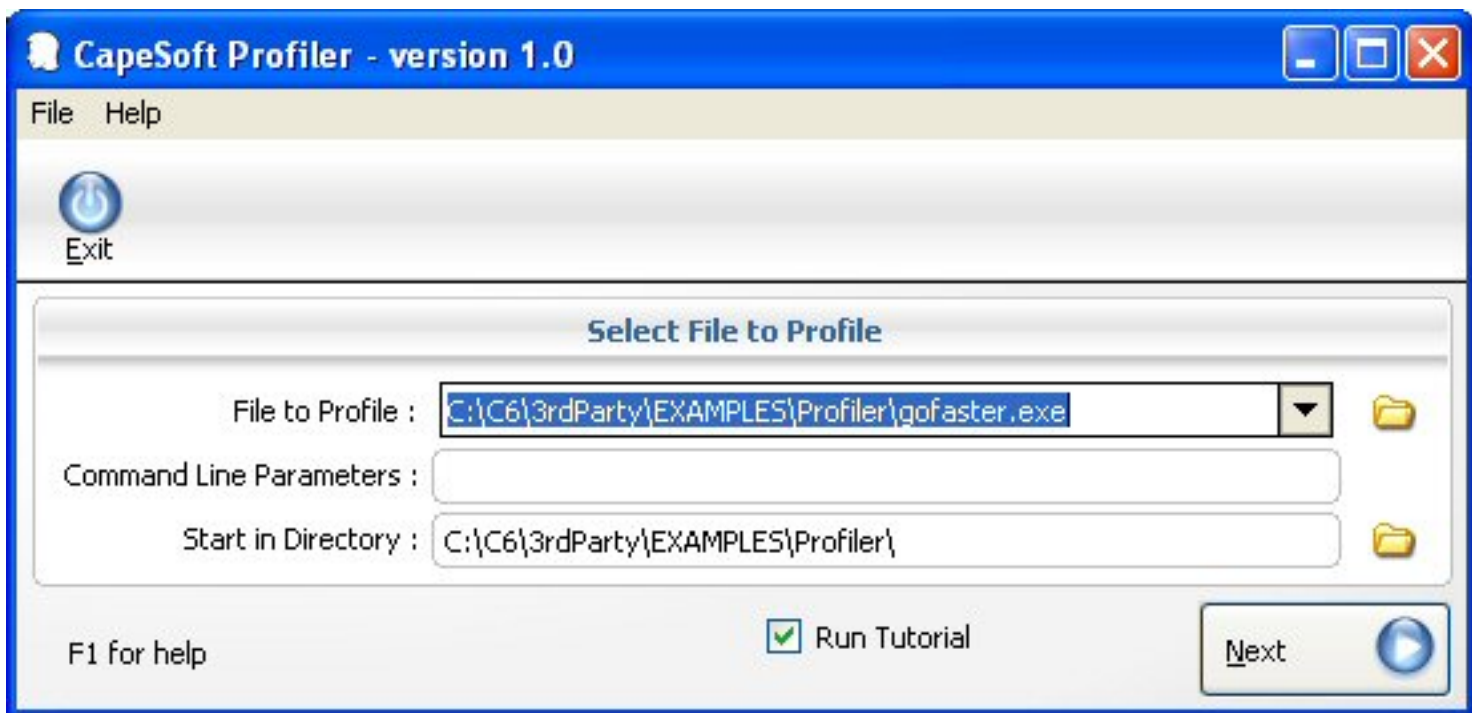
The following screen shots show the Profiler running the included GoFaster example application. You'll need to compile this application first, and you can find it in the 3rdParty\Examples\Profiler directory. The project already has the required debug options set.

When you run the Profiler from the Accessories menu, you'll see the window in Figure 1. Click on the Run Tutorial checkbox to bring up a series of explanatory tutorial windows which will accompany you through the tutorial. Since I'll be covering pretty much the same territory (with, I hope, a few useful additions), I won't show those screenshots here.



**Figure 1. Getting started**

The first thing you need to do is fill in the File to Profile field, as shown in Figure 2. The Start in Directory field is optional – if you leave this blank, Profiler will use the profiled application's directory as the default.



**Figure 2. Choosing a file to profile**

Click on Next to see the window shown in Figure 3, which asks you to specify your

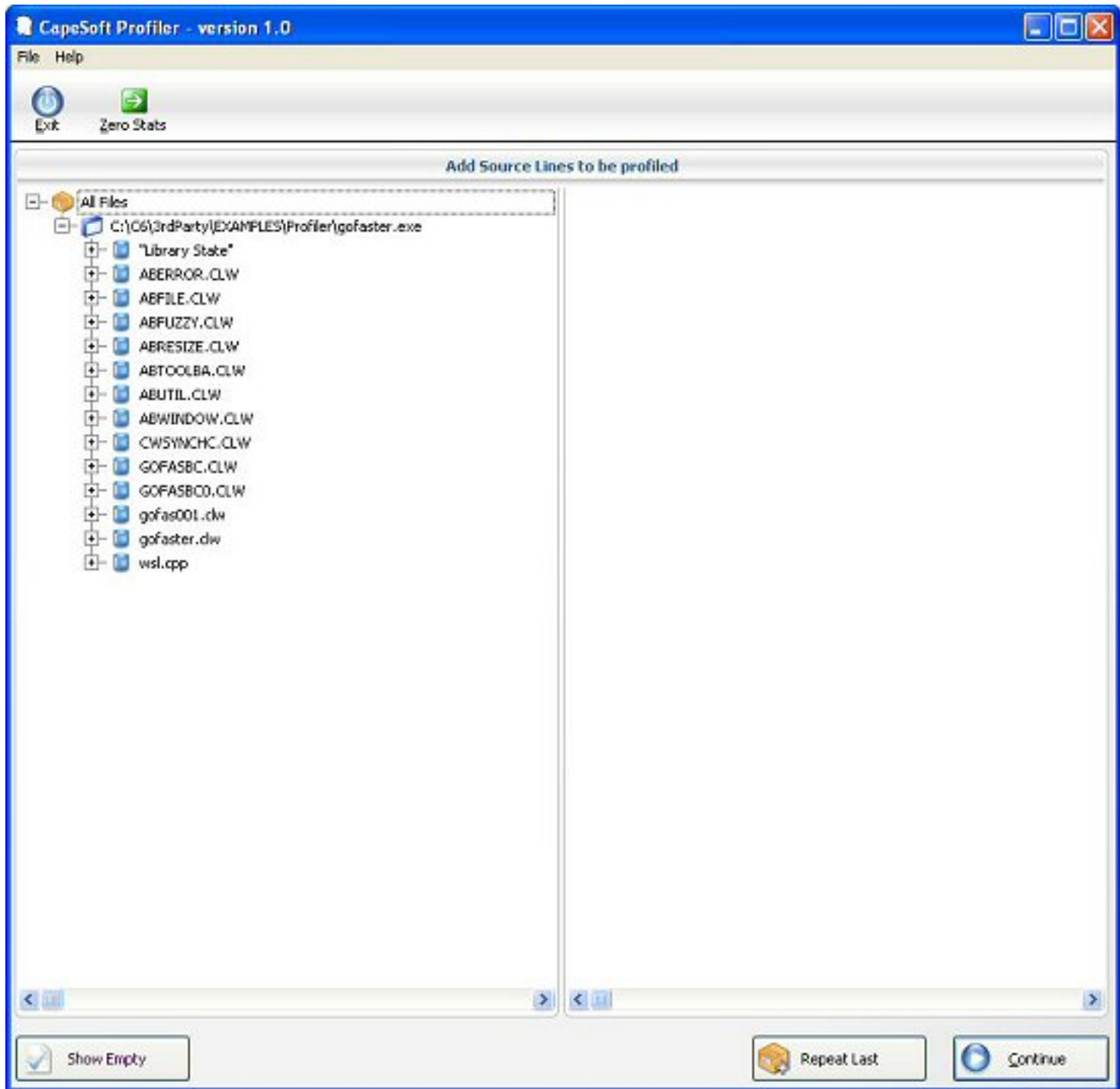
Clarion install directory. Profiler will use this information to locate all of the Clarion source code (ABC classes, etc), so you can, if you wish, profile SoftVelocity's code as well as your own.



**Figure 3. Specifying the Clarion directory**

You only need to fill in the Extra Source field if, for instance, you are profiling an EXE with a DLL, and the DLL's source is in a different directory than the EXE *and* you need to profile some code in the DLL.

Click on Load, and you'll see a two-paned selection window (Figure 4).



**Figure 4. Choosing which code to profile ([view full size image](#)).**

I didn't immediately grasp all of the options available in the Add Source Lines window. When the window first appears, there are just three buttons visible at the bottom of the window: Show Empty, Repeat Last, and Continue. Actually it's quite possible that Repeat Last only appears after you've used the profiler at least once. But click on one of the lines in the left-hand list box, and a fourth button will appear. If you select All Files, this new button is called Add All Files; select the EXE in the list, and the button becomes Add All Modules; select a source file and the button becomes Add All Procedures (that includes

class methods and routines).

Expand one of the source modules, as shown in Figure 5, and the button changes again to Add Lines. In all cases, once you've added some code, the relevant item in the left hand pane (application, module, procedure) appears in bold type. You can also add items by right-clicking and choosing Add [All Files | this file | this module | this procedure] from the context menu.

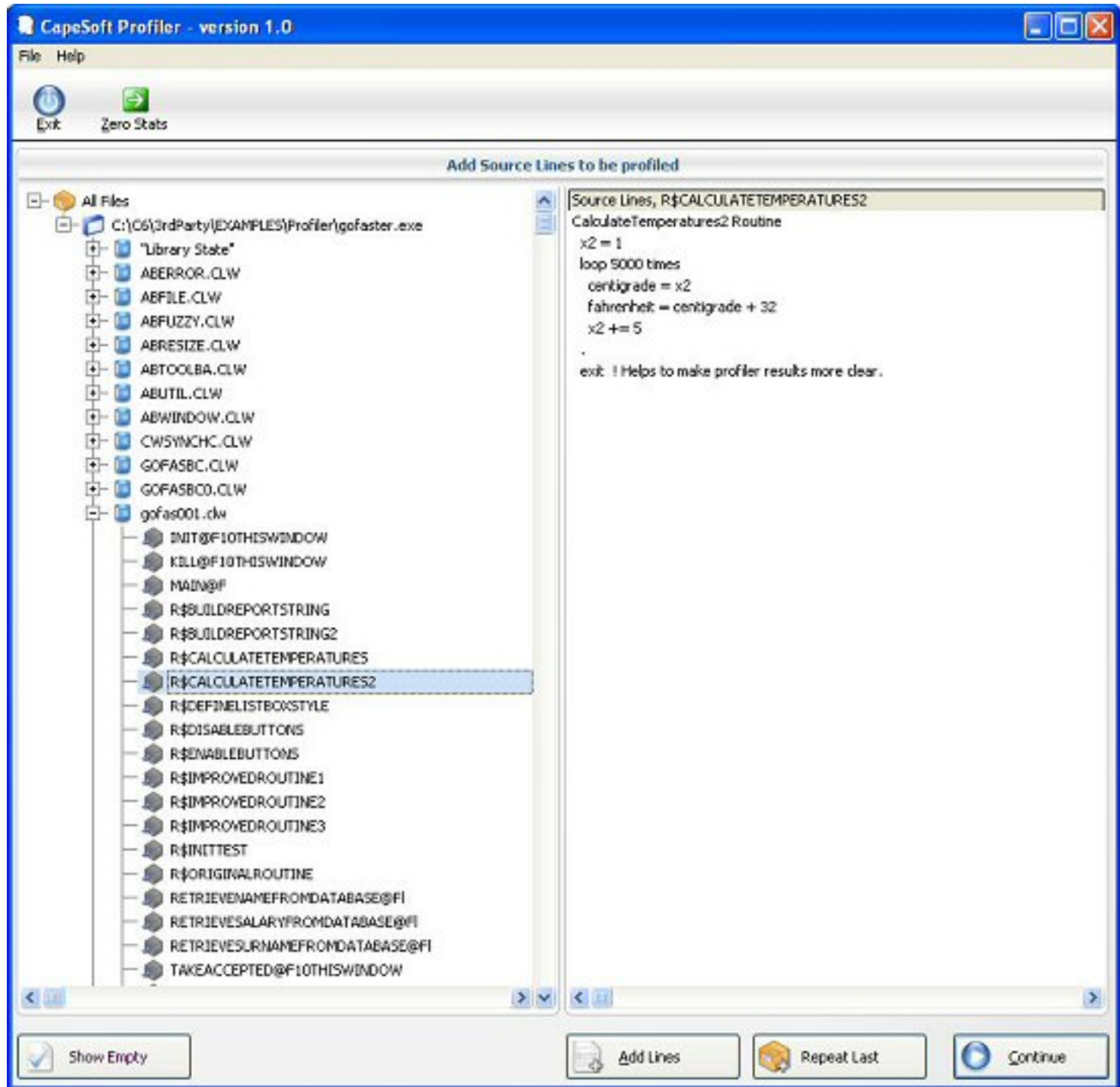


Figure 5. Adding procedure/method lines ([view full size image](#))



Once an item is selected you can remove it, but only via the context menu (as far as I can tell). And, importantly, you can return to this window from the running profiler at any time. I quite like the ability to change the files being profiled without having to restart Profiler.

**NOTE:** The second time I used Profiler I thought to myself that it would be cool if I could just repeat the last settings I'd used. And, since CapeSoft employs psychic developers, there is that Repeat Last button at the bottom of the window. This is very handy, and the only suggestion I have is it might be nice to make this into a list so I could save multiple session settings.

For starters, choose gofas001.clw, and add OriginalRoutine, and ImprovedRoutine1, 2 and 3. Click on Continue. You'll see the main Profiler window, as shown in Figure 6.

The screenshot shows the main window of CapeSoft Profiler. At the top, there is a menu bar with 'File' and 'Help'. Below the menu bar is a toolbar with icons for 'Exit', 'Zero Stats', 'Resume', 'Trace On', and 'View'. The main area is divided into two sections. The upper section contains a table with the following data:

Index	Calls	Total Time ms		CPU Clock Cycles X 100		Source Line
		Approximate	Total	Total	Average	
42312	50	2,057	20,668,245	413,365		R\$IMPROVEDROUTINE3
34162	50	8,155	81,945,259	1,638,905		R\$IMPROVEDROUTINE2
26012	50	15,918	159,961,684	3,199,234		R\$IMPROVEDROUTINE1
17862	1,000	21,527	216,326,397	216,326		R\$ORIGINALROUTINE

The lower section shows the details for the selected routine, R\$IMPROVEDROUTINE3. It includes the following information:

- Image : C:\C6\3rdParty\EXAMPLES\Profiler\gofaster.exe
- Module : gofas001.clw
- Procedure : R\$IMPROVEDROUTINE3

Below this information is a list of source lines for the routine:

```

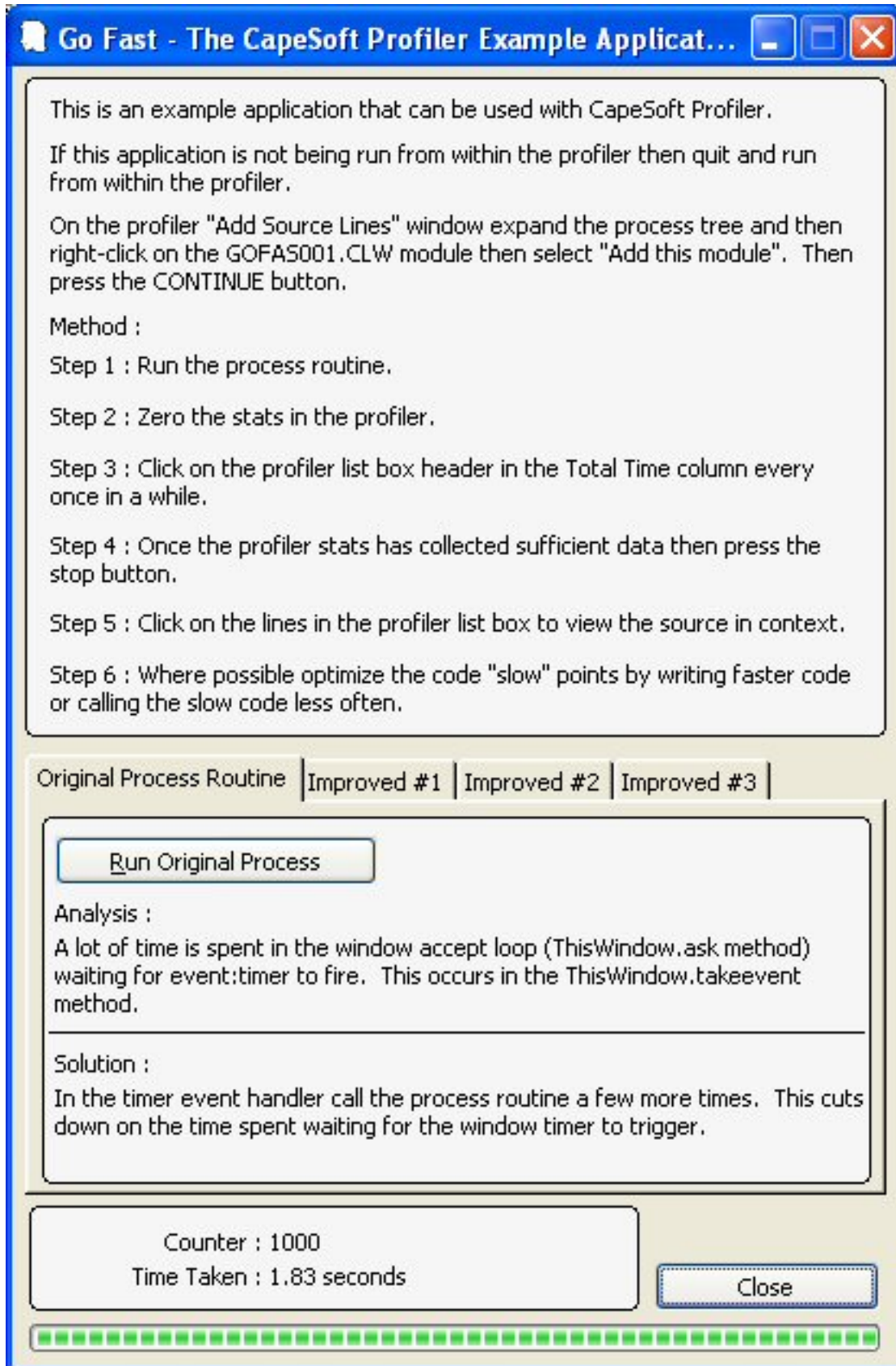
Line# Source Line
238 ImprovedRoutine3 Routine
239 loop 20 times
240 Counter += 1
241 progress1 = Counter / 10
242 display(?progress1)
243 display(?Counter)
244 do BuildReportString2
245 do CalculateTemperatures2
246 x = 0
247 end
248 exit ! Helps to make profiler results more clear.

```

At the bottom left of the window, there is a status bar showing 'Time Taken : 0:00:02.687' and 'Process Time : 0:00:47.657'. A button labeled 'Back to Add Lines' is also visible.

Figure 6. The main Profiler window ([view full size image](#))

Figure 7 shows the GoFaster application, which simply lets you run each routine in turn, so you can generate profiling data, locate bottlenecks, and see the results of specific code improvements.



## Figure 7. The GoFaster application

GoFaster demonstrates three speedup techniques which you may find useful: More routine calls per event timer (how many will depend on how long a particular task takes, which is also processor dependent and therefore potentially tricky); changing ULONG to LONG wherever possible (ULONGs are treated as DECIMALs internally); and using string slicing instead of CLIP on big strings.

Choosing which code to profile is half the battle; displaying the data the way you want it is the other half, and a big part of that has to do with those big toolbar buttons. Zero Stats lets you reset all counters; Pause|Resume starts/stops and stops the profiled application, which is handy if you want a snapshot of the profiling numbers; Trace On|Off toggles a highlight bar in source view (hang on a sec for an explanation of source view), meaning that as your code executes, Profiler will highlight the currently executing line (in most cases, code will be executing so fast that this will not be particularly helpful, but as Bruce pointed out to me, if you're stuck in an endless loop, or on a critical section, Trace can be very useful); View is a drop-down menu which lets you display execution times for each line, for each procedure/routine/method, for each module, or for each executable.

Finally, you can sort the profiling data by clicking on the various list box headers. That functionality isn't immediately obvious, as there are no active sort order indicators, at least in the beta. But if you click on the headers, you will get corresponding sort orders.

The key to getting Profiler to give you what you need is to use the right combination of View (line, procedure, module, executable) and the sort order. For instance, Figure 8 shows a View setting of procedure, and a sort order of Total Time ms.

The screenshot shows the CapeSoft Profiler interface. At the top, there are menu options (File, Help) and control buttons (Exit, Zero Stats, Pause, Trace ON, View). The main area is divided into two sections. The upper section is a table with the following data:

Index	Calls	Total Time ms		CPU Clock Cycles X 100		Source Line
		Approximate	Total	Total	Average	
42314	50	2,057	20,669,092	413,382		R\$IMPROVEDROUTINE3
34162	50	8,155	81,945,259	1,638,905		R\$IMPROVEDROUTINE2
26012	50	15,918	159,961,684	3,199,234		R\$IMPROVEDROUTINE1
17862	1,000	21,527	216,326,397	216,326		R\$ORIGINALROUTINE

The lower section shows the source code for the selected routine, R\$ORIGINALROUTINE. The code is as follows:

```

Image : C:\C6\3rdParty\EXAMPLES\Profiler\gofaster.exe
Module : gofas001.dwr
Procedure : R$ORIGINALROUTINE
Line# Source Line
208 OriginalRoutine Routine
209 Counter += 1
210 progress1 = Counter / 10
211 display(?progress1)
212 display(?Counter)
213 do BuildReportString
214 do CalculateTemperatures
215 exit ! Helps to make profiler results more clear.

```

At the bottom left, there are two buttons: "Time Taken : 0:00:11.406" and "Process Time : 0:00:47.657", and a "Back to Add Lines" button.

**Figure 8. Comparing routine execution times ([view full size image](#))**

What if I want to simply look at my source code, and see how much time is being taken by each line? Figure 9 uses the View|All Lines setting, and I've clicked on Source Line.

Index	Calls	Total Time ms	CPU Clock Cycles X 100				Line #	Source Line
			Approximate	Total	Longest	Shortest		
17855	1,000	1	12,672	1,581	7	13	208	OriginalRoutine Routine
17856	1,000	11	111,769	26,428	4	112	209	Counter += 1
17857	1,000	9	93,769	17,001	48	94	210	progress1 = Counter / 10
17858	1,000	68	682,273	6,238	359	682	211	display(?progress1)
17859	1,000	1,004	10,091,078	77,320	9,342	10,091	212	display(?Counter)
17860	1,000	3,005	30,197,471	82,731	28,951	30,197	213	do BuildReportString
17861	1,000	3,663	36,811,962	73,333	35,575	36,812	214	do CalculateTemperatures
17862	1,000	13,765	138,325,387	25,856,982	4,774	138,325	215	exit   Helps to make profiler results more clear.
25850	50	0	896	121	10	18	216	ImprovedRoutine1 Routine
25851	50	0	652	301	6	13	217	loop 20 times
26004	1,000	4	36,756	20,422	5	37	218	Counter += 1
26005	1,000	10	103,718	21,253	50	104	219	progress1 = Counter / 10
26006	1,000	129	1,294,136	78,985	535	1,294	220	display(?progress1)
26007	1,000	1,165	11,708,111	108,875	9,480	11,708	221	display(?Counter)
26008	1,000	4,289	43,097,919	86,494	29,106	43,096	222	do BuildReportString
26009	1,000	3,719	37,372,159	81,199	35,533	37,372	223	do CalculateTemperatures
26010	1,000	9	88,614	26,326	4	89	224	x = 0
26011	1,000	6	59,550	21,097	1	60	225	end
26012	50	6,588	66,199,177	56,482,413	82,891	1,323,984	226	exit   Helps to make profiler results more clear.
34000	50	2	19,666	18,275	9	393	227	ImprovedRoutine2 Routine
34001	50	0	324	9	5	6	228	loop 20 times
34154	1,000	2	21,363	5,360	5	21	229	Counter += 1
34155	1,000	10	101,679	836	64	102	230	progress1 = Counter / 10
34156	1,000	116	1,170,173	21,712	435	1,170	231	display(?progress1)
34157	1,000	1,157	11,621,997	127,290	9,355	11,622	232	display(?Counter)
34158	1,000	4,069	40,893,195	116,747	29,017	40,893	233	do BuildReportString
34159	1,000	107	1,074,399	21,236	933	1,074	234	do CalculateTemperatures2
34160	1,000	4	44,047	16,332	3	44	235	x = 0
34161	1,000	3	32,771	25,079	0	33	236	end
34162	50	2,683	26,965,630	17,124,939	76,270	539,313	237	exit   Helps to make profiler results more clear.
42150	50	0	589	23	8	12	238	ImprovedRoutine3 Routine
42151	50	0	786	503	4	16	239	loop 20 times
42304	1,000	3	32,073	24,617	3	32	240	Counter += 1
42305	1,000	4	41,188	1,814	30	41	241	progress1 = Counter / 10
42306	1,000	52	525,047	21,460	257	525	242	display(?progress1)

Time Taken : 0:00:16.250  
Process Time : 0:00:47.657

Image : C:\C6\3rdParty\EXAMPLES\Profiler\gofaster.exe  
Module : gofas001.dw  
Procedure : R\$ORIGINALROUTINE

Line#	Source Line
211	display(?progress1)
212	display(?Counter)

Figure 9. Line by line execution times ([view full size image](#))

As you can imagine, many combinations are possible. You can sort by the number of calls to a line, procedure, module, or executable, by total time, and by approximate time (which appears to just be the inverse of total time). CPU Clock Cycles X 100 appears to yield the same sort order as total time, but the subheadings let you sort by the total clock cycles (from most to least), by longest usage for that block, by shortest usage, and by average usage. Line number and source line appear to yield the same result, at least within one module.

There are a few things to note about Profiler. Depending on how much code you profile, the performance of your application may be significantly affected. This is to be expected, since the profiler introduces a fair bit of overhead.

There is a quirk in how execution time is allocated, when you are doing only partial code profiles. In particular, if code block A is not profiled, but it calls code block B which is profiled, all of the CPU time which occurs outside block B will be added to the end of block B, which means you may see some whopping big times associated with procedure RETURNS and routine EXITS (CapeSoft also recommends explicit RETURN and EXIT statements). You'll notice some lengthy EXIT times in Figure 9. But if you attempt to profile all of the code in `gofas001.clw`, in order to get more accurate EXIT times, you'll find that the routines run a lot slower because of the extra overhead. So it really is to your benefit to keep your profiling as narrow as possible. And don't get too hung up on the EXIT/RETURN times.

## Summary

Profiler is a powerful tool, and that means you need to think a little about how you want to use it to find performance bottlenecks. In other words, you can't just throw your entire application at Profiler and expect it to tell you where you've mucked up your code. There are hundreds, if not thousands, of method/procedure/routine calls in a typical Clarion app, and you can easily find yourself overwhelmed with data if you try to look at it all at once.

The beauty of Profiler is that it lets you choose the level at which you want to view profiling data; you can drill down from application level (in a multi-DLL app), to modules, to procedures (and methods and routines) until you find the ugly bits.

There are still a few things I'd like to see in Profiler, such as more complete documentation (which I'm sure will be there for the Gold release), and a list of common performance problems and their solutions. But Profiler already goes well beyond any previous Clarion profiling capability, and even in beta is an essential tool for anyone concerned with application performance.

Profiler is another terrific utility from the bright minds at CapeSoft. During the beta period it costs \$299; after Gold release, Profiler will be \$399.

For more information see the [Profiler product page](#) at CapeSoft.

[David Harms](#) is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of *Developing Clarion for Windows Applications*, published by SAMS (1995), and has written or co-written several Java books. David produces the [Planet Clarion](#) podcast, which he co-hosts with Andrew Guidroz II.

## Reader Comments

[Add a comment](#)

- [» This review reflects my experience with the product. I...](#)

# Clarion Magazine

## The ClarionMag Blog

Get automatic notification of new items! [RSS feeds](#) are available for:

**XML** [All blog entries](#)

**XML** [All new items, including blogs](#)

### Blog Categories

- » [All Blog Entries](#)
- » [Clarion 7 Clarion.NET](#)
- » [Future Articles](#)
- » [News flashes](#)
- » [Nifty Stuff](#)

## The SV blog lives!

[Direct link](#)

Posted Friday, March 10, 2006 by Dave Harms

The [SV blog](#) lives! Bob Foreman has [posted an item](#) about further server-side autoinc improvements in 6.3 9051, soon to be released. Bob also hints at what else has been going



on behind the company's mask of silence:

We have got so much going on at the home office. It is exciting times for us all, and I hope that it will be the same for you soon.

You're not alone, Bob, you're not alone.

## Origami unveiled

[Direct link](#)

Posted Thursday, March 09, 2006 by Dave Harms

So what's the deal with Origami anyway? Robert Scoble [interviews](#) the man behind the Origami class devices, Otto Berkes (who is also one of the original four Xbox team members).

Origami is meant to fill the large gap between laptops and other mobile devices. It's a touchscreen handheld, with a full Windows OS. You can run any Windows application on this device. Which means that you can develop Origami apps with Clarion, of course.

Check out the Haiku concept model at about 6:30 in the video. Now an Origami in that form factor would rock, but it's not practical just yet.

There are some hardware limits imposed by the size - this is a 1Ghz class device, with good (but not bleeding edge) graphics. This first device is running XP Tablet Edition, and has an integrated stylus with handwriting recognition. Two USB ports.

Some models will be dockable. You can hook up an external monitor and keyboard.



Native resolution is 800x480, 7" display, same as many portable DVD players. But higher resolutions can be displayed, with some image quality degradation.

Berkes did some hedging on the battery life - evidently the power consumption is still pretty high, and you probably can't expect much more than 2-3 hours for the initial product.

Features include WiFi, Bluetooth, network connectors, stereo speakers and a noise reduction microphone. You could plug in a GPS receiver, use a Bluetooth-enabled GPS receiver. You can also hook up to your phone, and the internet, via Bluetooth.

This is definitely a game-friendly machine, although without the bleeding edge graphics and major horsepower there are some Windows games you won't be able to run. Note that there is no DVD player, so if you want to watch movies (and that's a pretty obvious use) I expect you'll be encouraged to buy/download your media.

These devices have hard drives, and sizes vary from 30 to 120 GB. RAM? I'm not clear on that - again some hedging by Burkes on upgradability. Maybe you'll be able to buy additional RAM, maybe it'll be fixed. That will probably vary from model to model.

While the first models will ship with XP Tablet Edition, some prototypes are running early versions of Windows Vista.

Check out the Microsoft [Ultra-Mobile PC page](#) for more.

## SV news server moves

[Direct link](#)

Posted Wednesday, March 08, 2006 by Dave Harms

The SV news server went onto a new T1 over the weekend, and the transition seems to have been pretty smooth. How soon you got discuss.softvelocity.com back depended on how fast your ISP propagated the DNS changes. If you're still having problems you can use the IP address directly, as noted in [this post](#) by Bob Zaunere. But as Bob also notes, that's not a great way to go because you'll probably forget about the IP changes and you'll get nailed the next time the address changes. And if you have been using the IP address,

either in your hosts file or directly in your newsreader, now's a good time to change it back.

## Revving up the blog machine

[Direct link](#)

Posted Wednesday, March 08, 2006 by Dave Harms

One blog item in a month? Is that me or [SV](#)? Okay, it's me. But that sound you hear in the back ground is the ClarionMag blog machine revving up. In fact, I've accumulated a fair blog backlog, and I need to start clearing some of this stuff out so I can see my desk(top) again.

## Backup, backup, backup

[Direct link](#)

Posted Wednesday, March 01, 2006 by Dave Harms

In softvelocity.public.clarion6, on Feb 17, Ian Cook posted a tribute to iAlchemy's Cover Your Ass(ets). Ian thought he'd lost 12 hours of coding in an IDE crash, until he remembered he'd bought and installed CYA the month before. "I just selected the list of incremental backups and within a few minutes I was up and going again."

From the [CYA web page](#):

Cover Your Ass(ets) is implemented as a global extension as well as a utility that serves in the development process as an "automatic" generational backup mechanism at the application level. Simply dropping the 'iAlchemy: Enable Cover Your Ass(ets)' global extension into any application will automatically create a standard Clarion TXA & TXD at compile time into a user specified folder.

This is a slick idea - CYA runs automatically, generating a TXA with a timestamped

filename each time you compile.

Now, about the *CYA web page*...

Another backup program you should take note of is Jeff Slarve's [In Back](#):

Using a "Project System" in which you tell specifically which files belong to your project, all of those files can be quickly and safely backed up as frequently as desired. Any time that you are about do something to your work that means lots of changes or that could potentially "break" something, all you have to do is press a button and your work will be saved to that point in time. In addition, every time that you back up, a new backup file is created. You don't have to worry about accidentally overwriting your last "good" backup. When the number of backups for that project hits the max limit (that you specify), the oldest backup files are deleted from the disk.

# Clarion Magazine

## The ClarionMag Blog

Get automatic notification of new items! [RSS feeds](#) are available for:

**XML** [All blog entries](#)

**XML** [All new items, including blogs](#)

### Blog Categories

- » [All Blog Entries](#)
- » [Clarion 7 Clarion.NET](#)
- » [Future Articles](#)
- » [News flashes](#)
- » [Nifty Stuff](#)

## The sales dilemma

[Direct link](#)

Posted Tuesday, February 07, 2006 by Dave Harms

So you've written an awesome program, and now, like many independent developers, you need a way to market it. But you don't have (or don't think you have) the necessary skills

to get your program to your customers. Is finding the right marketing person/company the way to go? Sue Pichotta, a longtime Clarion developer turned [icon vendor](#), has a few thoughts on the subject. Read [I'll Write It, You Sell It - Why Not?](#) at the Association of Independent Software Industry Professionals (AISIP) [web site](#). Sue highly recommends AISIP membership. And you want to listen to Sue.

## IBM releases free DB2

[Direct link](#)

Posted Tuesday, February 07, 2006 by Dave Harms

There have been some newsgroup posts about [IBM's release of DB2 Express-C](#), a free version of DB2. In particular note the unlimited database size:

DB2 Express-C may be deployed on all systems up to 2 processor cores, and on AMD or Intel x86 with up to 2 dual core chips. With this offering there is no limit to database size. The maximum amount of memory supported is 4GB.

Bruce Johnson notes that MsSQL Server Express is limited to a 4 GB database, 1 CPU, and a gig of RAM. Oracle has similar limits.

## Popping those system tray balloons

[Direct link](#)

Posted Monday, February 06, 2006 by Dave Harms

If you're annoyed by system tray (okay, notification area) balloon tool tips, you'll be happy to know you can disable them, as explained in this MS TechNet article. [HT: Lee White]

## Microsoft PDC presentations online

[Direct link](#)

Posted Monday, February 06, 2006 by Dave Harms

Presentations from Microsoft's Professional Developers Conference last September, PDC05, are [available online](#). I've read elsewhere that these will only be freely available for six months from release (which was last fall). You will need to open this page in IE - FireFox doesn't get the job done, at least on my machine. Of particular interest, in the Tools and Languages section: [TLN306: The .NET Language Integrated Query Framework: An Overview](#)  
[Speaker: Anders Hejlsberg](#) and [TLN307: C#: Future Directions in Language Innovation from Anders Hejlsberg](#).

I've [blogged about LINQ](#) before; this really is some cool new technology, which you can play with right now. I think it shows particular promise for Clarion developers who need to deal with XML files (have a look at the XLINQ subproject).