This file contains the script for the video included on the CD. It may diverge from the actual spoken text slightly, due to a bit of ad-libbing.

Additionally, you'll find a field by field description of all the customizations done within the dictionary, at the bottom of the file.

Thank you for taking the time to look at the evaluation edition of Clarion for Windows.

What is Clarion for Windows about?

No coding. No compromise. Powerful compiled applications for a broad range of database users: novices, power users, and professional developers.

Clarion for Windows rises to “the next level” of Rapid Application Development—a new tier of programmer-friendly tools that maximize productivity, power, and performance.

The level of automation in creating applications with Clarion for Windows surpasses other RAD tools. With other tools, you create the user interface visually, but the behavior of an application must still be coded by hand. Clarion produces a complete business solution for you: immediately and automatically.

This is what Clarion for Windows brings you:

Rapid Application Development...
at a level beyond other RAD tools

32-bit Compiler...
with the capability to create 16 bit Windows applications, as well.

Application Generator...
an intelligent code generator that provides maximum code reusability with minimum effort

Professional Strength Database Dictionary/Replaceable Database Drivers
There are live links between the dictionary and your application that far surpass the capabilities of one way data wizards in other RAD tools. Direct drivers or ODBC can connect you to your data, no matter where it is and in what form.

Clarion for Windows is
The Only Windows Application Development Environment With:

16 and 32-bit Compilers

That gives you

Windows Platform Independence. You can develop the same app for Windows 3.1, Windows 95, and Windows NT from the same project, using the same tool. You can provide Windows 95 controls, like tool tips and tree controls, to your 16-bit Windows users. You can use your 16 bit .VBX's safely and reliably under 32 bit Windows. If your end users are making the move to 32 bit windows in stages, no other product can help you support them like this one can.

Clarion for Windows also has a powerful Application Wizard.

You get Complete Applications Without Coding. Full featured ones based on the database description, business rules, and application preformatting that you store in the dictionary. You get browse windows listing the records in each table, and links to related tables. Update forms with the control formatting you pre-select, and lists showing related child records where applicable. Reports. You get complete, threaded applications with separate record buffers for each browse the end user opens. All this with absolutely zero coding.

Clarion also has Code Generating Templates

Templates are Complete Business Solutions.

Clarion templates include data and code, like objects. They also include something additional: a design time user interface. That means you as a developer set template properties and functionality by checking a box, or choosing an item from a list. You don't need to consult a reference document to find an obscure function. The application generator takes your input, then generates only the Clarion language code needed to support the functionality you request. There's no such thing as Fatware in Clarion applications. Your completed applications are lean and mean.

More importantly, the templates include intelligent user interface controls. With other RAD tools, you place an object on a window form; for example, a combo box. But then you have to write code to make the combo box do something. With Clarion, the combo box already knows how to do a lookup or update a record. You use its template interface to pick a table and a field. If the end user types in a new value, it even knows how to update a child file. The templates represent complete business solutions -- not simple shortcuts.

The final component in the list is A True Optimizing Compiler

Clarion for Windows give you Small, Fast, Royalty-Free, Native Executables

Clarion features the TopSpeed backend compiler. You get true compiled applications for both 16 and 32 bit Windows. Your applications outperform any interpreter driven RAD applications by miles.

TopSpeed Corporation publishes Clarion for Windows

Founded 1983 as Clarion Software Corp.

Merged With JPI in 1992

which is Known for its TopSpeed Compiler Line, Today the Clarion Language Functions as the “Surface” for the Back End Compiler

In This Demo:

We'll Create a Complete, Complex Application With ZERO Coding

We'll incorporate a 32-bit ODBC Link to a Microsoft Access File

The end result will be a 32-bit Multi-Threaded application with

Browse Windows, Update Forms, and Reports for Seven Tables

Tabbed Dialogs Showing Related Records in Child Tables

And we'll do this
solely by describing the database, the business rules, and some application preformatting in The Database Dictionary

which also Stores a Description of Your Database:Tables, Referential Integrity Options, etc.

It Stores Preformatting Options for Your Application
- Types of Controls
- Appearance of Controls
- Menu and Sort-Tab Descriptions by File and Key

How We'll Do It

Clarion Templates Contain Data Structures and Code
Reads Database Description and Preformatting Options from Dictionary
The Application Generator generates Clarion Language Code
The Optimizing Compiler Creates a Royalty-Free Executable

Clarion Templates Are Complete Business Solutions
When You Place a Control, It Already Knows How To Perform a Lookup, Save A Record, Add a Child Record… and many more
Clarion Templates Include a DesignTime User Interface. To Set Properties and Functionaliy, You Check a Box, Choose From a List, etc.

We'll Do This in Twenty Five Minutes
We'll Do It Without Writing a Single Line of Code.
YOU Can Do the Same for Your Databases by Following the Online Guide!

Let's See...

The Demo:

This is the development environment. Our starting point is a new database dictionary. Clarion is the only Rapid Application Development tool with both a compiler, and a professional strength database dictionary. The dictionary stores a description of your database -- its tables, fields, relations, Referential Integrity options, and much more as you'll soon see.

We're importing table definitions from an existing database. Notice the list of direct drivers. Today we'll use ODBC to connect to a Microsoft Access database called Northwind Traders. It's a sample file that ships with Access 2.0. A wizard lists the tables included in the database. We select one. We type in a description in the File Properties dialog. The Clarion application wizard uses the description in menu item text, window captions, and other parts of the user interface that reference the table.

Notice that when you import an existing database, you never have to declare the fields one by one. It's done for you automatically. In fact, in the case of Microsoft Access files, the import procedure even takes the Access combined date and time fields, and provides Clarion group structures, so that you can place controls that edit either the date or the time separately.

Notice that the File Properties dialog also controls the THREAD attribute. Clarion applications support cooperative threads in 16-bit Windows, and system threads in 32-bit Windows. Each thread gets its own record buffer. This makes for quicker, safer record handling.

All that you need to do is to import the table definitions, define the relationships, optionally preformat controls referencing the database fields, run the Application Wizard, and compile. You get a complete application.

From the names of the tables, it looks like we're building an Order/Entry application. There's no limit on the number of tables you can define in a dictionary. If you're a business programmer, you may have a large accounting package with, say, 250 tables. And you probably have an application backlog. Accounting wants custom apps for managing payroll and G/L tables. Sales wants order/entry apps. Someone else wants mailing labels.

This is Clarion's strength: when you need to create many applications quickly, that share a common look and feel, you preformat the fields in the database dictionary, run the application wizard, pick the tables that each custom app needs to manage, and just hit the Make button.

By spending a couple of days up front, choosing the types of controls and their appearance for each database field, you can do half your development work for the next six months. Since many applications reference the same database, you describe the database once, then generate apps as needed.

The Application Wizard creates a full featured application automatically. There's a browse for each table, with buttons to browse any related tables, and a tab showing each sorting option. There are update forms. If it's a parent file, a property sheet in the update form displays related child records in a listbox. There are reports for each table and key.

This is all automatic. The level of automation in creating applications with Clarion for Windows surpasses other RAD tool. Your starting point is where the others leave off. It's the "Next Level" of Rapid Application Development.

This is the last table to add before we begin defining relationships. Our applications will maintain every table except this one, just to underline that your dictionary is a repository for your database, and that you can create many applications from the same dictionary, but not every application needs to reference all the tables in the dictionary.

Let's begin defining the relationships. The Relationship Properties dialog lists all the tables and all the keys in your database. You define the type of relationship, choose the tables, and choose the keys. We can map the link fields automatically with one button press. You also choose the Referential Integrity Constraint options here.

But wait -- we're using the Jet database engine here, and that handles RI in the engine, right -- wrong! If you're using VB to maintain Access data files, you don't get RI unless you handcode the BeginTransaction and EndTransaction statements. The Direct Access Objects don't support RI. That means VB by default doesn't do RI.

With Clarion, all you have to do is choose the RI behavior for updates and deletes. We'll support RI in the generated code. That means you can support RI in any database. For all the tables in this dictionary, we'll choose Cascade on Update, and Restrict on Delete. That will migrate any changes in a key field to the children, and prevent a deletion in a parent record if there are any related child records.

(4:39)

What's more, the live link between the Database Dictionary and the Application

Generator gives you a tremendous amount of flexibility. Let's say you have an AS/400 database. Up until recently the DBMS didn't support RI. So you add the option in the database dictionary to generate the code. But what happens when you update your DBMS to the latest version, which does support RI? With Clarion, you go back to the dictionary, revise your RI options, then regenerate and recompile. Period.

The links between your dictionary and application file are live. Contrast that with a one-way wizard in other RAD tools. Have you ever needed to revise a database using Delphi? Let's say you need to add a field. The Delphi Database Wizard set you up; but you can't run it a second time, to make a revision. In fact, in most cases, you're best off starting your project over from scratch rather than trying to revise your data definition!

Clarion represents a new type of programmer-friendly tool that maximizes your productivity, yet gives you a level of power and performance in the finished application that you don't expect from a RAD tools. It's a power tool for people who produce apps for a living-- or for anybody who wants to get the job done fast, and right the first time.

We've defined our tables -- let's do some serious pre-formatting. We'll go through the tables in order, first looking at the keys, then the fields.

In the Orders table, the first key is the Order ID number. The field description will appear on the sort tab in the browse dialog for this table. This is the primary key, and it's an auto-number field. We'll automatically generate code to increment the value for every new record entered.

Then we add descriptions for the other keys. In addition to providing the text for the sort order tab, a report will be created for each table and key.

Now is a good time to talk about the template system, which is the basis for the Application Wizard. The templates are like objects -- they contain data, and executable code. They also contain symbols that directly link to the data dictionary. So not only does a template know how to create a browse, or an update form -- it knows how to create a browse for your file, or an update form for your record.

Here you see a dictionary option that specifies that this key not be included as a sort order tab. The templates don't blindly create user interface options for every single item in your dictionary. You can tell it what to pass over.

Now we can preformat fields. Our first field is the Order ID number field. We set the numeric formatting display a whole number without comma separators. We had defined this as an auto increment key. We set the control type to a string control. Since the generated code takes care of updating the value for a new record, there's no reason for the end user to be able to edit it.

We set the numeric formatting for the next field. You can preformat the appearance of any value. This controls how it appears in all template generated user controls. You can do the same for any data variables you declare.

Here's the Order Date field. This is an Access Date/Time field. Our import procedure defined a Clarion group structure that allows us to address the date and the time separately. We first choose an option that tells the templates not to populate the Access

field in any windows. We'll do the same for the overall group structure. We will populate the date field. We won't populate the "space" that comes between the date and the time.

Now look what we can do for the time field. We can predefine a spin box that increases and decreases by one minute each time the end user presses the increase or decrease buttons on the spin box. All we have to do is choose the control type, and set the stepping increment for the spin box to 6000 hundredths of seconds.

The templates will automatically place the spin box in the update forms for you. But even if you design your own window, you can still use that same spin box.

Next we have another date field, and we'll do the same thing we did for the last one. Let's talk about templates a little more.

Templates have something that objects don't -- a design time user interface. The Application Generator will pick a set of templates to fit the database we design here. But each template offers additional customization possibilities for each window or report created. You can edit the procedures with the visual design tools. When you right click on a template object, such as a list box full of database records, the design time user interface appears. You can add a total for a column in that listbox. You can set a range limit, or change the type of locator control. The template provides a user friendly customizing method for you, the developer.

So you see, we provide a wealth of functionality as a starting point. But you have the capability to infinitely customize it. No coding, no compromise. Powerful, compiled applications, for Windows 3.1, Windows 95 and Windows NT. All from a single project file.

These spin boxes are Windows 95 controls. We'll use the native controls under Windows 95. But we also clone the resource for backwards compatibility for your Windows 3.1 apps. You don't have to worry about what the API on the end user's target machine supports. You just tell the compiler to make either a 16 bit or 32 bit app, and we'll do the rest.

It's going to be a while before the world standardizes on 32 bit Windows. No other programming tool that we know of will let you target both 16 and 32 bit Windows operating systems from a single project. Only Clarion can do this. Only Clarion lets you incorporate Windows 95 controls like tool tips and tree controls in your 16 bit apps. Only Clarion works the other way, as well, and lets you incorporate your 16-bit .VBX controls in your 32 bit apps, safely and reliably. You don't have to spend a fortune updating your custom control library until you're ready. This is for real world programmers.

We're up to the second table. The customer code, which is the primary key for this table is an alpha value, so we don't define the key as an auto number key. We'll let the end user type in a value.

Another great thing about Clarion is the Clarion language. For this situation, you might want to embed a few lines of custom code that add a new customer code when opening the window for a new record. The templates provide literally thousands of access points, at which you can insert your own hand coded Clarion language statements. It gives you total control over your application. And the Clarion language is remarkably easy to learn

and to read. It's much easier for programmers coming from a structured language background, such as COBOL and Xbase, to learn Clarion than it is to learn an Object Oriented language, such as Smalltalk.

We'll preformat two fields in this table. Here we have a phone field. We'll use a pattern picture, that displays the phone number in area code - exchange - 4 digit format, with the area code surrounded by parentheses, a space before the exchange, and a dash after the exchange.

And we'll do the same for the fax field. You can put any characters in a pattern picture. It's much easier for the end user, who can enter data more quickly and reliably, and it takes up less space in the database, since you don't store the parentheses, dash and space.

And on to the next table.

This is the Product Categories table. Notice that the keys were defined out of order in the Access database file. The primary key is listed second, rather than first. We'll put a description on the first key. Now for the second, the primary key, we put our description, we set the correct attributes, and now we select an option that will tell the Application Wizard to make this the first sort order tab in the browse window.

Now we preformat our fields. The Category ID number field is the primary key field. We want to display the numeric value without commas, and we want to prevent the end user from editing it, so we change that to a string control. The auto increase code will increment the value for each new record.

The description field is a memo field. You can see that the import procedure brought it in as a 31K string. We're going to demonstrate how you can preformat fields down to the last detail of their appearance. Here we'll set up a text control, whose font and font attributes will be Arial, 10, bold, and red. Let's remember to look for this control when we run the application!

Now the next control is the Picture field. In the Access sample application, this is actually an OLE field. You can't send OLE data via ODBC, because the ODBC spec doesn't support it. So we'll choose the option to skip this field.

If this were "real life," what I'd do is to set up a separate field listing file names, then display graphic files according to the file name. Remember that Clarion offers absolutely superb graphics support. For the best file compression, I'd take advantage of our support for JPG graphics, and display the external files in our Image controls. If you have to support, say, a human resources database, or a real estate database, JPG is simply the best way to store true-color photographic images.

We're up to the Products table. There are several keys to add descriptions for, and a few to tell the Application Wizard not to include.

This is a good point to talk about the default application paradigm embodied in the Application Wizard applications. For each table we define in the dictionary, you get a threaded browse window, which is essentially a list box displaying all the database records in a page loaded listbox. All the locating is done on keys, which provides for

really fast performance, even in very large databases. The browse listbox is on a property sheet, whose tabs list the keys for that table, as we've explained. When the end user clicks on a tab, the sort order of the browse listbox changes.

Each browse is a separate thread in its own MDI window. Contrast this to Delphi, in which MDI support must be completely hand coded. We should also mention that when accessing an SQL database, via ODBC, or with one of our optional SQL drivers, the templates automatically refresh the SELECT every so often. so the end user's data is always up to date. Delphi doesn't do this.

When the end user double clicks a record, or presses the Insert button, an update form appears. It shows all the fields in the database, with the preformatting options you select in the dictionary. The update form includes full support for concurrency checking. You don't have to do anything at all to support a multi-user database. The optimistic concurrency checking implemented in the templates is probably the safest, most reliable method we know of for use in a wide variety of databases.

We're up to the Unit_Price field. We want to pre-format this field as a currency value. Notice that even though the Access field is defined as a string, we can still apply a numeric picture. It's total customization control.

For the next pre-formatted control, we have the Discontinued field. It's a BYTE field, like a logical field in Xbase. We'll preformat the field so that the end user sees a checkbox. You can also define the text that appears alongside the checkbox. You can even define an icon, so that the end user sees a picture control that shows depressed for True, and not depressed for False.

Back to the application paradigm. Whether the Application Wizard places it there, or you place a field in a custom window, the templates contain all the code that implements updating a database record, or reporting back to the end user if it turns out another workstation has changed that record.

Our full name for the default application paradigm is the Browse-Form-Browse paradigm. The next level in the paradigm is a select browse. To fill in a field on the update form, you can optionally call another browse as a lookup. The end user chooses a record from a related file from a list box, presses the Select button, and then that value is filled into the field on the update form. All this can be done with zero coding.

You can really develop a complete application without coding. Any database. Any Windows platform. You get a compiled application.

There's a full description of the default application paradigm in the on line hypertext guide; it's the section called "Prototype."

And here we're pre-formatting fields in the Employees database. We're choosing not to populate portions of the date fields. That is, it's unlikely we need to know what time of day an employee was born -- we just want the date.

Clarion is all about practical programming. And the most practical thing about is that by preformatting so much in the database dictionary, you can save an enormous amount of time developing many applications from the same database dictionary.

It's a different way of development than other RAD tools. Because the Application Wizard and the Application Generator do so much, we move the emphasis to where it belongs for business programmers: to planning. Planning the database. Planning the business rules. Clarion consultants will tell you, that once you've got the database design, the application writes itself. Literally. And our tech support department will tell you something else: that 90% of the time, when a business application goes really wrong, it's the database design.

But let's not forget: you can still put in all the bells and whistles you like, using visual design tools and the Clarion fourth generation language. And even that part is easier using Clarion than with other tools!

Even if you're not a business user. Say you're the power user at the office, the one who realizes that you don't use a spreadsheet to maintain a database. You've been attempting to move it all to an end user database. Well, with Clarion, you can create compiled applications direct from your data files. Read the User's Guide chapter on database design, then save those Excel spreadsheets as .DBF files. You can create compiled applications directly from your database. The Application Wizard walks you through it all.

Do you have data in some PC database format, and want to move it to a more efficient file format? Read the chapter in the User's Guide on the Database Browser. It automatically converts your data files from one file format to another. And you can customize your data types, set up validity options, and preformat fields and keys along the way. You can even store multiple tables in a single physical file, using the TopSpeed file format. Which, by the way, automatically compresses memo fields. Like everything else in Clarion for Windows, it's small, tight, and fast.

Consider the size of your finished applications. This one, at 32-bits, multi-threading, and several reports, will be less than a megabyte and a half. That's about half the size of a VB app with the Jet database engine and Crystal Reports. And it's about a fifth the size of a Delphi app, with the Borland Database Engine and ReportSmith runtimes, and they're not even 32 bits yet.

Back to our database dictionary here. We're finishing up the Employees table, and moving on to the Suppliers table. We're taking care to describe any type of action which we haven't described previously. For this table, we'll add descriptions and attributes to the keys, choosing not to populate some of them.

The Suppliers table has a one to many relationship with the products table. When we run the application and display the Suppliers table in a browse, a button will appear that leads to another browse showing the products for the currently selected supplier. Likewise, an update record for a given supplier will include a tab that displays a listbox when clicked. The end user can then see any related product records for the current supplier.

The template interface makes it simple to add an additional field that counts the number of products included in the listbox, without writing a single line of code. You'll see exactly how to do this in exercise two of the Evaluation Edition on line guide. For your convenience, don't forget that we've provided a separate document with the same

tutorial, formatted to print on letter size paper.

Some of the additional template functionality includes field validation, text viewers, combo boxes that automatically update a child file if a new value is entered by the end user, and more. With other RAD tools, you typically place a user control, such as a button. Then you write code that implements some functionality. Clarion templates let you place a complete business solution. The user control knows how to update a record, display a file, or much more.

There's a thriving market in third party templates, some of which are included as demos on the CD. There's even an evaluation version of one third party template.

And of course, you can write your own templates. The Clarion programmer's motto is write it once, use it many. At some point, you may find that there's some custom procedure unique to your business, that you need to use in many applications. Write a template! The Template Language Reference guide shows you how!

Maximum code reusability. We believe that Clarion can provide far greater code reusabiltity than that promised, but frequently undelivered, by Object Oriented programming languages. That's not to say we're better -- we're different. If you want to work with a RAD tool that requires less coding. And for those situations where you need to code, if you want to work with a structured language that's easy to learn and read, then Clarion is the answer for you.

Is project maintenance killing you? Think about what we're doing in this exercise. We're storing application options on a field by field basis in a database dictionary. The dictionary options can be updated, and the update automatically migrates to the application at the next regeneration and compile. You'll save time.

Even in the custom parts. When you use the Clarion fourth generation language, you can read your own code. The other members of your development team can read your code. Have you ever gone back to a project to make a change six months after you thought you finished it? With an OOP language, it takes a long time to re-familiarize yourself with the code. With Clarion, you hit the ground running, and project maintenance requires less time.

That's it for the tables. We'll close the database dictionary and save it. Remember, we've described an entire application just by describing a database. We've actually customized it by setting up pre-formatting options on a field by field basis in the database dictionary.

Now we're going to run the Application Wizard. We name a fresh application file to create. We choose the database dictionary we just created. Now we run the Application Wizard, which, by the way, was constructed with the Clarion Template Language. You can write your own wizards by studying the source for the wizards, and reading the Template Language Reference.

Here's Mr. Wizard. The first panel is introductory. The second panel allows us to specify all or a selection of the tables in the dictionary. We'll choose to select the tables. Chances are, corporate programmers will create many different applications to maintain various parts of the same database.

We'll select all the files, then deselect the last file, which was the shipper table. We didn't do any preformatting on that table.

Once we press the Finish button, the Application Generator goes to work. It reads in the symbols in the database dictionary that describe the database, and the controls we preformatted. From the file layout, it creates a logical hierarchy of tables, and provides a browse procedure for each. It provides a selection button that displays any linked browses. It provides update forms for each table. And it even creates a report for each field.

It selects the appropriate templates from the template registry, and creates a procedure for each of the windows or reports that reference each table. It then writes the application file. The application file contains all the information that you stored in the database dictionary. It will also store any further customizations that you make, using the visual design tools, such as when you edit controls in a window, or in the Report Formatter. It will store any custom code you place in an embed point within a template procedure. You can even create a source code procedure, which solely contains Clarion language source code you write. You can call Windows API functions, or functions in an external .DLL. The .APP file stores all this.

And then it displays it all in a logical procedure call tree. It's a hierarchally arranged display of all the functionality in your application. It's a different programming paradigm than the forms-based paradigm in other RAD tools. This is more organized -- it's geared for the business programmer. You can find functionality at a glance. There's no code hiding "behind a thousand doors" which is often said about Visual Basic.

Let's set our project option. We'll target a 32-bit Windows system. You make one choice, and don't worry about the rest. Another option in the full version of Clarion for Windows allows you to store all functions in a single executable file. You don't need the library .DLL's that you need with the evaluation edition.

We'll save the application file, and compile. The Run button will generate the code, compile it, and run the application. It's a blue puff of smoke, because as a development tool, Clarion leaves the others in a cloud of dust.

At heart, the Application Generator is an intelligent code generator. It generates Clarion language code that incorporates your database layout, the preformatting options in the database dictionary, and any customizations you add to the application file, using the visual design tools, or Clarion language source.

The Application Generator reads the application file, reads the database dictionary, loads the template registry, then intelligently generates the Clarion language source code.

The project system then generates intermediate symbolic language common to all the TopSpeed compilers. This is passed to the back end TopSpeed compiler. Whether Clarion, C, C++, Modula-2, or another language, the actual end result, that is, the machine code that constitutes your application, is the same. Bottom line, that means there's absolutely no performance difference between an application written with a fourth generation language, in this case Clarion, and a third generation language such as C.

It's speed of development and a speedy application. It's the best of both worlds.

What you see now is the TopSpeed backend compiler creating the object files on a source code module by source code module basis. It's a fast, optimizing compiler, though in this case we've left the debug code in. If you're interested in a little of the history of the TopSpeed compiler, play the audio file we've included on the CD entitled "John Dvorak Radio Show." It's an interview with Niels Jensen, who in addition to founding Jensen and Partners, which is responsible for the TopSpeed compiler line, also founded Borland International. It's a good story.

The file which we're creating is in the new Portable Executable format, for Windows 95 and Windows NT. It's a fully 32-bit, native application for those operating systems. Clarion was the first RAD tool to market with a 32 bit compiler. At the time of this narration, it's the only RAD tool that creates compiled 32 bit applications. And as far as we can tell, it's the only RAD tool that will allow you as a developer to support mixed environments with a single project. If your end users are migrating to 32-bit Windows in steps, perhaps with one department adopting a new operating system a few months ahead of another, you need this product. All the other tools will require that you use two separate tools, maintaining two separate projects, creating two dissimilar applications, if you need to support both 16 and 32 bit Windows for the immediate future.

Clarion for Windows just makes sense. If you need to develop applications quickly to clear an application backlog. If you need to develop applications fast. If you need to maintain a database with a tool that was designed to create database applications from the ground up. If performance is important. If the size of the finished application is important, then Clarion is the answer for you.

And here's the proof. The finished application. We've finished linking, and now we'll execute the application. There's the MDI frame. Let's maximize it. Let's take a look at the Browse menu.

There's all those descriptions we typed in for each table. A menu item for each table that leads to a browse window. It'll take an extra second for the first browse, because the system needs to load the 32-bit ODBC libraries.

Here we are. It's a page loaded listbox showing the records in the Orders file. Notice the tabs for the sort orders. Notice the resizable columns. When you double-click a record, you get the update form.

There's a string control for the key field for which we set the auto number attribute. There were more fields than could fit on one page, so there's an extra tab for the second page. There are our spin boxes for the time field. Each button press adds one minute.

On this tab, we have the child file records with the order detail records for this order. Here's an order detail record, with currency formatting for the unit price, and two string fields for a multiple component key.

Let's see another browse. The Customer file. This is a separate thread. Here's our records. Let's change the sort order. Here's the update form. Notice the phone field formatting. The fax field formatting.

The related order records for this particular customer.

Here's the product categories file. Remember we preformatted a text field to appear in red, arial bold 10 point. Here it is.

Next file. The product list. Here are our products. Here's the update form. The price formatting looks correct. There's our check box for the Discontinued field. Did we write any code for any of this? Not a line.

Next file is the employees file. Here's our records. Here's our update form. Next is our suppliers file. There's the phone field formatting. Look how easy it is for the end user to display the product information from this vendor.

Here's our Invoice Details file. If you look at exercise two, you find out how to add line item totals and column totals without writing a single line of code.

Remember, all the concurrency checking for a multi-user system is built in automatically. All the referential integrity constraints required only a choice from a listbox in the data dictionary.

How about reports. Let's choose one. It automatically print previews. It builds the data for the report, then displays it. These reports are pretty simple. Be sure to check the Order Entry example in the examples subdirectory for an invoice with a pretty form incorporated. Notice the built in zoom function.

Notice the additional functionality. The standard printer selection dialog. Automatic window lists for all open windows. That's a complete nuisance using other tools.

Let's close down the application now. The threads shut down in an orderly fashion. And now we'll close the application development environment.

Presentation part 2

Now Imagine What You Can Do

“It (TopSpeed) designed Clarion for Getting The Job Done, and it is unexcelled in this area, even by Delphi... So whereas Delphi makes it easy to create applications with complex custom interfaces and a unique look and feel, Clarion makes it far easier to generate robust, tight business database applications with a common look and feel—and do it quickly... Clarion is the best for business applications, and that's business with a capital 'B.'”

Infoworld, 8/14/95

What do other Reviewers Say

“Clarion for Windows is better suited for breaking up application backlogs created by complex database requirements.”

PC Week, 10/2/95

“If I were stranded on a desert island and could only take one Windows development tool with me, it would be Clarion for Windows.”

Data Based Advisor, 2/95

“It is arguably the best Windows development tool on the market today.”

PC Techniques, June-July ’95

“TopSpeed Corp. has a winner here.”

DBMS, July ’95

Be sure to look over all the reviews -- they're included on the CD!

So, when you use Clarion for Windows as your development tool:

You'll Write Less Code and be More Productive

That makes you a Better Programmer

The Optimizing Compiler Makes Your Apps Fly

That makes your applications Faster

And you'll find that Project Maintenance is easier

That makes you a Smarter programmer.

This tool will change the way you develop applications.

What You'll Need

Clarion for Windows:
One Copy per Developer
No Other Charges

Distribute Applications Royalty-Free
See the Sales Letter that came with your TrialPak to find out how to order!

Demo Example Instructions

Preliminary: using the ODBC Administrator, set up an ODBC data source for the Northwind Traders database (\ACCESS\SAMPAPPS\NWIND.MDB). In CW, create a new dictionary, then choose File/Import, selecting ODBC as the database type for the tables:

1. Import files in this order:

Table	Description
ORDERS	Invoice List
CUSTOMERS	Customer List
CATEGORIES	Category List
PRODUCTS	Product List
EMPLOYEES	Employee List
SUPPLIERS	Supplier List
ORDER_DETAILS (DTL)	Invoice Line Items
SHIPPERS	Shipping Company List

2. Set the relations and RI constraints:

Tables	Keys
ORDERS->>ORDER_DETAILS	Primary/Reference 4
CUSTOMERS->>ORDERS	Primary/Key_Customer_ID
CATEGORIES->>PRODUCTS	Primary/Key_Category_ID
PRODUCTS->>ORDER_DETAILS	Primary/Key_Product_ID
EMPLOYEES->>ORDERS	Primary/Key_Employee_ID
SUPPLIERS->>PRODUCTS	Primary/Key_Supplier_ID

3. Set Key and Field options:

   a. Orders

Key	Description/Options
PrimaryKey	by Order Number Primary Auto Number
Key_Customer_ID	by Customer Number
Key_Employee_ID	by Employee Number
Key_Order_Date	by Order Date
Reference	(do not populate)
Reference3	(do not populate)
Reference5	(do not populate)

Fields	Options
Order_ID	Picture: @n_13 Window: from Entry to String
Employee_ID	Picture: @n_13
Order_Date	(do not populate)
Order_Date_Group	(do not populate)

Order_Date_Space (do not populate)
Order_Date_Time Spin box
Step=6000 (1 minute)
Required_Date (do not populate)
Required_Date_Group Over Required_Date
(do not populate)
Required_Date_Space (do not populate)
Required_Date_Time Spin box
Step=6000 (1 minute)
Shipped_Date (do not populate)
Shipped_Date_Group Over Shipped_Date
(do not populate)
Shpped_Date_Space (do not populate)
Shipped_Date_Time Spin box
Step=6000 (1 minute)

b. Customers

Key	Description/Options
PrimaryKey	by Customer Number Primary
Key_City	by City
Key_Company	by Company
Key_Region	by Region

Fields	Options
Phone	@P(###) ###-####)P
Fax	@P(###) ###-####)P

c. Categories

Key	Description/Options
Key_Category_Name	by Category Name
PrimaryKey	by Category Number Primary Auto Number Option: Populate 1st

Fields	Options
Category_ID	Picture: @n_13 Window: from Entry to String
Description	Font for text control: Arial, blue
Picture	Do Not AutoPopulate

d. Products

Key	Description/Options
PrimaryKey	by Product Number Primary Auto Number

Key_Category_ID	by Category
Key_Product_Name	by Product Name
Reference	(do not populate)
Reference1_Products	(do not populate)
Key_Supplier_ID	by Supplier ID

Fields	Options
Product_ID	Picture: @n_13 Window: from Entry to String
Supplier_ID	Picture: @n_13
Category_ID	Picture: @n_13
Unit_Price	Picture: @n$6.2
Discontinued	Window: from Entry to Checkbox

e. Employees

Key	Description/Options
PrimaryKey	by Employee Number Primary Auto Number
Key_Last_Name	by Last Name

Fields	Options
Employee_ID	Picture: @n_13 Window: from Entry to String
Birth_Date	(do not populate)
Birth_Date_Group	(do not populate)
Birth_Date_Space	(do not populate)
Birth_Date_Time	(do not populate)
Hire_Date	(do not populate)
Hire_Date_Group	Over Hire_Date (do not populate)
Hire_Date_Time	(do not populate)
Photo	(do not populate)
Notes	Over Photo (do not populate)
Reports_To	Over Photo (do not populate)

f. Suppliers

Key	Description/Options
PrimaryKey	by Supplier Number Primary Auto Number
Key_Company_Name	by Supplier Name

Fields	Options
Supplier_ID	Picture: @n_13 Window: from Entry to String

g. Order Details

Key	Description/Options
PrimaryKey	by Order/Product Primary Auto Number
Key_Product_ID	by Product Number
Reference2	(do not populate)
Reference4	(do not populate)

Fields	Options
Order_ID	Picture: @n_13 Window: from Entry to String
Product_ID	Picture: @n_13 Window: from Entry to String
Unit_Price	Picture: @n$6.2

4. Save and close the dictionary.
5. Choose File/New/App. Do not specify QuickStart.
6. Type in the .APP file name, select the dictionary, and press OK.
7. Step through Application Wizard, populating all files except Shippers.
8. When the Application Tree appears, press the Project button.
9. In the Global dialog, press the Properties button.
10. Set the Target OS to 32. (This assumes a data source for 32-bit ODBC!)
11. Return to the Application Tree and compile!

Final Notes:

1. You must be sure that your ODBC driver is version 2.00.23.17 or higher -- you **cannot** use the ODBC drivers that come with MS Office 4.x -- those are only meant to be used within MS Office. If you're creating a 32-bit app, you need a 32 bit ODBC driver.

2. If you're creating a 16 bit app, you must also reduce the size allocated to memo fields (say, to about 2048 bytes), in the field properties dialogs.