CET W/32 Application Builder Version 9

Overview of the Product,
Technical Specifications,
And
Installation Guide

cet software, incorporated
6595 odell place
boulder, colorado, 80301

Table of Contents

Installation Notes

Introduction

W/32 Application Builder is distributed in two product forms : Development System and Runtime System. In keeping with Microsoft standards, both of these products include an executable program called SETUP.EXE, which will guide you through the installation of W/32 App Builder.

Obtaining W/32 Application Builder

W/32 Application Builder is available from your CET Software Distributor.

W/32 App Builder is enabled by a number of protection mechanisms, including a small hardware device, the CET W/32 KeyPlug, which is plugged into the parallel port of your PC-compatible computer. If you have purchased a prior version of W/32 App Builder, you have already obtained the enabling mechanism, and may simply download and use new versions as they are available without further need to contact a CET Distributor regarding upgrade.

Installing W/32 Application Builder

To install W/32 App Builder, simply launch SETUP.EXE (from an icon or START->RUN) and follow the instructions for installation.

If you choose to download a current release, you should look in the W32-Ver9 directory for the most current SETUP.EXE file, and will be accompanied by a file called RNOTXXXX.TXT (such as RNOT900A.TXT) which will describe the changes in Version XXXX and will provide any other useful information about the release.

Overview and Technical Specifications

Introduction

W/32 Application Builder is a system for designing, developing and installing commercial applications on Microsoft Windows networked systems. W/32 App Builder is an integrated development environment which manages the compilation of BASIC programs, resource files, dialogs, menus and other source objects for integration into a deployable ensemble.

CET recognizes the unique nature of most vertical commercial applications. Vertical commercial application (VCA) systems have a number of unique characteristics which differentiate them from office applications and other commonly used software. These differences include:

- VCA's normally target a specific industry, specialty or company function. Examples are dental office management systems, in-store point-of-sale systems, freight tracking systems, or industry-specific accounting systems. VCA's are often core applications which run an end-user's business and remain operational all of the business day.

- VCA's typically *coordinate data entry and file/database updates*, while maintaining the state of the user's system. This aspect involves the opening, access and update of dozens or hundreds of files (or database tables).

- VCA's are almost always intended for *multi-user operation*. There must exist a reliable mechanism for insuring the integrity of stored data during the course of use by many different users on many different workstations.

- VCA's have, in the past, used *hierarchical menu systems* to organize user workflow. Some method must be used to coordinate the choices made by a user, while preventing other choices from being made depending upon the state of the system.

- VCA's are normally *developed function by function*, and then tied together by means of program jumps from function to function. New functions must be easy to develop and integrate.

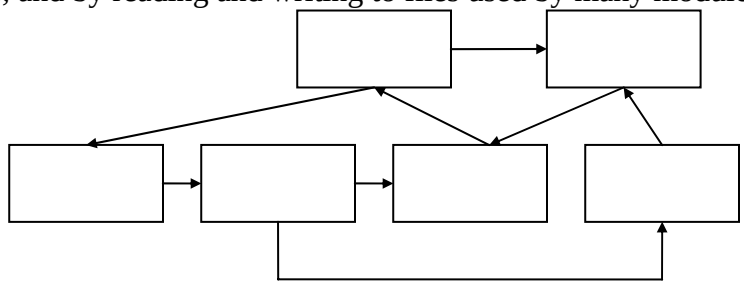
- VCA's are often *customized for each installation*, and upgraded in version levels as the VCA matures or the target industry develops new needs.

- VCA's are usually *developed and marketed by small firms*. These firms must be specialists in the application area, and typically have less resources for application development.

CET has developed W/32 Application Builder with goal of maximizing the functionality of the desired application with the smallest investment on the part of the developer. W/32 App Builder uses CET BASIC, a powerful and familiar 3d generation language, as the base tool of expression for applications.

W/32 Application Architecture

W/32 applications can be designed in the same way as applications written for UNIX and other multiuser systems. In a typical application, program execution proceeds from function to function, depending upon the last user input and the state of the system. Many application modules are reusable (such as print routines), and a system function typically can be reached by different execution paths. Communication between programs is normally performed by passing a list of common variables, and by reading and writing to files used by many modules.



CET BASIC, the principal base language of W/32 App Builder, provides five ways to effect transition among program functions : intra-program jumps (GOTO), intra-progam calls (GOSUB), subroutine calls (CALL), inter-program jumps (CHAIN/LINK/RUN), and event-driven procedure invocations (via mouse movement or clicks). The first three of these (GOTO's, GOSUB's, and CALL's) are common programming constructs which exist in many programming languages. Program chaining is a useful mechanism that is available only in a few commercial languages (such as BASIC, COBOL and RPG). Event-driven program transition is a new method, available in Visual BASIC and other 4th generation development systems, and most commonly associated with languages designed for use in mouse-aware environments (such as MS Windows, Mac/OS and X Window systems).

GOSUB's, GOTO's and CALL's are commonly employed statements in any programming language, and even object-oriented languages recognize the need for their occasional use. Event-driven constructs (such as the event-related routines in MS Access) help organize the coding of routines that execute as the result of asynchronous events, such as a user selection via mouse or other pointing device.

Program chaining is a powerful and useful mechanism that has been left out of many modern programming languages. For many programs, such as office automation applications, program chaining is relatively unnecessary. In the MS Windows environment, program chaining invariably means transition among many large executable programs, a mechanism with many undesirable side effects. These side effects are all the result of the nature of MS Windows: each executing program is associated with a very large and complex state -- open windows, mode of execution, execution priority, files-in-use, locking information, attached DLL's, and shared memory use. It is *expensive* (from a system resource viewpoint) to invoke a new MS Windows executable, and virtually impossible to maintain a smooth transition from one executable to another *in the eyes of the user*.

The W/32 Runtime System : RUNW32

W/32 Application Builder, Version 9, is based upon an old concept – the runtime manager -- deployed in a powerful, event-driven environment – Microsoft Windows.

All W/32 programs are executed under the control of the W/32 Runtime System. This Runtime System operates in much the same way as that used in CET BASIC for DOS, or (using a familiar MS example) as does the MS Access Runtime System. Unlike MS Access and other 4th generation development systems, however, W/32 applications are real compiled code with all of the performance expected in a real executable program.

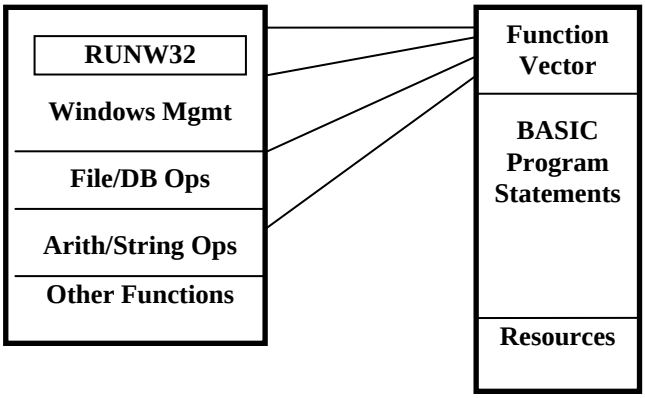
The W/32 Runtime System, RUNW32, is a high-level manager in which resides all of the functionality needed by W/32 BASIC programs. Fundamental program needs (such as opening files and creating windows) are provided by RUNW32. All application library requirements (such as communications, bar-code operations, mail and messaging) are resident in RUNW32 as well. W/32 BASIC programs are the essence of the original coding, and include only the program statements that were originally written.

RUNW32 runs constantly during the course of execution of a W32 application, and it is in RUNW32 that window, data input, and file operations are coordinated. A single RUNW32 executable can service hundreds of W/32 Application modules, and multiple RUNW32 instances on a single workstation will share code, thus reducing memory use and other system resources.

The "magic" of RUNW32 is entirely due to the way in which W/32 BASIC programs are compiled. A W/32 program is compiled and linked to become a single MS Windows Dynamic Linked Library or DLL.

How RUNW32 Works

A Dynamic Linked Library or DLL is a form of executable program that can be attached to a running program at any time during the latter's execution. W/32 programs are DLL's which contain only the original BASIC statement code, and a jump table of W/32 functions.



Basic program requirements are provided by RUNW32. The W32 application only contains the sequencing logic of the program, BASIC program variables, the resource file which was specified at the time of linkage, and an invisible function vector that is linked during execution of OBW32 or W32APBLD. Compiling and linking a W32 BASIC program is very fast, and the resulting "executable" is normally only 15 to 40K (compared to the average Windows executable that can reach many megabytes).

An application is started by the invocation of the command:

RUNW32 <application-name> <application-parameter-list>

The W/32 Runtime System loads the first W32 application, loads the associated resources, and begins execution of the program. The normal delays due to Windows startup overhead is isolated to the first few moments of RUNW32 invocation, and subsequent CHAIN, LINKs, or RUNs of additional W32 BASIC programs is very fast. Because each W32 BASIC application is a small DLL, many of them may reside in system or swap memory, and be available for use almost instantly (and this management is optimized and made invisible by Windows).

You can also invoke an application by simply typing

<application-name> <application-parameter-list>

by clicking on an icon representing the application. The reason is it possible to use the CET program <application-name> as an executable is that *it is an executable* and the first part of every CET W/32 application knows how to invoke RUNW32 using itself as a target.

The OBW32 Compile Manager

The OBW32 Compile Manager coordinates the compilation of W32 BASIC, C and Assembly modules, and links them together into a W32 DLL. The syntax of OBW32 is virtually identical to that of the CET OB managers available on UNIX, DOS and other environments. Combinations of different modules may be linked together, and compiled object modules or libraries may be linked to the executable as well. The resulting "DLL" has the extension ".EXE", because it is an executable and can be invoked either with or without explicitly using RUNW32 on the command line (or in the icon target field).

An application is compiled by invoking the command:

OBW32 <BASIC-Pgm>.b -o <Target-Name> <Other-Parameters>

Please refer to the section on OBW32 (or OBWIN) in the CET W/32 Application Builder Reference Guide.

The W32APBLD Application Builder

The W32APBLD Applications Builder is an integrated development environment in which you may develop and test various objects (programs, resources, forms and dialogs) which will compose your application project. W32APBLD is a pure Windows application and can be started by launching it from its icon.

Please refer to the section on W32APBLD (or W32APP) in the CET W/32 Application Builder Reference Guide.

The W/32 Application Builder Release

W/32 App Builder is shipped in the form of one or more SETUP files. W/32 Application Builder SETUP files are Windows executables which guide the developer through the process of installing

W/32 App Builder and the components which are associated with it (libraries, utilities, and executables).

W/32 App Builder is available from electronic media (the CET FTP Site and CET Worldwide Web Site) in the form of a single executable called SETUP.EXE. When W/32 App Builder is purchased on diskette, the installation procedure is split into separate SETUP files, the first of which is called SETUP.EXE residing on the first diskette.

The W/32 App Builder Runtime Components

The W/32 App Builder Runtime System, RUNW32.EXE, is relatively self-contained. RUNW32 relies on the DLL libraries that are common components of a Windows 95 or Windows NT system, and the re-distributable DLL's from MS Visual C++ in which it is developed. The Visual C++ DLL's include files of the form MSVC*.DLL and MCF*.DLL and are loaded into your Windows System directory when you run SETUP. Other files which affect the behavior of RUNW32 include the W/32 Error file (BRUN.ERR) which resides in the CETBIN directory, the *.W32 file associated with the application, and any variables which have been recorded in the Windows system registry.

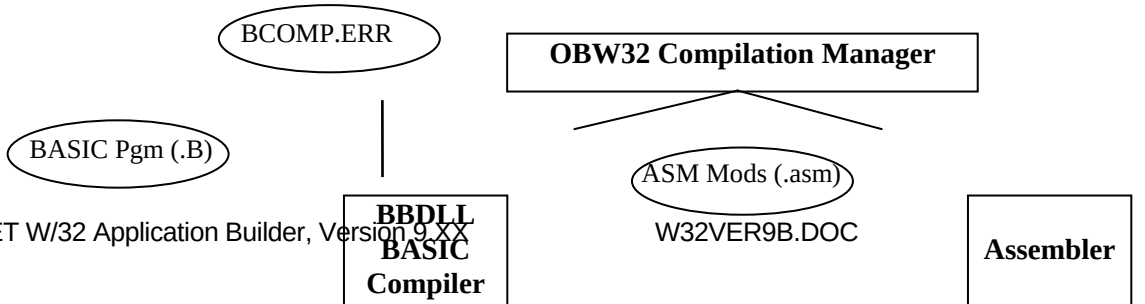
The W/32 OBW32 Compilation Components

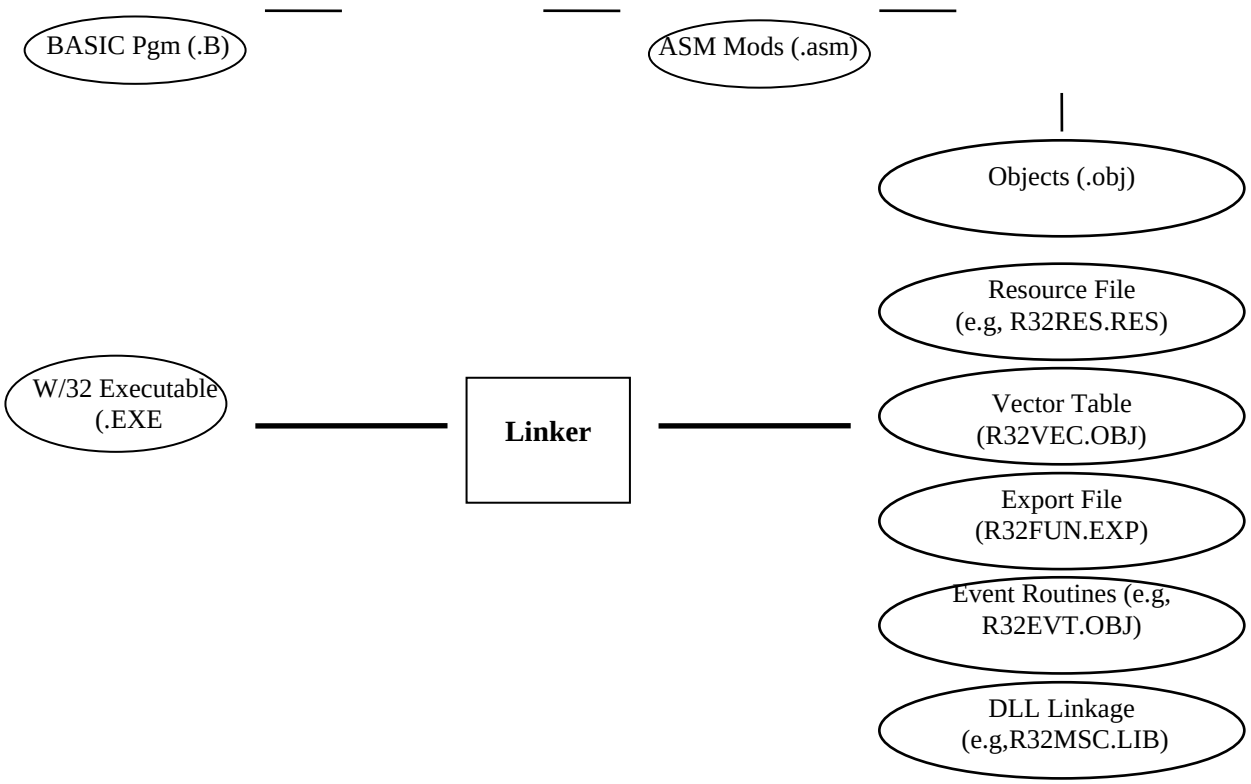
The W/32 App Builder Compiler, OBW32.EXE, is a compiler manager which requires a variety of other executables, libraries, resource files, and object modules to work properly. The simplest OBW32 invocation, such as:

OBW32 MYTEST.B -o MYTEST

will invoke BBDLL.EXE to compile the BASIC program producing MYTEST.ASM, and then invoke the assembler to produce MYTEST.OBJ.

The following diagram shows the components that are bound to make up a CET W/32 program:





The W/32 App Builder Development System is shipped with all of the components necessary for development of W32 Applications. This includes the files and executables shown in the diagram above:

- BCOMP.ERR is the error file containing the text for compilation errors.
- BBDLL.EXE is the compiler executable.
- OBW32.EXE is the compilation manager.
- R32RES.RES and R32RES.RC are the default resource files used in the link step unless the -wr flag is specified to OBW32, in which case a user-defined resource file may be linked. In prior releases of W/32 App Builder, these files were called CETWIN.RC and CETWIN.RES.
- CETRT.DLL contains code which is used by CET W/32 programs at runtime.
- R32EVT.OBJ (and R32EVT.B, which is also shipped in the release) is the default event handling routine BASIC module, unless -wo is used in the OBW32 command to override this file with one of the developer's choice.
- R32MSC.LIB is a small library of object modules providing linkage to Microsoft Visual C++ DLLs.

W/32 App Builder is available from electronic media (the CET BBS, CET FTP Site and CET Worldwide Web Site) in the form of a single executable called SETUP.EXE. When W/32 App

Builder is purchased on diskette, the installation procedure is split into separate SETUP files, the first of which is called SETUP.EXE residing on the first diskette.

Compatibility with Prior Releases

W/32 App Builder Version 9.00 and above represents a radical departure from the method in which prior W/32 App Builder applications were built. Version 8.XX versions of W/32 App Builder and its related components (such as OBWIN) created individual executable programs for each collection of source files which represented a project. W/32 App Builder Version 9.00 and above creates individual loadable DLL files for each project, though they retain the suffix type .EXE.

The functional code for W/32 App Builder Version 9.00 and above resides entirely in the Runtime System, RUNW32.EXE. The RUNW32 executable contains all of the libraries (including CET Application Libraries) that were formerly linked into W/32 applications executables.

To ensure that former versions of W/32 App Builder work properly, CET has renamed any component of W/32 App Builder that would conflict with a prior component of an earlier release. The following list shows the correspondence of Version 9.XX components with Version 8.XX components:

Component Name	Version 8.XX File	Version 9.XX File	In Directory
W/32 App Builder	W32APP.EXE	W32APBLD.EXE	CETBIN
W/32 BASIC Compiler	BB32.EXE	BBDLL.EXE	CETBIN
W/32 Compilation Manager	OBWIN.EXE	OBW32.EXE	CETBIN
Default Resource File	CETWIN.RC	R32RES.RC	CETLIB
User Resource File	CETUSER.RC	R32USER.RC	CETLIB
W/32 Include Directory	CETLIB\INC	CETINC	
W/32 Event Routine	CETWIN.B	R32EVT.B	CETLIB
W/32 Link Library	CETBFSL.LIB	R32MSC.LIB	CETLIB
W/32 Runtime Manager	<None>	RUNW32.EXE	CETBIN
W/32 Uninstall	<None>	UNSETUP.EXE	CETBIN

Differences Between Version 9.XX and Prior Releases

The principal difference between Version 9.XX and prior releases of W/32 App Builder is the use of a runtime system, RUNW32.EXE, to launch applications. There are no intended differences in the behavior of applications compiled for use with RUNW32 *vis a vis* those compiled as executables with Version 8.XX of W/32 App Builder. There are some advantages, however, to the use of Version 9.00, which include:

- **Application Size :** Version 9.XX applications are typically from 50 to 90% smaller than equivalent applications compiled as executables under Version 8.XX. Because of this, distribution of your applications is made easier and the applications launch much faster than prior versions.

- **Load Time :** Version 9.XX applications are DLL's, and are managed by Windows by means of a least recently used scheme for loading and unloading DLL's into memory. In many cases, a CET W/32 application will be small enough that it will still reside in memory at the time of a LINK, CHAIN or RUN statement execution. In these cases, loading and execution will be virtually immediate.

- **CHAIN Time :** Version 9.XX applications are run under the control of the W/32 Runtime Manager, RUNW32.EXE. This manager performs the complex and time-consuming launch process only once, upon startup. In prior versions of W/32 App Builder, each CET application was required to perform Windows startup regimens at every CHAIN, LINK and RUN.

- **Compilation Time :** Version 9.XX of W/32 App Builder uses a much simpler linkage editor step in creating applications. For this reason, compilations of W/32 applications can take from 30 to 80% less time than for Version 8.XX.

- **No Recompilation :** Once an ensemble of W/32 applications is compiled by a developer, any future Runtime Manager can be used to execute the system of programs. Improvements in the W/32 Libraries and corrections to RUNW32 behavior can be obtained by simply downloading a new RUNW32.EXE and installing it. There is never the need to recompile the applications. The absence of recompilation can be an enormous advantage to developers who must update installed applications at remote customer sites.

Other differences to note between Version 9.XX and prior versions include:

- **CETINC:** You must change your INCLUDE environment variable, or the B_CETINC variable in W32APP.W32 to C:\CETINC or whatever directory you specified for the include directory during installation! This is the only major directory that has changed from Version 8.XX. The CETSAMP and CETDOC directories are new, but did not exist in prior releases.

- **OBWIN Flags :** It is not necessary to specify the –O flag, or to include ODBC.OBJ, on the command line or in W32APBLD settings to obtain ODBC support. ODBC support is now available at all time in RUNW32.

- **Iconifying :** You can make an application launchable by creating a shortcut with

  \CETBIN\RUNW32.EXE < app>.EXE

  as the pathname and then select any icon you wish for the shortcut.