CA-Clipper®

For DOS

Version 5.3

Error Messages and Appendices Guide

June 1995

COMPUTER ASSOCIATES®
Software superior by design.

PRINTED ON RECYCLED PAPER

Printed in the United States of America
Computer Associates International, Inc.
Publisher

Contents

Chapter 1 : Introduction

Chapter 2 : Compiler Error Messages

Chapter 3 : Protected Mode Linker Error Messages

Chapter 4: Blinker Error Messages

Chapter 5 : Runtime Errors

Chapter 6 : RMAKE Errors

Chapter 7 : Build and Debugger Error Messages

Chapter 8: DOS Error Messages

Appendix A : ASCII Character Chart

Appendix B : Color Table

Appendix C : CA-Clipper INKEY() Codes

Appendix D : Character Tables

Appendix E : CA-Clipper Reserved Words

Appendix F : dBASE Commands and Functions Not Supported by CA-Clipper

Appendix G : Categorized Language Tables

Appendix H : Obsolete Language Items

Chapter 1 Introduction

This is the *Error Messages and Appendices Guide* for CA-Clipper 5.3. It contains documentation for the error messages associated with several CA-Clipper utilities and an additional chapter containing a table of DOS errors. This guide also includes several useful appendices that are presented in tabular form for quick and easy access.

If you get an error message while running one of the CA-Clipper utilities or while running a CA-Clipper application, use this guide to find an explanation of the error and possible solutions.

For an online version of the error messages, accessible while operating your program editor or any other development utility, use *The Guide To CA-Clipper* error messages database. *The Guide To CA-Clipper* is an online documentation system that uses the Norton Instant Access Engine™.

Most of the appendix tables are also available online using *The Guide To CA-Clipper*. They are categorized under the Tables menu of *The Guide To CA-Clipper* reference database.

Error Messages

This section consists of eight chapters listing the error messages for the CA-Clipper compiler (CLIPPER.EXE), protected mode (EXOSPACE.EXE) and real mode (BLINKER.EXE) linkers, make program (RMAKE.EXE), as well as runtime error messages. Additionally, it includes build and debugger messages you may receive when building or debugging from within the CA-Clipper Workbench. In most of these chapters, the error messages are divided into categories organized by severity. DOS error messages are also included as a separate chapter in this section.

Appendices

There are several appendices at the end of this guide:

- Appendix A is a complete ASCII character table
- Appendix B is a table of color codes used by SETCOLOR() and SET COLOR
- Appendix C is a table of INKEY() codes
- Appendix D is a series of ASCII character tables, divided into categories
- Appendix E is a list of CA-Clipper reserved words
- Appendix F is a list of dBASE commands and functions, not supported in CA-Clipper
- Appendix G consists of several categorized tables of contents for the *Reference Guide*
- Appendix H is a list of obsolete language items with recommended replacements
- Appendix I consists of information regarding CA-Clipper limits, benchmarks, and definitions

Chapter 2
Compiler Error Messages

Introduction

This section is a summary of error messages possible when compiling a program (.prg) file with CLIPPER.EXE. The messages are divided into categories according to severity, and the point where the error occurs in the compilation process. Each category is described below, followed by a listing of all messages organized by category.

The listed compiler error and warning messages are a subset of all the possible messages the compiler can display, and they represent the most likely problems that will be encountered during compilation.

Compiler Warning Messages

Compiler warning messages indicate potential program errors that are not fatal to the compilation process. After a warning, compilation continues with no affect on the generation of the output object (.OBJ) file. The DOS return code is not set when a compiler warning is generated.

The general format of a compiler warning message is as follows:

```
<filename>(<line>):  Error C1xxx <message text>[: <symbol>]
```

Compiler Error Messages

Compiler error messages identify program errors that may be fatal to the compilation process. After a compilation error, the compiler attempts to recover and continue. In most cases, however, recovery will not be possible, and compilation terminates with no object (.OBJ) file generated. The DOS return code is set to 1 when a compiler error is generated.

The general format of a compiler error message is as follows:

```
<filename>(<line>):  Error C2xxx <message text>[: <symbol>]
```

Compiler Fatal Error Messages

Compiler fatal error messages indicate a very serious problem has occurred during the compilation of a program (.prg) file. Because of the severity, compilation terminates immediately and no object file is generated. The DOS return code is set to 1 when a fatal compiler error is generated.

The general format of a fatal error message is as follows:

```
<filename>(<line>):  Error C3xxx <message text>[: <symbol>]
```

Compiler Warning Messages

C1001 Return statement with no value in function

Explanation: You specified a RETURN statement with no return value within a function definition. In CA-Clipper, all function definitions should return a value even though the return value may not be used in the calling routine. If you specify a RETURN statement without a return value, the compiler generates NIL as the default.

Action: Change the declaration from FUNCTION to PROCEDURE, or add the missing return value to the erroneous RETURN statement.

C1002 Procedure returns value

Explanation: You specified a RETURN statement with a return value within a procedure definition. In CA-Clipper, function definitions may return values but procedure definitions may not. If you specify a return value within a procedure definition, the compiler generates this warning and the runtime system returns the specified value in the same way as for a function definition.

Action: Change the declaration from PROCEDURE to FUNCTION, or remove the return value from the erroneous RETURN statement.

C1003 Ambiguous variable reference

Explanation: You referred to an undeclared or unaliased variable when compiling with the /W option. This includes variables you specify within PRIVATE, PUBLIC, or PARAMETERS statements and not declared MEMVAR or referred with the MEMVAR -> alias. Declaration statements include FIELD, LOCAL, MEMVAR, or STATIC.

There are several special cases where this warning can occur:

1. When compiling program (.prg) files containing @...GET, CLEAR, CLEAR ALL, READ, or READ SAVE commands, a warning will be generated for the system variable *GetList*.
2. When compiling program (.prg) files containing database commands such as AVERAGE or SUM, a warning will be generated for undeclared result variables.

Note: This warning is often followed by the "C1004 Ambiguous reference, assuming memvar" warning message.

Action: In general, declare all variables used in a procedure or function. If the variable is a database field, specify the variable prefaced by the work area, and declare it within a FIELD statement.

If the warning refers to the system variable *GetList*, declare it within a MEMVAR statement either at the top of the current procedure or function, or at the top of the current program (.prg) file if you are compiling with the /N option.

C1004 Ambiguous variable reference, assuming memvar

Explanation: You made a reference to an undeclared or unaliased variable when compiling with the /W option, but the context of the program indicated that a MEMVAR declaration could be assumed. This warning usually follows a "C1003 Ambiguous variable reference" warning message.

Action: In general, declare all variables used in a procedure or function. If the variable is a database field, specify the variable prefaced by the work area or FIELD -> alias, or declare it within a FIELD statement.

C1005 Redefinition or duplicate definition of #define

Explanation: You have attempted to #define an identifier that is already defined. This happens when you redefine an identifier without first undefining it with #undef. This problem can occur because you simply omitted the #undef directive (which is an unintended side effect of including a header file that defined the same identifier), or it can occur due to architectural problems in the use of #define directives exist in the current program (.prg) file.

Action: Check first to see if you intended to redefine the specified identifier. If you did, add a #undef directive above the #define directive. If this is not the source of the problem, check to see if the identifier was first defined or redefined in any header file used by the current program (.prg) file.

C1007 Function does not end with RETURN

Explanation: You have defined a function that ends with a statement other than RETURN. This warning can occur if you have specified a statement earlier in the function definition, or if there is no RETURN statement in the function definition. In the case where you have not specified a RETURN statement, the compiler generates NIL as the default return value.

Action: Check first to see if there is a RETURN statement anywhere in the function definition, and add one if there is not. If you have specified one or more RETURN statements within the function, add a RETURN NIL to the end of the function definition.

Compiler Error Messages

C2001 Syntax error

Explanation: A syntax error has occurred somewhere in the current statement.

See Also: The correct syntax for all commands, functions, and statements can be found in the "Language Reference" chapter of the *Reference Guide*.

C2002 Statement unterminated at end of line

Explanation: A syntax error has occurred at the end of the current statement. This usually occurs when there is an unterminated expression or unbalanced parentheses at the end of the statement.

Action: Check the statement carefully, correct, and recompile.

C2003 Syntax error in statement

Explanation: A syntax error has been encountered somewhere in the current statement.

See Also: The correct syntax for all commands, functions, and statements can be found in the "Language Reference" chapter of the *Reference Guide*.

C2004 Illegal character

Explanation: You specified an illegal character within a program statement and not within a literal string. Legal characters for identifiers and keywords include _, a-z, A-Z, ä, Ä, ö, Ö, ü, Ü, ß for the first character, followed by _, a-z, A-Z, 0-9, ä, Ä, ö, Ö, ü, Ü, ß.

Action: Remove the offending character from the program statement and recompile.

C2005 Statement not recognized

Explanation: The current statement cannot be recognized by the compiler. This happens when a command is not recognized by the preprocessor and is passed to the compiler without translation.

This error commonly happens when you fail to specify a required comma separator for certain statement options, such as items in a list. Some examples are the REPLACE command, which requires multiple WITH clauses separated by commas, and the COPY TO command, which requires fields specified as a comma-separated list.

Action: Correct the statement containing the erroneous command and recompile. If you are defining your own command, check the command definition to determine why the specified command is not translated.

C2006 Statement not allowed outside procedure or function

Explanation: You specified a statement other than a FIELD, MEMVAR, or STATIC before the first PROCEDURE or FUNCTION statement and the current program (.prg) file was compiled with the /N option.

Action: Remove the erroneous statement or move it inside the appropriate procedure or function definition. Alternatively, compile without the /N option to force the compiler to automatically generate a procedure definition with the same name as the program (.prg) file.

C2007 Unterminated string

Explanation: You specified a literal character string without a terminating quotation mark. This error may be caused by the inadvertent use of an apostrophe instead of a single quote mark, or it may be a simple omission.

Action: Add the terminating quotation mark. Valid character string delimiters include single quotes ('), double quotes (" "), and square brackets ([]).

Note: Use of square brackets is strongly discouraged.

C2009 Invalid use of @ (pass-by-reference) operator

Explanation: You applied the pass-by-reference operator (@) to an array element or a variable explicitly declared with the FIELD statement. The CA-Clipper runtime system passes both array elements and field variables by value in all cases.

Action: If you applied the @ operator to a field variable, pass the field by value. Fields have global scope, so there may be no reason to pass the field at all.

C2010 Incorrect number of arguments

Explanation: You called a built-in CA-Clipper function with the wrong number of arguments. Unlike calls to user-defined functions or non-built-in functions, the CA-Clipper compiler performs argument checking on built-in functions. Refer to the "Reserved Word" appendix in this guide for a complete list of predefined functions.

Action: Correct the call to the built-in function with the correct number of arguments and recompile. The correct syntax for all functions can be found in the "Language Reference" chapter of the *Reference Guide*.

C2011 EXIT statement with no loop in sight

Explanation: You specified an EXIT statement outside of a DO WHILE or FOR loop. EXIT statements are valid only between a DO WHILE and an ENDDO statement or between a FOR and a NEXT statement. This type of error usually occurs because of a nesting error when specifying control structures.

Action: Check the flow of control in your program's logic for a missing DO WHILE or FOR statement above the erroneous EXIT statement, or remove the erroneous statement.

C2012 LOOP statement with no loop in sight

Explanation: You specified a LOOP statement outside of a DO WHILE or FOR loop. LOOP statements are valid only between a DO WHILE and an ENDDO statement, or between a FOR and a NEXT statement. This type of error usually occurs because of a nesting error when specifying control structures.

Action: Check the flow of control in your program's logic for a missing DO WHILE or FOR statement above the erroneous LOOP statement, or remove the erroneous statement.

C2013 EXIT statement violates enclosing SEQUENCE

Explanation: You specified an EXIT statement between BEGIN SEQUENCE and RECOVER statements without an enclosing DO WHILE or FOR loop. This could happen for several reasons:

1. You specified an EXIT statement with no enclosing DO WHILE or FOR control structure within a SEQUENCE.
2. You are attempting to EXIT from a DO WHILE or FOR control structure enclosing the current SEQUENCE from within the SEQUENCE.

Action: There are different ways of solving this problem:

1. Check first to see if you omitted the DO WHILE or FOR control structure statements. If so, add the control structure statements.
2. Determine if you intended to EXIT from a control structure enclosing the current SEQUENCE. If so, you must BREAK to the RECOVER or the END statements, and then EXIT.
3. Remove the offending EXIT statement.

C2014 LOOP statement violates enclosing SEQUENCE

Explanation: You specified a LOOP statement within BEGIN SEQUENCE and RECOVER statements without an enclosing DO WHILE or FOR loop. This could happen for several reasons:

1. You specified a LOOP statement with no enclosing DO WHILE or FOR control structure within a SEQUENCE.
2. You are attempting to LOOP to a DO WHILE or FOR statement outside the current SEQUENCE from within the SEQUENCE.

Action: Some suggestions to resolve the problem are:

1. Check first to see if you omitted the DO WHILE or FOR control structure statements. If so, add the control structure statements.
2. Determine if you intended to LOOP to a control structure enclosing the current SEQUENCE. If so, you must BREAK to the RECOVER or the END statements and then LOOP.

C2015 Illegal initializer

Explanation: You declared a STATIC variable with an initializer that is not a literal value. Because a static initializer is computed at compile time and assigned before the beginning of execution, it must consist entirely of constants and simple operators (no variables or function calls). Initializers for variables of other storage classes (e.g., LOCAL, PRIVATE, and PUBLIC) can be any valid CA-Clipper expression.

Action: Some suggestions to resolve the problem are:

1. Change the static initializer to a constant or literal value and recompile.
2. If you want to initialize a static variable with an expression, you must do it within the body of a function or procedure. The following method shows how it can be done:

```
STATIC cVar
.
.
.
IF cVar == NIL
   cVar := <xExp>
ENDIF
```

C2016 Name conflicts with previous declaration

Explanation: You specified a PRIVATE, PUBLIC, or PARAMETER statement containing a variable with the same name as a declared LOCAL STATIC, FIELD, or MEMVAR currently within scope. If you use the /N (no automatic procedure) option, this includes any filewide declaration that occurs before the first PROCEDURE or FUNCTION declaration.

Action: Change the erroneous variable name to another name.

C2017 Duplicate variable declaration

Explanation: You declared a variable with the same name as a previous variable of the same scope. For example, STATIC x, followed by LOCAL x.

Action: This usually occurs when declaring many variables in several declaration statements and a variable is specified twice. Remove one of the duplicate declarations.

C2018 Outer block variable out of reach

Explanation: Within a nested code block, an inner block contains a reference to a variable declared as a parameter within the outer block.

Action: Put the parameter declaration in the same block that references the variable.

C2019 CALL of CA-Clipper procedure or function

Explanation: You invoked a previously declared procedure or function with the CALL command. CALL is designed to invoke separately compiled or assembled routines and not CA-Clipper-compiled procedures and functions.

Action: Some suggestions to resolve the problem are:

1. Rename either the external routine name, or the conflicting procedure or function name.
2. If you are compiling multiple program (.prg) files into a single object (.OBJ) file, check to see if the conflicting procedure or function is in another program (.prg) file. If it is in another program (.prg) file, and if it is not called by procedures and functions outside of the program (.prg) file, consider declaring it a STATIC FUNCTION or PROCEDURE.

C2020 Mistreatment of CALLed symbol

Explanation: A previously CALLed procedure has been invoked with either a DO statement or a function call.

Action: Make sure that routines you intend to CALL do not have the same names as CA-Clipper routines defined in your program.

C2021 Redefinition of CA-Clipper procedure or function

Explanation: You specified a PROCEDURE or FUNCTION declaration for a procedure or function name already declared within the same scope. Procedures and functions in CA-Clipper are either *public* (visible to all other routines in a program) or *static* (visible only to routines declared within the same program file).

Action: This error is usually generated because of a name conflict between two public functions. If both functions are indeed public functions, rename one of them. If either or both could be STATIC, declare them accordingly.

C2022 Redefinition of predefined function

Explanation: You declared a function with the same name as a predefined CA-Clipper function. Unlike ordinary CA-Clipper functions, predefined function names are reserved and cannot be redefined. Refer to the "Reserved Word" appendix in this guide for a complete list of predefined functions.

Action: Rename your function or procedure to resolve the conflict and recompile.

C2023 CA-Clipper definition of CALLed symbol

Explanation: A previously CALLed procedure name has been declared in a PROCEDURE or FUNCTION statement.

Action: This conflict can only be resolved by renaming either the CALLed procedure name or the declared procedure or function name.

C2024 Unclosed control structures

Explanation: You specified a control structure without specifying the terminating END statement.

Action: Check all BEGIN SEQUENCE, DO CASE, DO WHILE, FOR, and IF statements for corresponding END statements and make sure that all structures are properly nested.

C2025 ELSE does not match IF

Explanation: You specified an ELSE statement outside an IF...ENDIF control structure. ELSE is part of the IF control structure syntax and, therefore, cannot occur outside of the structure.

Action: Remove the offending ELSE statement or fix the nesting error, and recompile.

C2026 ELSEIF does not match IF

Explanation: You specified an ELSEIF statement outside an IF...ENDIF control structure. ELSEIF is part of the IF control structure syntax and, therefore, cannot occur outside of the structure.

Action: Remove the offending ELSEIF statement or fix the nesting error, and recompile.

C2027 ENDIF does not match IF

Explanation: You specified an ENDIF statement without a corresponding IF statement. ENDIF is part of the IF control structure syntax and, therefore, cannot occur outside of the structure. This error may be due to improperly nested IF structures.

Action: Remove the offending ENDIF statement or fix the nesting error, and recompile.

C2028 ENDDO does not match WHILE

Explanation: You specified an ENDDO statement without a corresponding DO WHILE statement. ENDDO is part of the DO WHILE control structure syntax and, therefore, cannot occur outside of the structure. This error may be due to improperly nested DO WHILE structures.

Action: Remove the offending ENDDO statement or fix the nesting error, and recompile.

C2029 NEXT does not match FOR

Explanation: You specified a NEXT statement without a corresponding FOR statement. NEXT is part of the FOR control structure syntax and, therefore, cannot occur outside of the structure. This error may be due to improperly nested FOR...NEXT structures.

Action: Remove the offending NEXT statement or fix the nesting error, and recompile.

C2030 ENDCASE does not match DO CASE

Explanation: You specified an ENDCASE statement without a corresponding DO CASE statement. ENDCASE is part of the DO CASE control structure syntax and, therefore, cannot occur outside of the structure. This error may be due to improperly nested DO CASE structures.

Action: Remove the offending ENDCASE statement or fix the nesting error, and recompile.

C2031 CASE or OTHERWISE is not immediately within DO CASE

Explanation: You specified a CASE or OTHERWISE statement that does not fall within a DO CASE...ENDCASE control structure. CASE and OTHERWISE are part of the DO CASE control structure syntax and, as such, cannot occur outside of the structure.

Action: Check to see if the CASE structure begins with a DO CASE statement, adding one if it is missing. Remove the offending CASE or OTHERWISE statements and recompile.

C2032 TEXT statement error

Explanation: A problematic TEXT statement has been encountered.

Action: This error might occur if you attempt to redefine the TEXT command with a translation directive and have some error in the underlying output routine.

C2033 Missing ENDTEXT

Explanation: You specified a TEXT statement without a corresponding ENDTEXT statement. TEXT is similar to a control structure; it must be terminated with an ENDTEXT statement.

Action: Add an ENDTEXT statement to the end of the TEXT block.

C2034 Formal parameters already declared

Explanation: You declared a function or procedure with both declared parameters (e.g., FUNCTION (a, b, c)) followed by a PARAMETERS statement. Specifying only one of the two parameter definitions is legal.

Action: Move all parameters listed in the PARAMETERS statement to the procedure or function declaration statement, and then remove the PARAMETERS statement.

C2035 Invalid declaration

Explanation: An error was detected in a variable declaration statement (e.g., LOCAL, MEMVAR).

Action: Check the indicated declaration statement for proper syntax.

C2036 Mayhem in CASE handler

Explanation: You specified statements between a DO CASE statement and the first CASE or OTHERWISE statement. No statements are allowed between these statements.

Action: Either remove the statements, or move them to the proper case handling block.

C2037 Invalid procedure name in DO statement

Explanation: You specified a DO statement with an *<idProcedure>* argument that is not a valid identifier. Identifiers in CA-Clipper must begin with a letter or an underscore which may be followed by any combination of letters, numbers, or underscores.

Action: Correct the procedure name and recompile.

C2038 Invalid target name in CALL statement

Explanation: You specified a CALL statement with an *<idProcedure>* argument that is not a valid identifier. Identifiers in CA-Clipper must begin with a letter or an underscore which may be followed by any combination of letters, numbers, or underscores.

Action: Correct the procedure name and recompile.

C2039 Invalid selector in send

Explanation: An unknown message has been sent (with the : operator) to an object class. The reason for this error is either a misspelled method name or the message was sent to an object of the wrong class.

Action: The "Language Reference" chapter of the *Reference Guide* has a complete list of methods for each class of objects.

C2040 Invalid unary inline operator

Explanation: The expression in question is not specified correctly.

Action: Examine the expression carefully and correct any obvious mistakes. Also, use parentheses to clarify the expression. This increases readability and makes problems more evident.

C2041 Invalid binary operator

Explanation: The expression in question is not specified correctly.

Action: Examine the expression carefully and correct any obvious problems. Also, try using parentheses to clarify the expression and make it more readable.

C2042 Invalid lvalue

Explanation: You specified a value on the left side of an assignment operator that is not a valid identifier. Only identifiers can be assigned values or references.

The error could happen if you attempted to use the inline assignment operator (:=) when you intended to use the compare operator (==), or it may be a more obvious error (e.g., 5 := y + 10).

Action: Correct and recompile.

C2043 Invalid alias expression

Explanation: You specified an identifier on the left side of the alias operator (->) that is not a valid alias name. The left side of the alias operator can only be an identifier or a macro expression.

Action: Check to see that you have not specified an expression as the alias name.

C2044 Invalid function name

Explanation: You called a function with a name that is not a valid identifier. Identifiers in CA-Clipper must begin with a letter or an underscore which may be followed by any combination of letters, numbers, or underscores.

Action: Correct and recompile.

C2045 Target name was used previously in non-CALL context

Explanation: You specified a CALL to a procedure or function that you previously invoked with a DO statement or a function CALL.

Action: Make sure that routines you CALL do not have the same names as CA-Clipper routines defined in your program. Correct and recompile.

C2046 SEQUENCE nesting error

Explanation: You specified a BEGIN SEQUENCE...END control structure with a nesting error. Control structures must be nested properly. Furthermore, RECOVER and BREAK cannot be used outside of a BEGIN SEQUENCE...END structure.

Action: Make sure that each BEGIN has a matching END and that any other control structure used within the SEQUENCE is also closed with a matching END.

C2047 GET contains complex macro

Explanation: You attempted to GET a macro expression (e.g., &(<expression>)). Legally, you may GET a memory, field, or macro variable.

Action: Some suggestions to resolve the problem are:

1. Assign the macro expression to a variable and GET the variable.
2. Check to see if you meant to GET a macro variable and change the macro expression to a macro variable.

C2048 GET contains both macro and declared symbol references

Explanation: An attempt to specify the GET variable using a declared variable in combination with a macro operator (&) has been encountered.

Action: Assign the expression to a variable and then GET the variable.

C2049 Code block contains complex macro

Explanation: An attempt to use a macro expression (e.g., &(<expression>)) within a code block has been encountered.

Action: Assign the macro expression to a variable and use the variable in the code block.

C2050 Code block contains both macro and declared symbol references

Explanation: A code block has been encountered containing a declared variable used in combination with the macro operator (&).

Action: Assign the expression to a variable and use the variable in the code block.

C2051 LOCAL declaration follows executable statement

Explanation: You specified a LOCAL statement following an executable statement. All variable declaration statements (FIELD, LOCAL, MEMVAR, STATIC) must precede any executable statement in a procedure or function definition.

Action: Move the LOCAL statement before any executable statement within the procedure or function definition, and recompile.

C2052 MEMVAR declaration follows executable statement

Explanation: You specified a MEMVAR statement following an executable statement. All variable declaration statements must precede any executable statement in a procedure or function definition.

Action: Move the MEMVAR statement before any executable statement within the procedure or function definition, and recompile.

C2053 FIELD declaration follows executable statement

Explanation: You specified a FIELD statement following an executable statement. All variable declaration statements (FIELD, LOCAL, MEMVAR, and STATIC) must precede any executable statement in a procedure or function definition.

Action: Move the FIELD statement before any executable statement in the procedure or function definition, and recompile.

C2054 STATIC declaration follows executable statement

Explanation: You specified a STATIC statement following an executable statement. All variable declaration statements must precede all executable statements in a procedure or function definition.

Action: Move the STATIC statement before any executable statement within the procedure or function definition, and recompile.

C2055 Syntax error in #define

Explanation: You specified a #define directive containing a syntax error. The syntax for this and all other preprocessor directives can be found in the "Language Reference" chapter of the *Reference Guide.*

Action: Correct and recompile.

C2056 Unexpected end of file in #define

Explanation: You specified a #define directive containing an end of file CHR(26) mark. The syntax for this and all other preprocessor directives can be found in the "Language Reference" chapter of the *Reference Guide.*

Action: Correct and recompile.

C2057 Label missing in #define

Explanation: A #define directive has been encountered with no identifier.

See Also: The syntax for this and all other preprocessor directives can be found in the "Language Reference" chapter of the *Reference Guide.*

C2058 Comma or right parenthesis missing in #define

Explanation: You specified a pseudofunction definition with a missing comma or right parenthesis. When you define a pseudofunction, you must specify matching left and right parentheses as well as separate all function arguments with commas.

Action: Check the definition closely, supply the missing character, and recompile.

C2059 Missing => in #translate/#command

Explanation: You specified a #translate/#xtranslate or #command/#xcommand directive without the => separator. Each #translate/#command directive requires the => symbol even if the definition has no result pattern. The syntax for this and all other preprocessor directives can be found in the "Language Reference" chapter of the *Reference Guide.*

Action: Correct and recompile.

C2060 Unknown result marker in #translate/#command

Explanation: You specified a #translate/#xtranslate or #command/#xcommand directive containing a reference to a result marker name in the result pattern which was not defined in the match pattern.

Action: Check the spelling of the result marker to make sure that it has a corresponding match marker name on the left side of the translation directive.

C2061 Label error in #translate/#command

Explanation: You specified a #translate/#xtranslate or #command/#xcommand directive containing a reference to an erroneous match marker. The syntax for this and all other preprocessor directives can be found in the "Language Reference" chapter of the *Reference Guide.*

Action: Correct or remove the offending match marker and recompile.

C2062 Bad match marker in #translate/#command

Explanation: You specified a #translate/#xtranslate or #command/#xcommand directive containing a reference to an illegal match marker. The syntax for this and all other preprocessor directives can be found in the "Language Reference" chapter of the *Reference Guide.*

Action: Correct and recompile.

C2063 Bad result marker #translate/#command

Explanation: You specified a #translate/#xtranslate or #command/#xcommand directive containing a reference to an illegal result marker. The syntax for this and all other preprocessor directives can be found in the "Language Reference" chapter of the *Reference Guide.*

Action: Correct and recompile.

C2064 Bad restricted match marker in #translate/#command

Explanation: You specified a #translate/#xtranslate or #command/#xcommand directive containing a reference to an illegal restricted match marker. The syntax for this and all other preprocessor directives can be found in the "Language Reference" chapter of the *Reference Guide.*

Action: Correct and recompile.

C2065 Empty optional clause in #translate/#command

Explanation: You specified a #translate/#xtranslate or #command/#xcommand directive containing an empty optional clause. Optional clauses are defined in the #command and #translation directives using square brackets. This error indicates consecutive opening and closing square brackets with nothing in between.

Action: Add the missing optional clause or delete the extraneous square brackets.

C2066 Unclosed optional clause in #translate/#command

Explanation: You specified a #translate/#xtranslate or #command/#xcommand directive containing an optional clause with a missing right square bracket. All optional clauses must be enclosed in square brackets.

Action: Add the missing square bracket and recompile.

C2067 Too many nested #ifdefs

Explanation: The maximum number of 16 levels of nested #ifdef directives has been exceeded.

Action: Reduce the number of nested definitions.

C2068 Error in #ifdef

Explanation: You specified a #ifdef directive containing an error. The syntax for this and all other preprocessor directives can be found in the "Language Reference" chapter of the *Reference Guide.*

Action: Correct and recompile.

C2069 #endif does not match #ifdef

Explanation: You specified a #endif directive without a corresponding #ifdef or #ifndef.

Action: Check to see that each #endif has a corresponding #ifdef or #ifndef directive, correct, and recompile.

C2070 #else does not match #ifdef

Explanation: You specified a #else directive that does not fall within a #ifdef or #ifndef control structure.

Action: Check to see that each #else directive you specified is nested between an #ifdef or #ifndef and a #endif directive, and then, correct and recompile.

C2071 Error in #undef

Explanation: You specified a #undef directive that contains an error. The syntax for this and all other preprocessor directives can be found in the "Language Reference" chapter of the *Reference Guide.*

Action: Correct and recompile.

C2072 Ambiguous match pattern in #translate/#command

Explanation: You specified a #translate/#xtranslate or #command/#xcommand directive containing two consecutive optional match marker clauses with no intervening keyword.

Action: This is not allowed since the preprocessor has no way of knowing which match marker to use for matching input text. Refer to the "Language Reference" chapter of the *Reference Guide* for more information and examples using the translation directives.

C2073 Result pattern contains nested clauses in #translate/#command

Explanation: You specified a #translate/#xtranslate or #command/#xcommand directive containing a nested repeating result clause (i.e., nested square brackets). Nested clauses are not allowed in result patterns. Attempting this usually means that you need two rules instead of one.

Action: Correct and recompile.

C2075 Too many locals

Explanation: You have exceeded the maximum of 255 local variables per function definition. Use of this many local variables is probably an indication of a function that is too complex.

Action: Divide the routine into two or more subroutines.

C2076 Too many parameters

Explanation: You specified a function or procedure definition with more than the maximum of 255 declared parameters. Use of this many parameters indicates a routine that is too complex.

Action: Some suggestions to resolve the problem are:

1. Divide the routine into two or more subroutines.
2. Use an array instead of variables to pass a large number of values between routines.

C2077 Too many parameters

Explanation: You specified a function or procedure definition with more than the maximum of 255 parameters in a PARAMETERS statement. Use of this many parameters indicates an overly complex routine.

Action: Some suggestions to resolve the problem are:

1. Divide the routine into two or more subroutines.
2. Use an array instead of variables to pass a large number of values between routines.

C2078 Circular #define

Explanation: You specified a #define directive in terms of itself (e.g., #define x y followed by #define y x). A circular #define cannot be resolved by the preprocessor and is, therefore, not allowed.

Action: Correct and recompile.

C2079 Circular #translate/#command

Explanation: You specified a #translate/#xtranslate or #command/#xcommand directive in terms of itself. A circular #translate or #command cannot be resolved by the preprocessor and is, therefore, not allowed.

When using #command and #translate, keep in mind that both directives match four-character abbreviations for keywords. This may cause an unexpected match which could lead to this error.

Action: Some suggestions to resolve the problem are:

1. Check to see if the problem is related to the four-letter rule using either #translate or #command. If it is, change the directive to either #xtranslate or #xcommand as both require an exact match.
2. Remove the circular definition and recompile.

C2086 RETURN violates enclosing SEQUENCE

Explanation: You specified a RETURN statement between a BEGIN SEQUENCE statement and the corresponding RECOVER statement (or END statement, if no RECOVER statement is used).

The only valid ways to exit from a SEQUENCE are to BREAK or to execute across the end of the code in the SEQUENCE.

Action: Reorganize your code so that the RETURN occurs in the RECOVER code or outside the SEQUENCE, and recompile.

Note: This error is a deviation from previous releases of CA-Clipper in which it was allowed.

Compiler Fatal Error Messages

C3001 Out of memory

Explanation: The compiler has run out of memory.

Action: The easiest solution is to make the program smaller by reducing the number of procedures or functions defined in the program (.prg) file or reducing the number of program files in the compiler script file. Reducing the number of preprocessor directives used in the application may also help. If possible, you could make more memory available to the compiler by temporarily unloading any memory-resident programs currently in memory.

C3002 Input buffer overflow

Explanation: You specified a program statement that is too long for the preprocessor.

Action: Divide the statement into several smaller statements and recompile.

C3003 Can't open intermediate file

Explanation: The compiler created a temporary file to contain intermediate results and could not open the file at a later time. This error is extremely unlikely to occur.

Action: If the TMP environment variable is set to a network drive, make sure that the drive mapping is still valid. Ensure that there is a sufficient number of file handles.

C3004 Bad command line option

Explanation: You specified an illegal or unknown compiler command-line option. For a complete list of the valid command-line options, execute CLIPPER.EXE with no arguments.

Action: Correct and recompile.

C3005 Bad command line parameter

Explanation: You specified a compiler command-line option with an unknown or unrecognizable argument. For the correct syntax of the compiler command line, execute CLIPPER.EXE with no arguments.

Action: Correct and recompile.

C3006 Can't create preprocessed output file

Explanation: The compiler cannot create the preprocessed output file when the /P command-line option is specified. This can happen for several reasons:

1. You have run out of file handles.
2. You have run out of disk space.
3. You do not have create rights on a network.

Action: This error can be resolved in the following ways:

1. Increase the number of file handles by changing the FILES statement in our CONFIG.SYS file. Reboot and recompile.
2. Delete unwanted files from disk and recompile.
3. Ask your network administrator for create rights in the directory in which you are compiling.

C3007 Can't open #include file

Explanation: The preprocessor cannot open the file specified in a #include directive. The file is either misspelled or not located in any of the directories specified by the INCLUDE environment variable.

Action: Correct the problem by changing the filename including its directory in your DOS INCLUDE environment variable, or by using the /I option to temporarily add one or more directories to the front of the INCLUDE directory list.

C3008 Bad filename in #include

Explanation: You specified a #include directive with a file specification that is not a literal string enclosed in quotation marks (e.g., "Initial.ch"). In CA-Clipper, brackets (<>) are not supported.

Action: Enclose the literal filename in quotation marks and recompile.

C3009 Too many nested #includes

Explanation: The maximum number of nested #includes has been exceeded. CA-Clipper allows a maximum of 16 nested #includes.

Action: Change your header file architecture to a flatter structure.

C3010 Invalid name follows #

Explanation: An unknown or invalid directive was encountered by the preprocessor. You have either misspelled the directive or used one that the preprocessor does not support. A complete list of preprocessor directives can be found in the "Language Reference" chapter of the *Reference Guide*.

Action: Correct and recompile.

C3011 Can't open standard rule file

Explanation: The compiler cannot open the standard header file (i.e., either Std.ch or the file specified with the /U compiler option).

Action: The compiler searches for the standard header file as it does for any other header file, looking in the current directory and then in the INCLUDE path. Update the INCLUDE environment variable with the proper location or add a new location to the front of the current include list with the /I option.

C3012 Too many standard rules

Explanation: There are too many rules in the standard header file (i.e., either Std.ch or the file specified with the /U compiler option).

Action: Take some of the standard rules and put them in another header file. Then, #include this header file at the beginning of each program file in your application.

C3013 Expression stack overflow

Explanation: You have specified an extremely large procedure or function, or an extremely complicated expression.

Action: If possible, break up the procedure, function, or expression.

C3014 Expression stack underflow

Explanation: You have specified an extremely large procedure or function, or an extremely complicated expression.

Action: If possible, break up the procedure, function, or expression.

C3015 Control stack overflow

Explanation: This error can be caused by too many nested control structures.

Action: Examine your program and simplify it.

C3016 Control stack underflow

Explanation: This error can be caused by too many nested control structures.

Action: Examine your program and simplify it.

C3017 Error reading or opening script file

Explanation: The compiler cannot find or open the specified script (.clp) file. Compiler script (.clp) must be located in the current directory unless you explicitly specify the path. You must explicitly specify the script file extension if the file has an extension other than .clp.

Action: This error can be corrected in two ways:

1. Make sure the file is located in the current directory or specify the path as a part of the script file specification.
2. If the extension is not .clp, specify the extension as a part of the file specification.

C3018 Too many symbols

Explanation: The program file(s) compiled into the current object file uses too many symbols.

Action: Simplify the program so that you use fewer symbols (e.g., reuse variable names when possible instead of defining new ones).

C3019 Too many publics

Explanation: This error can occur if you are using automatic compilation to compile too many program (.prg) files into a single object (.OBJ) file at one time.

Action: Some suggestions to resolve the problem are:

1. Try breaking up the compile into two or more smaller groups of files.
2. Change your overall program building strategy to use single-file compilation where a single program (.prg) file is compiled into a single object (.OBJ) file. To compile using single-file compilation mode, use the /M option.

C3020 Too many segments

Explanation: This error can occur if you are using automatic compilation to compile too many program (.prg) files into a single object (.OBJ) file at one time.

Action: Some suggestions to resolve the problem are:

1. Try breaking up the compile into two or more smaller groups of files.
2. Change your overall program building strategy to use single-file compilation where a single program (.prg) file is compiled into a single object (.OBJ) file. To compile using single-file compilation mode, use the /M option.

C3021 Too many fixups

Explanation: This error can occur if you are using automatic compilation to compile too many program (.prg) files into a single object (.OBJ) file at one time.

Action: Some suggestions to resolve the problem are:

1. Try breaking up the compile into two or more smaller groups of files.
2. Change your overall program building strategy to use single-file compilation where a single program (.prg) file is compiled into a single object (.OBJ) file. To compile using single-file compilation mode, use the /M option.

C3022 Too many external references

Explanation: This error can occur if you are using automatic compilation to compile too many program (.prg) files into a single object (.OBJ) file at one time.

Action: Some suggestions to resolve the problem are:

1. Try breaking up the compile into two or more smaller groups of files.
2. Change your overall program building strategy to use single-file compilation where a single program (.prg) file is compiled into a single object (.OBJ) file. To compile using single-file compilation mode, use the /M option.

C3023 Too many labels

Explanation: This error can occur if you are using automatic compilation to compile too many program (.prg) files into a single object (.OBJ) file at one time.

Action: Some suggestions to resolve the problem are:

1. Try breaking up the compile into two or more smaller groups of files.
2. Change your overall program building strategy to use single-file compilation where a single program (.prg) file is compiled into a single object (.OBJ) file. To compile using single-file compilation mode, use the /M option.

C3024 Too many procs

Explanation: This error can occur if you are using automatic compilation to compile too many program (.prg) files into a single object (.OBJ) file at one time.

Action: Some suggestions to resolve the problem are:

1. Try breaking up the compile into two or more smaller groups of files.
2. Change your overall program building strategy to use single-file compilation where a single program (.prg) file is compiled into a single object (.OBJ) file. To compile using single-file compilation mode, use the /M option.

C3025 Too many proc requests

Explanation: This error can occur if you are using automatic compilation to compile too many program (.prg) files into a single object (.OBJ) file at one time.

Action: Some suggestions to resolve the problem are:

1. Try breaking up the compile into two or more smaller groups of files.
2. Change your overall program building strategy to use single-file compilation where a single program (.prg) file is compiled into a single object (.OBJ) file. To compile using single-file compilation mode, use the /M option.

C3026 Segment too big

Explanation: This error can occur if you are using automatic compilation to compile too many program (.prg) files into a single object (.OBJ) file at one time.

Action: Some suggestions to resolve the problem are:

1. Try breaking up the compile into two or more smaller groups of files.
2. Change your overall program building strategy to use single-file compilation where a single program (.prg) file is compiled into a single object (.OBJ) file. To compile using single-file compilation mode, use the /M option.

C3027 Proc too big

Explanation: This error can occur if you are using automatic compilation to compile too many program (.prg) files into a single object (.OBJ) file at one time.

Action: Some suggestions to resolve the problem are:

1. Try breaking up the compile into two or more smaller groups of files.
2. Change your overall program building strategy to use single-file compilation where a single program (.prg) file is compiled into a single object (.OBJ) file. To compile using single-file compilation mode, use the /M option.

C3028 Symbol table too big

Explanation: This error can occur if you are using automatic compilation to compile too many program (.prg) files into a single object (.OBJ) file at one time.

Action: Some suggestions to resolve the problem are:

1. Try breaking up the compile into two or more smaller groups of files.
2. Change your overall program building strategy to use single-file compilation where a single program (.prg) file is compiled into a single object (.OBJ) file. To compile using single-file compilation mode, use the /M option.

C3029 Write error to intermediate file

Explanation: The compiler created a temporary file to hold intermediate results and cannot write to the file. This error can occur if you run out of disk space during the compile.

Action: Some suggestions to resolve the problem are:

1. Free some disk space and recompile.
2. Direct the compiler to write its temporary files to another drive, where there is enough disk space, using the /T option.

C3030 Write error to OBJ

Explanation: The compiler created an object file but cannot write to the file. This error can occur if you run out of disk space during the compile.

Action: Some suggestions to resolve the problem are:

1. Free some disk space and recompile.
2. Direct the compiler to write the object (.OBJ) file to another drive using the /O option.

C3031 Can't create OBJ

Explanation: For some reason, the compiler cannot create an object file. This error can occur if you run out of directory entries or if you do not have create rights on a network drive.

Action: Some suggestions to resolve the problem are:

1. Check with your network administrator to see if you have create rights in the directory in which you are compiling.
2. Compile using the /O option to write object (.OBJ) files to another drive or directory.

C3032 Can't create intermediate file

Explanation: The compiler is attempting to create a temporary file to hold intermediate results and for some reason it cannot create the file. This error can occur if you run out of directory entries or if you do not have create rights on a network drive.

Action: Some suggestions to resolve the problem are:

1. Check with your network administrator to see if you have create rights in the directory in which you are compiling.
2. Compile using the /T option to write temporary files to another drive or directory.
3. Use DOS SET TMP= to a valid drive/directory.

Chapter 3
Protected Mode Linker Error Messages

Introduction

This chapter is a summary of VMM error messages and DOS/16M error messages for the protected mode linker. The messages are divided into categories on the basis of the point in the program at which they occur and the circumstances in which they occur.

VMM Error Messages

This section lists VMM error messages that can occur in the main program or when starting and running a VMM-managed program.

DOS/16M Error Messages

This section lists DOS/16M error messages, with descriptions of the circumstances in which the error is most likely to occur, and suggestions for remedying the problem. The chapter covers the following:

- DOS return codes
- Kernel error messages (single- or double-digit error numbers)
- Runtime Support error messages (four-digit numbers)
- CA-Clipper/Exospace messages (four-digit numbers)

VMM Error Messages

Loader Errors

VMM displays a loader error message only if the error occurs in your main program, not in a spawned program. When a loader error occurs while you are spawning a program, an error code is returned to the main program during the spawn. Error messages returned during the spawn process are displayed with only the last three digits of the error number. The loader error messages are listed below. Each message is preceded by the words "VM error [number]:" and the error code. The mnemonic appears in parentheses at the end of each error code description.

[5102] could not allocate transparent seg

Explanation: Your program was created with transparent selectors, and VMM could not allocate them.

[5105] no available VM memory

Explanation: The virtual space is too small.

[5108] too many spawned programs

Explanation: Your program is attempting to spawn beyond 15 levels.

[5110] VM loader error

Explanation: A loader error has occurred that cannot be identified as one of the errors above.

[5111] VM disk error

Explanation: A DOS error occurred while you were trying to load the program.

Startup and Runtime Messages

This section lists the messages you may receive when starting and running a VMM-managed program. For each message there is a brief explanation. Each error message is preceded by a "VM error [number]:" and the error code. Some of these messages indicate problems in VMM; in which case, call Computer Associates Technical Support.

[5004] can't allocate special seg

Explanation: There is probably not enough DOS memory available to allocate this segment.

Action: Increase SET Clipper = // LOWMEM parameter.

[5005] can't allocate VM info

Explanation: There is not enough physical memory to start VMM.

Action: Remove TSRs and system software. Install additional physical memory.

[5010] memory free

Explanation: A DOS error occurred when VMM was freeing memory at exit.

[5011] execute prog failure

Explanation: The application program cannot start. The startup code may be damaged, or the .EXE file may be corrupted.

[5251] insufficient memory

Explanation: VMM cannot find enough physical memory to load your program. You do not have 2Mb of *free* extended memory.

[5253] can't allocate VM tables

Explanation: There is not enough physical memory to allocate the swap file allocation table.

Action: Install more physical memory, or remove TSRs and system software.

[5254] insufficient DOS memory

Explanation: There is not enough low memory for your program.

Action: You may need to remove TSRs and system software, or reduce your program's dependency on transparent storage.

[5255] VM memory allocation error

Explanation: This indicates a DOS or DOS/16M error. The memory appears to be present, but VMM cannot allocate it.

[5257] can't create swap file: <filename>

Explanation: This indicates a DOS error. The path specified may not exist.

[5258] not enough disk space for swapping required bytes: #bytes

Explanation: There is not enough disk space to create the swap file. This error message displays the total number of bytes you need in the swap file.

Action: Delete files.

[5302] swap out

Explanation: VMM internal errors. There may be a problem with the disk, or the VMM internal tables may be corrupted.

Action: Call Computer Associates Technical Support.

[5303] incomplete swap out

Explanation: VMM internal errors. There may be a problem with the disk, or the VMM internal tables may be corrupted.

Action: Call Computer Associates Technical Support.

[5305] swap in

Explanation: VMM internal errors. There may be a problem with the disk, or the VMM internal tables may be corrupted.

Action: Call Computer Associates Technical Support.

[5306] incomplete swap in

Explanation: VMM internal errors. There may be a problem with the disk, or the VMM internal tables may be corrupted.

Action: Call Computer Associates Technical Support.

[5310] codefile seek

Explanation: DOS error occurred when VMM tried to retrieve a segment from the .EXE file. The file, or VMM internal tables, may be corrupted.

Action: Call Computer Associates Technical Support.

[5311] codefile read

Explanation: DOS error occurred when VMM tried to retrieve a segment from the .EXE file. The file, or VMM internal tables, may be corrupted.

Action: Call Computer Associates Technical Support.

[5312] moveseg

Explanation: VMM could not move a segment. There are discrepancies in VMM's internal tables.

Action: Call Computer Associates Technical Support.

[5313] moveseg 2

Explanation: VMM could not move a segment. There are discrepancies in VMM's internal tables.

Action: Call Computer Associates Technical Support .

[5314] vmFree

Explanation: VMM cannot free a segment. Memory may be corrupted.

Action: Call Computer Associates Technical Support.

[5315] user exit (swap out)

Explanation: A user-controlled interception of a segment swap out has generated an error of some sort.

Action: Call Computer Associates Technical Support.

[5316] user exit (segment fault)

Explanation: A user-controlled interception of a segment fault has generated an error of some sort.

Action: Call Computer Associates Technical Support.

[5317] insufficient virtual memory

Explanation: You have run out of virtual space.

Action: Increase your virtual space.

[5318] DOS memory error

Explanation: VMM received an error while trying to release or reallocate the DOS memory.

[5319] not enough disk space

Explanation: There is not enough disk space for the swap file to expand into.

[5391] out of memory

Explanation: VMM is not reserving enough virtual memory for your program.

Action: Increase the SET Clipper = //VMSIZE parameter.

DOS/16M Messages

This chapter lists DOS/16M error messages, with descriptions of the circumstances in which the error is most likely to occur, and suggestions for remedying the problem. The chapter covers the following:

- Link-time DOS return codes
- Runtime DOS return codes
- Kernel error messages (single- or double-digit error numbers)
- Runtime Support error messages (four-digit numbers)
- CA-Clipper/Exospace messages (four-digit numbers)

Link-Time DOS Return Codes

The following are the DOS ERRORLEVEL values returned by the CA-Clipper/Exospace linker if an error occurs while linking your program.

Link-Time Error Status

Link-Time Error Value	Link-Time Error Meaning	DOS Return Codes With NODELETE	DOS Return Codes Without NODELETE
0	Normal	0	0
1	Warning	0	0
2	Symbol error	0	2
3	Serious	3	3
4	Fatal	4	4
5	Internal error	5	5

Runtime DOS Return Codes

The following are the DOS ERRORLEVEL codes returned from the DOS/16M kernel if an error occurs while loading your program into memory. After your DOS/16M program begins execution, the code it specifies at exit will be returned instead.

0 Normal completion.

-1 General error.

2 System does not have a 286, 386 or 486 CPU.

3 Obsolete. If you get this error, call Computer Associates Technical Support.

4 A memory resident program has put your 386 or 486 CPU into Virtual 8086 mode. In this mode, DOS/16M cannot switch to protected mode unless the resident software follows the DPMI protocol or the VCPI protocol.

5 This computer has no protocol for managing extended memory. You are running on a non-AT-compatible MS-DOS machine that has no memory manager. Use the DOS/16M environment variable to set a range of memory for DOS/16M to use.

6 The DOS/16M environment variable has been set to an invalid value. You should only have to set the DOS/16M environment variable if you are running on a non-AT-compatible MS-DOS machine.

7 Cannot find the program .EXE file because you are running under DOS 2.x. DOS/16M requires DOS version 3.0 or later.

8 Cannot allocate transfer stack (not enough DOS memory).

9 Cannot allocate extended memory under VCPI (not enough available).

10 DOS/16M internal errors. If you get this error, call Computer Associates Technical Support.

11 Cannot allocate memory for the GDT.

Kernel Error Messages

This section describes error messages from the DOS/16M kernel. Kernel error messages may occur as the result of a problem in either the DOS/16M kernel or the application program. If the message originates in the kernel, it will be tagged as a DOS/16M error. If the error occurred in the application program, DOS/16M will use the name of the program file in the header. Messages that have a space immediately after the open quote occur only at load time. After your program is loaded, these messages are discarded to save memory. When a percent symbol (%) appears in the error message text, the string it represents is in parentheses after the error message text.

0: "Involuntary switch to real mode"

Explanation: The computer was in protected mode but switched to real mode without going through DOS/16M. This error most often occurs because of a stack segment exception (stack overflow), but can also occur if the Global Descriptor Table or Interrupt Descriptor Table is corrupted.

Action: Increase the stack size, recompile your program with stack overflow checking, or look into ways that the descriptor tables may have been overwritten.

1: "Insufficient memory (extended)"

Explanation: There is not enough extended memory to load your program.
Action: Check the memory to make sure that memory above 1MB is configured as extended (not expanded) memory, or use a VCPI host (such as QEMM or EMM386).

2: "%s is not a DOS/16M executable" (filename)

Explanation: The executable file you are trying to load has been corrupted in some way.

Action: Relink it or recopy it.

3: "No DOS memory for transparent segment"

Explanation: DOS has indicated that there is not enough memory available for the segment.

Action: Free some DOS memory.

4: "Unable to make transparent segment"

Explanation: DOS/16M cannot allocate a transparent segment. Most likely, your program is running under DPMI, where transparency is not available.

5: Obsolete

Explanation: This error should no longer occur.

Action: Call Computer Associates Technical Support.

6: "Insufficient memory to load program"

Explanation: The program that the DOS/16M loader is trying to load or overload is larger than available memory.

7: "Relocation tables required"

Explanation: Your program uses transparent segments, but the segment containing relocation information about them cannot be located.

8: "Cannot open file %s" (filename)

Explanation: The DOS/16M loader cannot open the executable file (file name). If you receive this message after executing the command loader file name, the file name is probably incorrect. Otherwise, DOS may not have enough file units configured (the FILES= entry in CONFIG.SYS should be larger).

9: Obsolete

Explanation: This error should no longer occur.

Action: Call Computer Associates Technical Support.

10: Obsolete

Explanation: This error should no longer occur.

Action: Call Computer Associates Technical Support.

11: Obsolete

Explanation: This error should no longer occur.

Action: Call Computer Associates Technical Support.

12: Obsolete

Explanation: This error should no longer occur.

Action: Call Computer Associates Technical Support.

13: Obsolete

Explanation: This error should no longer occur.

Action: Call Computer Associates Technical Support.

14: "Premature EOF"

Explanation: When loading your protected mode program into memory, the loader found the end of the file before it was expected. Most likely, the file has been corrupted, incompletely copied at some point, or incorrectly linked.

15: "Protected mode requires 80286 or superset"

Explanation: DOS/16M requires an 80286 CPU or a superset (such as a 386 or 486). It cannot run on an 8086, 8088, or 80186 CPU.

16: "Unable to run under OS/2"

Explanation: DOS/16M programs can run in an OS/2 2.x virtual DOS machine, but not as OS/2 applications.

17: "System software is neither VCPI nor DPMI"

Explanation: Some memory resident program has put your 386 or 486 CPU into Virtual 8086 mode. This is done to provide special memory services to DOS programs, such as EMS simulation (EMS interface without EMS hardware) or high memory. In this mode, it is not possible to switch into protected mode unless the resident software follows a standard that DOS/16M supports. VCPI (Virtual Control Program Interface) is one such standard and is followed by QEMM from Quarterdeck, 386MAX from Qualitas, and EMM386 from Microsoft, among others. DPMI (DOS Protected Mode Interface) is another standard that DOS/16M supports.

18: "Machine type requires ext memory range (SET DOS16M=)"

Explanation: This computer is not IBM AT-compatible, and has no protocol for managing extended memory. Add an extended memory range specification to the DOS/16M environment variable.

20: "Unsupported machine type %s (SET DOS16M=)" (machine_type)

Explanation: The value specified for the DOS/16M environment variable is one that DOS/16M cannot interpret. You should not get this message unless you have set DOS16M=D bad_machine_type.

Action: If you need the debugging switch (D), do not set the machine type on AT-compatible machines.

21: "Requires DOS 3.0 or later"

Explanation: DOS/16M programs do not run under DOS 2.x.

22: Obsolete

Action: This error should no longer occur. If you get it, call Computer Associates Technical Support.

23: Obsolete

Action: This error should no longer occur. If you get it, call Computer Associates Technical Support.

24: Obsolete

Action: This error should no longer occur. If you get it, call Computer Associates Technical Support.

25: "Unable to initialize VCPI"

Explanation: DOS/16M has detected that VCPI is present, but VCPI returns an error when DOS/16M tries to initialize the interface. The VCPI host is flawed.

26: "8042 timeout"

Explanation: On most IBM AT-compatible computers, DOS/16M must use the 8042 keyboard processor to enable access to extended memory. This message means that a command was sent to the 8042 to enable or disable access, but the 8042 has not acknowledged the command. Most likely, you are running on a non-AT-compatible machine.

28: "Free memory list corrupt"

Explanation: DOS/16M discovered an error in the memory allocation chains.

Action: Call Computer Associates Technical Support.

29: Obsolete

Action: This error should no longer occur. If it does, call Computer Associates Technical Support.

31: "Protected mode already in use in this DPMI virtual -machine"

Explanation: Due to limitations in the DPMI 0.9 specification, only one DOS/16M program at a time can execute in a DPMI virtual machine.

35: Various exceptions

Explanation: The processor has detected an exception or an unexpected interrupt. Error 35 is a fatal error. DOS/16M displays register values before it shuts down to help you debug your code.

General protection fault in <module> at <address> Exception 0Dh occurred.

Explanation: The program has violated one of the protected-mode programming restrictions.

Page Fault <linear address>

Explanation: Your program has accessed memory that has not been properly allocated.

Invalid Opcode Exception 06h occurred.

Explanation: Most likely, an assembly language program has attempted to execute data, or the module was compiled with the wrong instruction set.

Unexpected Interrupt <interrupt number>

Explanation: Your program generated an interrupt that DOS/16M could not manage in an appropriate way.

38: "HIMEM.SYS version < 2"

Explanation: DOS/16M works with versions 2.0 and later of HIMEM.SYS. The version on your machine is earlier than 2.0.

CA-Clipper/Exospace Linker Error Messages

This section lists and describes CA-Clipper/Exospace messages. The first section lists error messages in numeric order. The error level is enclosed in parentheses after the error text.

Error Messages

4001: Command line syntax error on argument <arg number> <"text"> (Fatal)

Explanation: CA-Clipper/Exospace did not understand an argument on the command line.

Action: Check the option list for correct syntax.

4002: Invalid library directory format (Fatal)

Explanation: CA-Clipper/Exospace cannot read your library. The library may be corrupted, or it was not built with a Microsoft-compatible librarian.

4003: Unable to process Microsoft Link command stream (Fatal)

Explanation: CA-Clipper/Exospace cannot interpret the Microsoft Link commands or script file.

Action: Make sure you used valid Microsoft syntax in the command line or script file.

4004: Don't know Microsoft Link command <name> (Serious)

Explanation: You have entered a Microsoft command line option that CA-Clipper/Exospace does not recognize.

4005: Bad small block allocation (Internal)

Explanation: An internal CA-Clipper/Exospace error.

Action: Call Computer Associates Technical Support.

4007: Can't find default library (Serious)

Explanation: The default library is not where CA-Clipper/Exospace expects to find it, or where it does not exist. CA-Clipper/Exospace looks for libraries in the current directory first, and then in the path set in the DOS environment variable LIB. The default libraries are those specified by the compiler through special records in the object file.

4008: Can't find indicated file (Serious)

Explanation: You have specified a file without adequate path information, or the file does not exist. For .OBJ files, CA-Clipper/Exospace searches the current directory first, then looks in the path set in the DOS environment variable OBJ.

4009: <Segment(text)>: Can't find named segment (Warning)

Explanation: CA-Clipper/Exospace cannot find the named segment or one of the segments that you specified for a group.

Action: Check for spelling errors or missing underscores.

4010: Unable to load library object <obj_name> (Fatal)

Explanation: The named object in the library has an unsupported format or is corrupted.

4013: Can't open indicated file. (Serious)

Explanation: CA-Clipper/Exospace cannot open a file you indicated. This error message appears most often if you are linking across a network that locks files.

4014: Unable to process file (Serious)

Explanation: The file can be found, but not opened.

Action: Make sure you have enough memory available to CA-Clipper/Exospace.

4025: Invalid record type <nn> at file offset <0xnnnn>. (Warning)

Explanation: Using the Microsoft object module format, CA-Clipper/Exospace has found a 2-digit hexadecimal record type that it does not recognize. The object module may be corrupted.

Action: Try recompiling the object.

4028: Insufficient memory. Program aborted. Try GLUV. (Fatal)

Explanation: CA-Clipper/Exospace does not have enough memory to link your program.

Action: Increase DPMI Memory.

4029: Multiple definition of <symbol_name> from file <filename> (Symbol definition)

Explanation: You have defined the same public symbol in two different object files. The name of the second file is shown in the error message. Your .MAP file tells you the name of the first file.

4030: Attempt to re-assign segment <segment_name> in group <group_name> to group <group_name> (Serious)

Explanation: You have assigned the same segment to two different groups.

4032: No specified files found to link. (Fatal)

Explanation: You have not specified any recognizable object modules to link.

4034: Not enough file handles to perform link (Fatal)

Explanation: CA-Clipper/Exospace reuses file handles after it reads a file into memory, starting with the least recently used, but you must have enough handles for all open output files, and three extra.

Action: Make sure that your CONFIG.SYS file specifies enough file handles.

4036: Can't open output file (Fatal)

Explanation: One of the output files specified in your link cannot be opened. The output file already exists and is write locked, or the drive is write protected. Possibly your network denied access to the drive.

4037: No more space for library symbol tables (Fatal)

Explanation: CA-Clipper/Exospace was denied a huge memory allocation it needed to read in the library directory.

4038: File read error. Position: <where>, Size: <read_length> (Fatal)

Explanation: Your file contains an error at the indicated position.

4039: Segment <segment_name> is greater than 64KB in length (Serious)

Explanation: The segment has grown beyond the 64KB limit for 16-bit executables. Most likely, the segment has contributions from more than one object module. _DATA is a likely candidate.

4040: Stack size override less than computed stack length (Warning)

Explanation: Your stack size override is less than the stack size CA-Clipper/Exospace estimates that you need. To arrive at its estimate, CA-Clipper/Exospace uses the sum of the lengths of all the stack segment definitions that it encounters during the link. CA-Clipper/Exospace honors your override.

4041: Syntax error, command line argument <argument> (Fatal)

Explanation: You have entered a command line argument incorrectly.

4042: Unknown file type (Fatal)

Explanation: CA-Clipper/Exospace opens each file listed on the command line and examines the first byte in an attempt to determine whether it is an object or library file. The first byte in one of your input files is not valid.

4045: Total: <number> unresolved symbols (Symbol definition)

Explanation: You have the displayed number of undefined symbols in your compiled modules. The symbol names are listed. This message normally appears during development, while you are still writing program modules.

4046: Unsupported fixup type found (Warning)

Explanation: CA-Clipper/Exospace has found a fixup type that it does not recognize. CA-Clipper/Exospace recognizes all Microsoft fixup types, but the Intel OMF specification defines additional fixup types that CA-Clipper/Exospace does not recognize.

4047: Current and previous file versions inconsistent (Serious)

Explanation: After a file has been read into memory, its handle may be reused if necessary. If the file is needed again, it is assigned another handle and read again. If the file has changed in the meantime, you see this message. It is most likely to occur if the file is on a network.

4049: Invalid seg record <segname> from module -<module_name> (Internal)

Explanation: The named module contains an invalid segment record.

Action: Call Computer Associates Technical Support.

4050: Error reading command script file (Serious)

Explanation: The command file you are using to run CA-Clipper/Exospace contains errors, or does not match the personality set in the CA-Clipper/Exospace environment variable.

4052: Unable to read <library_name>, invalid library format (Fatal)

Explanation: You have specified a file as a library, but it does not have the expected library format. Possibly the file is corrupted.

4053: Object file contains too many <class> records (Fatal)

Explanation: In a single object file, you have exceeded the 16,384 record limit for one class, such as public, external, or segment.

4054: Too many relocation selectors (Fatal)

Explanation: An internal CA-Clipper/Exospace error.

Action: Call Computer Associates Technical Support.

4055: Insufficient disk space. Program aborted (Fatal)

Explanation: The output files cannot be written because you have run out of disk space.

4056: Abort error level exceeded (Fatal)

Explanation: CA-Clipper/Exospace error level severity exceeded.

4057: Premature end of file encountered, file ignored (Serious)

Explanation: CA-Clipper/Exospace encountered the end of an object file before it encountered the MODEND record, which marks the logical end of an object file. CA-Clipper/Exospace ignores the invalid object file.

4058: Bad memory free operation (Internal)

Explanation: An internal CA-Clipper/Exospace error.

Action: Call Computer Associates Technical Support.

4060: Multiple starting addresses found. Address from -<module_name> used (Warning)

Explanation: You have linked two modules containing a starting address. CA-Clipper/Exospace ignores the second address. The modules are named so that you can correct the problem.

4061: No starting address found (Serious)

Explanation: Your program does not contain a module with a starting address.

Action: Make sure you have linked all library or object modules that you need.

4063: Bad symbol table vector (Fatal)

Explanation: An internal CA-Clipper/Exospace error.

Action: Call Computer Associates Technical Support.

4067: Out of memory. Last module processed = <modulename> (Serious)

Explanation: Any modules linked after this module will not have symbols. Symbols for modules are processed based on the order specified to the linker.

Action: Try compiling object modules without debugging information.

4080: Segment <name> already assigned to group <name> (Warning)

Explanation: You tried to assign a segment that is already in one group to another group.

4081: Segment <name> definition inconsistent with group <name>, grouping ignored (Warning)

Explanation: You tried to assign a segment of one access class to a group of another. For example, you may have tried to assign a code segment to a data group.

4085: FILE: <file> Line <line> Incomplete symbol type list (Serious)

Explanation: This indicates an internal CA-Clipper/Exospace error.

Action: Call Computer Associates Technical Support.

4087: Illegal index field in fixup (Serious)

Explanation: CA-Clipper/Exospace encountered a malformed or unsupported fixup in the object file. If you use a Borland compiler without the Vs switch, you will get this error.

Action: Recompile the object file; if the problem persists, call Computer Associates Technical Support.

4089: Can't read/write relocation temp file (Possibly insufficient disk space)

Explanation: CA-Clipper/Exospace is unable to read the temporary file it created for relocation information. Somehow, the file has been damaged.

Action: Set the TMP environment variable to a directory on a local drive.

4090: Can't open relocation file

Explanation: CA-Clipper/Exospace is unable to create a temporary file for relocation information. You either do not have enough memory; or, if you are on a network, you do not have write access to the current drive.

Action: Use the TMP environment variable to specify a directory for the temporary file.

4091: Conflicting alias of <symbol> from <file> (Symbol definition)

Explanation: CA-Clipper/Exospace encountered an attempt to use the same symbol as an alias for two different target symbols.

4092: Conflicting weak linkage of <symbol> from <file> (Symbol definition)

Explanation: CA-Clipper/Exospace encountered an attempt to define two default resolution symbols for the same target symbol.

Not a DOS/16M Executable

If you get the message "not a DOS/16M executable," your copy of the application may be corrupted. This could indicate that a virus may exist on your system. Use a virus scanning program to determine if there is a virus on your system; if there are no viruses, try relinking the application.

If you encounter this error on any of the CA-Clipper/Exospace distribution files, try reinstalling CA-Clipper/Exospace from the distribution diskettes.

Extended Memory Not Available

CA-Clipper/Exospace uses extended memory, not expanded memory. If you can configure your memory either way and you are not running under an expanded memory manager, be sure to configure the memory as extended memory.

If you are using an expanded memory manager that follows the VCPI protocol (such as EMM386, QEMM, or 386MAX), your CA-Clipper/Exospace application allocates its memory from the VCPI host (unless you also have an XMS manager). Therefore, you do not have to configure it as extended memory.

If you have a VCPI host and an XMS manager (such as HIMEM.SYS), CA-Clipper/Exospace allocates memory from both the VCPI pool and the XMS pool.

If you have another program that uses extended memory, see the Troubleshooting section in the "CA-Clipper Protected Mode Linker—EXOSPACE.EXE" chapter of the *Programming and Utilities Guide*.

Information Messages

As CA-Clipper/Exospace works, the messages listed below are displayed on the screen. They are not error messages. Many information messages are headings for lists of files or other data.

Reading object files and library headers.

Processing library directories.

Extracting library objects.

Assigning selectors...

Preparing GDT image.

Writing code image to .EXP file.

Sorting <number of>relocations.

Building relocation tables.

Writing GDT image to .EXP file.

Writing .MAP file.

Chapter 4
Blinker Error Messages

Introduction

The following sections contain a complete list of the Blinker error messages with a description of each error and reasons for their occurrence. The errors are divided into the following classes:

- 10xx - Link-time warning messages
- 11xx - Link-time fatal messages
- 12xx - DOS real mode runtime messages

Link-time warning messages are only for information. They do not set the DOS ERRORLEVEL. As a result, the execution of Blinker and any MAKE utilities being used will continue unaffected. These warning messages may be disabled altogether using the BLINKER MESSAGE NOWARNING command.

Link-time fatal messages and runtime messages, on the other hand, are of a more serious nature. They cause immediate termination of program execution, and set the DOS ERRORLEVEL to 1.

Link-Time Warning Messages

1001: ignoring invalid link script file command parameter in '<command>'

Explanation: This message indicates that the link script file command specified has been ignored as it contains an invalid parameter value.

See also: The Linker Reference Commands section in the "CA-Clipper Real Mode Linker—BLINKER.EXE" chapter of the *Programming and Utilities Guide.*

1002: duplicate input file <filename> ignored

Explanation: This message indicates that the same input file appeared in more than one FILE or LIBRARY command. The duplicated input file will be ignored.

See also: The Linker Reference Commands section in the "CA-Clipper Real Mode Linker—BLINKER.EXE" chapter of the *Programming and Utilities Guide.*

1003: warning - no stack segment in .EXE file

Explanation: This message indicates that the program .EXE file was created without a stack segment, which may cause the program to fail at runtime unless it is designed to set up its own stack, in which case this message may be ignored.

1004: <filename(modulename)> : '<symbol>' duplicated in <filename(modulename)>

Explanation: This message indicates that the name symbol defined as public in the first module is also defined as public in the second module. This will occur if two procedures in separate .OBJ or .LIB files have the same name. The first one found is the one which is linked into the program, and the second one is ignored.

1004: DEFINE command : '<symbol>' duplicated in <filename(modulename)>

Explanation: This message indicates that the name symbol defined in a DEFINE command, is also defined as public in the second module. This will occur if two procedures in separate .OBJ or .LIB files have the same name. The first one found is the one which is linked into the program, and the second one is ignored.

See also: The Linker Reference Commands section in the "CA-Clipper Real Mode Linker—BLINKER.EXE" chapter of the *Programming and Utilities Guide.*

1005: the default overlay class '<class>' contains the word 'DATA'

Explanation: This message indicates that the default overlay class, taken from the class of the first segment in the first .OBJ being linked, contains the word 'DATA'. Blinker currently overlays code only, so this message is displayed as a warning in case a non-standard .OBJ file is being linked as the first .OBJ file.

Action: If the class name displayed is not a code class, check that the first .OBJ being linked contains a valid code segment as the first segment, or place one of your own program .OBJ files as the first .OBJ file in the script file.

1007: warning - program has no start address

Explanation: This message indicates that no module containing a record defining the starting point of the program was found during the link. A module of this type is usually found in the CA-Clipper 5.3 default language library.

Action: Make sure you link CLIPPER.LIB into your program.

1011: '<symbol>' has been aliased to two different symbols

Explanation: This message indicates that the named symbol has been aliased to more than one different symbol name. The second alias symbol is ignored.

1017: Blinker for CA-Clipper 5.3 does not support the command '...'

Explanation: This message indicates that the link script file command has been ignored as it refers to a feature which is only available in the full product version of Blinker, not in Blinker for CA-Clipper 5.3.

See also: Refer to file BLINKER.TXT in the directory in which CA-Clipper is installed for details on upgrading to the full version of Blinker.

Link-Time Fatal Messages

1101: terminated by user

Explanation: This message indicates that the user interrupted Blinker while it was running by pressing the Ctrl+C or Ctrl+Break keys.

1102: no input files were specified

Explanation: This message indicates that no FILE commands were found on the Blinker command line or in any of the script files specified on the command line.

Action: Blinker requires at least one FILE command to indicate which object file(s) is to be linked, so specify a FILE command.

1103: '.....' is not a valid link script command

Explanation: This message indicates that the command shown is not a valid Blinker link script command.

Action: Make sure that the command is a valid Blinker link script command.

See also: The Linker Reference Commands section in the "CA-Clipper Real Mode Linker—BLINKER.EXE" chapter of the *Programming and Utilities Guide*.

1104: unbalanced BEGINAREAs and ENDAREAs

Explanation: This message indicates that a different number of BEGINAREA and ENDAREA commands was found in the script file(s), which implies an invalid overlay structure. Generally only one BEGINAREA and ENDAREA section is needed for Blinker's dynamic overlays.

Action: If an overlay structure is being carried over from a structured linker, this can be simplified by reducing the number of overlay sections.

1105: maximum script file nesting depth exceeded

Explanation: This message indicates that a script file name (@<filename>) was found inside another script file, which increased the total nesting level for script files to greater than the maximum permissible five levels.

Action: It may be possible to alleviate the problem by splitting the script files into two or more files specified one after the other on the Blinker command line. For example, BLINKER @script1 @script2.

1106: not enough real memory to link

Explanation: This message indicates that Blinker could not allocate enough real (DOS) memory to continue the link. Blinker has a link-time virtual memory system that can use EMS (3.2 and higher), XMS (2.0 and higher) memory (or disk space) when conventional DOS memory is fully allocated. However, Blinker still requires a certain amount of conventional DOS memory to complete the link cycle.

Action: To make more memory available, either remove one or more TSRs currently resident, run Blinker from outside a make utility, or reduce the number of buffers specified in the CONFIG.SYS. Reducing the number of SECTION INTO and SEARCH commands will also reduce the memory needed at link time.

See also: The Linker Reference Commands section in the "CA-Clipper Real Mode Linker—BLINKER.EXE" chapter of the *Programming and Utilities Guide.*

1107: disk full writing output file

Explanation: This message indicates that the destination disk is full. There was an error writing to the disk, possibly because write access has been denied or the output file name is invalid.

Action: If there is plenty of space available, then this message indicates that there may be a problem writing to the disk, which can be checked by attempting to copy some other files to the destination disk and path. An externally overlaid program will use less temporary disk space at link time than an internally overlaid program.

1108: stack overflow during linking

Explanation: This message indicates that the runtime stack for BLINKER.EXE has overflowed and should not be displayed under normal circumstances.

Action: If this message is displayed, reinstall the Blinker software. If the error persists, contact Computer Associates Technical Support.

1109: unable to open file <filename>

Explanation: This message indicates that Blinker was unable to find the specified file in the current directory where it was expected to be, so the full directory path for the file should be specified. If no file extension is specified for script files, Blinker assumes an extension of .LNK. This error can also occur when Blinker is attempting to create a swap file and the TEMP environment variable points to an invalid directory. When running on a network, it is also possible that another user deleted the file, or that you do not have network access rights to the file.

Action: If the file has been specified in the link script with a FILE command, check the OBJ environment variable; if it has been specified with a LIBRARY or SEARCH command, check the LIB environment variable.

1110: <file>: invalid object file or library file

Explanation: This message indicates that the file shown, which may be an .OBJ file or a .LIB file, contains an invalid object record. If the file is expected to be valid, it has probably become corrupt.

Action: Either recreate the file by recompiling, or reinstall the appropriate file from the installation disk. For a CA-Clipper application, place EXTEND.LIB after CLIPPER.LIB in the link script or do not mention these libraries, allowing them to be brought in by default in the correct order.

1111: <filename(modulename)> : invalid segment definition

Explanation: This message indicates that the file shown, which may be an .OBJ file or a .LIB file, contains an invalid object record. If the file is expected to be valid, it has probably become corrupt.

Action: Either recreate the file by recompiling, or reinstall the appropriate file from the installation disk. For a CA-Clipper application, place EXTEND.LIB after CLIPPER.LIB in the link script, or do not mention these libraries, allowing them to be brought in by default in the correct order.

1112: <filename(modulename)>: invalid group definition

Explanation: This message indicates that the file shown, which may be an .OBJ file or a .LIB file, contains an invalid object record. If the file is expected to be valid, then it has probably become corrupt.

Action: Either recreate the file by recompiling, or reinstall the appropriate file from the installation disk. For a CA-Clipper application, place EXTEND.LIB after CLIPPER.LIB in the link script, or do not mention these libraries, allowing them to be brought in by default in the correct order.

1113: segment <name> or its group exceeds 64kb

Explanation: This message indicates one of a number of situations:

1. Two related libraries may have been separated by other libraries in the link script file, so that the code in one library cannot reach the code in the other library. This may also occur when related .OBJ files, possibly extracted from the same library, have been separated by too many other .OBJ files in the script file.

2. Non-overlayable code may have been overlaid. When third-party products are linked into the program, refer to the third-party documentation for their suggested method for dynamic overlaying.
3. Two libraries may have declared the same segment names in different classes. This should not occur in most programs.
4. Two or more libraries may have been specified in an incorrect order within the link script.
5. The program's default data segment (usually named DGROUP in the program's MAP file) may have exceeded 64K in size.

   Action for Situation 5: If the default data segment exceeded 64K when using CA-Clipper, determine which third-party product is using the most of DGROUP by systematically removing each library and running the program with the //INFO parameter of the CLIPPER environment variable enabled. When the offending library is found, contact Computer Associates Technical Support.

Note: For CA-Clipper applications, the usual cause of this problem is that the first .OBJ file specified on the command line or in the script file is not a CA-Clipper .OBJ file. When linking a CA-Clipper application the first .OBJ file specified must be the CA-Clipper starting program.

If these approaches do not work, and recreating or reinstalling the specified file does not help, then contact Computer Associates Technical Support.

1114: <file(module)> : '<symbol>' duplicated in <file(module)>

Explanation: This message indicates that the name symbol defined as public in the first module is also defined as public in the second. This will occur if two procedures in separate .OBJ or .LIB files have the same name. The first one found is the one which is linked into the program, and the second one is ignored.

Action: One of the symbols must be given a different name and the appropriate .OBJ file recompiled.

1114: DEFINE command : '<symbol>' duplicated in <file(module)>

Explanation: This message indicates that the name symbol defined as public in a DEFINE command, is also defined as public in the second module. This will occur if two procedures in separate .OBJ or .LIB files have the same name. The first one found is the one which is linked into the program, and the second one is ignored.

Action: One of the symbols must be given a different name and the appropriate .OBJ file recompiled.

1115: <file(module)> : '<symbol>'unresolved external

Explanation: This message indicates that the '<symbol>' defined as external in the file module cannot be found in any of the other .OBJ files or libraries in the script file. The usual cause of this problem is that an object or library file required by the program has been omitted from the script file. Another possible cause of this message is that some libraries require the use of the SEARCH command, as their library cross-reference records are not complete.

Action: Try a SEARCH on the library that you suspect contains the missing '<symbol>'. If you are aware of the cause for this message and want to continue linking to create an executable file, use the command BLINKER EXECUTABLE NODELETE.

See Also: The Linker Reference Commands section in the "CA-Clipper Real Mode Linker—BLINKER.EXE" chapter of the *Programming and Utilities Guide.*

1116: fixup overflow at <offset> in segment in <module> referencing <symbol> in module <module>

Explanation: This message indicates the exact point in the code that refers to another symbol more than 64KB away. This may occur in one of a number of situations:

1. Two related libraries may have been separated by other libraries in the link script file, so that the code in one library cannot reach the code in the other library. This may also occur when related .OBJ files, possibly extracted from the same library, have been separated by too many other .OBJ files in the script file.
2. Non-overlayable code may have been overlaid. When third-party products are linked into the program, refer to the third-party documentation for their suggested method for dynamic overlaying.
3. Two libraries may have declared the same segment names in different classes. This should not occur in most programs.
4. Two or more libraries may have been specified in an incorrect order within the link script.

Action: Determine which library is causing the problem by systematically removing each library and relinking. When the offending library is found, contact Computer Associates Technical Support. If the offending library is not found, modify the code so that *<symbol>* is accessed with a far reference, or ensure that the specified segment and symbol are accessible to one another by examining the MAP file for their respective locations.

1118: too many external overlays

Explanation: This message indicates that more than 15 unique external overlay file names were specified in the link script file. This version of Blinker only supports up to 15 external overlays. It should be noted that with Blinker's dynamic overlaying, the only reason to have more than one external overlay file is so that each overlay file will fit onto a 360KB disk, if required for distribution purposes.

Action: Reduce the number of external overlay file names in the link script file.

See also: The Linker Reference Commands section in the "CA-Clipper Real Mode Linker—BLINKER.EXE" chapter of the *Programming and Utilities Guide.*

1119: no main module found

Explanation: This message indicates that no startup procedure was found in the files and libraries to be linked. It usually indicates that the language library containing the startup code has not been specified in the link script.

1121: this is not a CA-Clipper 5.3 application

Explanation: This message indicates that the application being linked is not a CA-Clipper 5.3 application. This special limited version of Blinker will only link CA-Clipper 5.3 applications.

1122: <file(module)> : compiled with Summer '87 not 5.x

Explanation: This message indicates that Blinker has determined that the specified module was compiled with CA-Clipper Summer '87, but that the first FILE in the application was compiled with CA-Clipper 5.3. All CA-Clipper compiled modules in a single application must be compiled with the same version of CA-Clipper.

Action: Ensure you are linking the correct versions of the CA libraries and any third-party libraries for the version of CA-Clipper that you are using, and also make sure all CA-Clipper compiled .OBJ files were compiled with the same version of CA-Clipper.

1124: <file(module)> : first .OBJ not compiled with CA-Clipper

Explanation: Within an application that contained a CA-Clipper module, the first module in a FILE command was not compiled with CA-Clipper. In a CA-Clipper application, the first FILE command must refer to the CA-Clipper starting program.

Action: If you receive this message, reorganize your link script so that the first FILE statement refers to a CA-Clipper compiled module.

1125: <file(module)> communal '<symbol>' has an inconsistent element size

Explanation: This message is applicable to C-compiled modules. Blinker has detected that the size of the far static uninitialized data item (communal variable) indicated, had been declared to have a different element size in a previously encountered module.

Action: If you receive this message, modify and recompile your C modules so that the communal variable is declared to be the same size in all the modules.

1126: <file(module)> communal '<symbol>' has an invalid data type

Explanation: This message is applicable to C-compiled modules. Blinker has detected that the type of the indicated communal variable is neither NEAR nor FAR. If you receive this message, it is likely that the indicated file has been corrupted.

Action: Recompile or reinstall the module.

1127: near communals segment exceeds 64kb

Explanation: The total size of the NEAR communal variables segment exceeds the maximum permissible size (64KB).

Action: To reduce the size of the segment change some NEAR uninitialized data declarations to FAR in the program source.

1128: <file(module)> : unrecognized .OBJ record type nnh

Explanation: Blinker encountered an object record type which is not currently supported. It is possible that the module in question is corrupted. If recreating or reinstalling the module does not solve the problem, contact Computer Associates Technical Support. Also, Blinker may have encountered a new type of object record; perhaps one which has been created by a new compiler version which is not supported by the version of Blinker being used.

Action: Refer to the file BLINKER.TXT in directory in which CA-Clipper is installed for details on upgrading to the full version of Blinker.

1130 : unable to create file <filename>

Explanation: Either there is already an existing file of the same name which has been assigned read-only status, or no create access has been granted to the destination drive or directory, or the destination disk is full.

Action: Make sure that the file name is different and that there is sufficient disk space.

1131 : error closing file <filename>

Explanation: It is likely that the file specified has been inadvertently deleted by another user on the system while Blinker was still using it, or there could be a network or disk failure.

Action: Make sure the file actually exists and that you have sufficient rights on the network. If you are not on a network, make sure your disk is free of errors.

1132: error deleting file <filename>

Explanation: Either the file specified is still in use, or no delete access has been granted, or the file exists in an alternate directory specified with the DOS APPEND command or network search command.

Action: Make sure the file is not marked read-only, is not in use, and does not appear elsewhere in the DOS APPEND or network search path.

1133: version n.n is required to use this .LNK file

Explanation: This message indicates that the .LNK file being used contains a request for a version of Blinker later than the version you are using.

Action: Make sure you have the correct version of Blinker.

1134: error reading file <filename>

Explanation: Blinker was unable to read from the specified file.

Action: Check to make sure the file is accessible via the appropriate PATH and other environment variables, or is specified with a full path in the link script. Alternatively, the file may be in use or corrupt.

1135: error writing file <filename>

Explanation: Blinker was unable to write to the specified file.

Action: Check available disk space, and ensure that an undeleteable file with the same name is not preventing Blinker from writing to the file.

1136: no more XMS handles available

Explanation: This message indicates that there were no more XMS handles available for Blinker's link-time virtual memory.

Action: Increase the number of handles provided by your XMS memory manager.

1137: no more EMS handles available

Explanation: This message indicates that there were no more EMS handles available for Blinker's link-time virtual memory.

Action: Increase the number of handles provided by your EMS memory manager.

1142: <filename(modulename)> : already belongs to group <group>

Explanation: This message indicates that a segment in the specified module has been defined as being in two distinct groups, which is not permitted.

1147: <filename(modulename)> contains incremental compilation errors

Explanation: This message indicates that the specified .OBJ file was created with a compiler which supports incremental compilation, but that the compilation failed.

Action: Correct and recompile the specified .OBJ file and then relink.

1150: <filename> is a demo so cannot be linked with this Blinker

Explanation: This message indicates that the specified .OBJ or .LIB file was created as a demonstration version. These files can only be linked by the demonstration version of Blinker.

Action: Use the demonstration version of Blinker supplied with that library, or the latest Blinker demonstration version, to link this application.

DOS Real Mode Runtime Error Messages

Blinker errors 1201 through 1204 can occasionally result from the effects of a virus. In their case, first try external overlays (via the SECTION INTO xxx command) marking the .EXE read-only, and then follow the procedures outlined below. If nothing else helps, perform an extensive check for viruses and take the necessary measures for removal.

1201: unable to find overlay file <filename> in current PATH

Explanation: This message indicates that either too many files are open or that the external overlay file <filename> specified and created at link time, can no longer be found in the current directory or the current PATH.

1202: DOS read error in file <filename>

Explanation: This message indicates that DOS returned an error while reading an overlay file. A number of situations can cause this error to occur frequently and consistently:

1. The most common is the use of a 'file handles' routine within the program to increase the maximum number of file handles available to the program under DOS versions 3.0 to 3.2. If this 'file handle' routine is called after Blinker has opened the overlay file and loaded one or more overlays from the file, the file handle table is recreated, wiping out Blinker's existing file handle for the overlays.

   Note: A combination of three factors can cause this to occur in a CA-Clipper application—Calling any handle table expanding routine from an overlay, using versions of DOS 3.3 or above, and specifying the CLIPPER Fn environment parameter. Since the handle-increasing routine is only needed for DOS versions 3.0 to 3.2, this error can be avoided by checking the DOS version before calling that routine.
2. This error can also occur when non-overlayable modules are placed in the overlay area.
3. Should the program stack overflow, it can overwrite the nearby file handle area, resulting in this error.
4. A network could be configured to deny access to the .EXE once it had been loaded, preventing Blinker's overlay manager from reading an internal overlay. Also swap functions can result in this error if the file handle table has not been restored correctly.

5. Finally, if this error occurs sporadically, it is possible that the overlay file has been closed by an external influence such as a TSR or being logged off a network, or the overlay file has become corrupt because of a disk error and should be reinstalled.

Action: Some suggestions to resolve this error are:

1. Ensure that the handles function is called from the root before Blinker has loaded any overlays.
2. If there are any .OBJ or .LIB files in the overlay area which are not known to be overlayable, these should be moved to the root and the program relinked. If the program then runs successfully, this indicates that the file(s) are not currently overlayable.
3. Try increasing the size of the stack with either the STACK or BLINKER PROCEDURE DEPTH command.

See also: The Linker Reference Commands section in the "CA-Clipper Real Mode Linker—BLINKER.EXE" chapter of the *Programming and Utilities Guide*.

1203: file <filename> is not a valid .EXE file

Explanation: This message indicates that the specified file does not have a valid .EXE file header so the Blinker internal overlay cannot be located within the file. This indicates that the file has become overwritten by another file, the PATH has changed so that an alternative file of the same name is found instead, or the .EXE file has become corrupt.

1204: overlay file <filename> does not match the .EXE file

Explanation: This message indicates that when the overlay manager opened the overlay file specified and created at link time, it was not a Blinker overlay file, or it was not prepared at the same time as the currently executing .EXE file. This indicates that the correct copy of the overlay file has become overwritten by an older version, the newer version cannot be found in the current PATH, or the previous link was terminated before the overlay file was created correctly.

1206: maximum procedure nesting depth exceeded

Explanation: This message indicates that during execution of the program the depth of nested procedures, i.e., the number of procedures calling one another without returning, has exceeded the maximum that may be handled on the procedure call stack.

Action: To increase the size of the procedure call stack simply use a BLINKER PROCEDURE DEPTH or a STACK command in the Blinker script file.

See also: The Linker Reference Commands section in the "CA-Clipper Real Mode Linker—BLINKER.EXE" chapter of the *Programming and Utilities Guide*.

1211: overlay manager stack overflow

Explanation: The Blinker overlay manager maintains a reload stack used in the management of dynamic overlays, and this stack has overflowed. The most common cause of this error is uncontrolled or unintentional recursion within an overlaid procedure or function, where a particular procedure or function is calling itself repeatedly.

Action: Increase your BLINKER PROCEDURE DEPTH when it is low, and check your code for unintentional recursion.

See also: The Linker Reference Commands section in the "CA-Clipper Real Mode Linker—BLINKER.EXE" chapter of the *Programming and Utilities Guide*.

1212: Overlay Opsize exceeded - increase Opsize

Explanation: This message applies only to the BLINKER OVERLAY FIXED overlay allocation method, and should not occur in normal operation. Normally, Blinker calculates the minimum required opsize at link time, and does not allow the opsize to be set below this minimum size (if you set the opsize to a value smaller than the minimum, Blinker will use the minimum instead). This message indicates that the Blinker overlay manager could not find space in the overlay area to load an overlay (the opsize was exceeded).

Action: Increase the overlay opsize by 1-2KB and relink the program. If you continue to receive this message, call Computer Associates Technical Support.

1213: attempt to call DEFINED routine

Explanation: The DEFINE link time command has been used to 'stub out' a particular routine, but a call to the routine has been made during execution of the program.

Action: Either remove all calls to the defined routine from the program, or remove the DEFINE command from your link script.

1214: error accessing EMS overlay cache

Explanation: This message indicates that an error has occurred while the Blinker overlay manager is accessing EMS for the storage and/or retrieval of overlays in the overlay cache.

Action: Disable the BLINKER CACHE EMS command in the link script file. If this fixes the problem, it is possible that you have a TSR or device driver working in the background which is manipulating EMS directly in a non-standard manner. Try removing all TSRs and device drivers from the system except for the EMS driver, and test if the problem still occurs. If it does, contact Computer Associates Technical Support.

See also: The Linker Reference Commands section in the "CA-Clipper Real Mode Linker—BLINKER.EXE" chapter of the *Programming and Utilities Guide*.

1215: error accessing XMS overlay cache

Explanation: This message indicates that an error has occurred while the Blinker overlay manager is accessing XMS for the storage and/or retrieval of overlays in the overlay cache.

Action: Disable the BLINKER CACHE XMS command in the link script file. If this fixes the problem, it is possible that you have a TSR or device driver working in the background which is manipulating XMS directly in a non-standard manner. Try removing all TSRs and device drivers from the system except for the XMS driver, and test if the problem still occurs. If it does, contact Computer Associates Technical Support.

See also: The Linker Reference Commands section in the "CA-Clipper Real Mode Linker—BLINKER.EXE" chapter of the *Programming and Utilities Guide*.

1217: overlay vector corrupted during execution

Explanation: This message indicates that the overlay manager has detected an invalid overlay vector. The most likely cause of this is corruption at runtime due to an invalid pointer, or an attempt to overlay a non-overlayable module. Another possible cause is the use of a debugger to place breakpoints in the code after an overlay call, since this replaces the vector data with a debugger breakpoint.

Action: Use of the MURPHY command for debugging overlays may help to identify whether non-overlayable modules are responsible for this message.

Approaches to Debugging

Usually link-time errors are well identified and the approach to their resolution is given in the individual error message descriptions in the above section. Some additional tools for discerning link-time problems are:

- The VERBOSE link script command, to see what Blinker is doing when an error occurs.
- The MAP command, to see how an .EXE has been constructed.
- The ECHO command, to output your own messages from within the link script.
- The command(s) BLINKER LINK EMS/XMS OFF, which disable the use of non-conventional memory.

Runtime error conditions that arise from the linking process can be more difficult to diagnose and resolve.

The two most common link-time mistakes which can lead to runtime errors are: overlaying something which should not be overlaid, and including the wrong module of two or more like-named modules. Both of these can lead to a variety of runtime symptoms which include, but are not limited to, corruption of variables or memory, inappropriate characters displayed to the terminal, a memory manager exception, and system lockups.

When encountering any such problems that are not attributable to the code itself or to other factors in the environment, the following steps are suggested in analyzing the linker's role in the problem:

1. Ensure that the program stack is sufficiently large using the STACK or BLINKER PROCEDURE DEPTH commands.
2. Verify that everything in the overlay area is reliably overlayable. Check the third-party documentation for their recommended method of dynamic overlaying. If the problem is reproducible, remove all suspected files and libraries from the overlay area and test the program after each is systematically reinserted.

3. Check for duplicate modules using the BLINKER MESSAGE DUPLICATES command. See the Linker Reference Commands section of the "CA-Clipper Real Mode Linker—BLINKER.EXE" chapter in the *Programming and Utilities Guide* for the definition of this command. Usually this situation can be resolved by altering the order of the libraries and, if necessary, using the MODULE FROM command.
4. Disable Blinker's runtime use of non-conventional memory either in the link script or via the BLINKER environment variable.
5. Perform a worst-case test of Blinker's overlaying process by using the MURPHY link-script command. For details on using this command, see the Linker Reference Commands section of the "CA-Clipper Real Mode Linker—BLINKER.EXE" chapter of the *Programming and Utilities Guide.*
6. Attempt to isolate a small portion of the program which produces the error.

Chapter 5
Runtime Errors

Runtime Recoverable Errors

Runtime recoverable errors can be expected to happen. These errors generally occur either because of mistakes in your code (e.g., type mismatch, or divide by zero) or because of some condition of the environment (e.g., out of file handles, file sharing violations, or memory low). These errors can be trapped in the error system and, therefore, do not necessarily terminate the application.

If the default error system is being used in the application, then runtime errors are reported in the following format:

```
Error | Warning <subSystem>/<subCode> <message text>
   <filename> | <operation>
```

Overview on Error Recovery

Error Recovery Failure

CA-Clipper's error system depends on communication taking place between the error handler and the subsystem that generates the error. The error handler communicates with the subsystem by returning a value indicating what the subsystem should attempt to do to recover from the error. The legal values that can be returned are determined by the values contained in the error object passed to the error handler for Error:canRetry, Error:canDefault, and Error:canSubstitute. If the error handler returns an invalid value to the subsystem (or returns to the subsystem at all when these values are all false), then an error recovery failure is reported and the application is terminated. This exit condition always has the same format:

```
Error recovery failure, <operation> (<line number>)
```

User Abort

The user can abort your application by pressing Alt-C or Ctrl-Break at anytime during the execution of your application unless you have specifically disabled this feature. You can disable it with SETCANCEL (.F.) or SET(_SET_CANCEL, .F.).

This exit condition always has the same format:

```
Cancelled at: <operation> (<line number>)
```

Missing Error Handler

If code is executed before any ERRORBLOCK() can be installed, an unrecoverable error will be generated that indicates that no error handler is present. This usually occurs if there is code in ErrorSys() before the ERRORBLOCK() function is called. All code should be moved after this line if possible:

```
No ERRORBLOCK() for error at: <operation>
   (<line number>)
```

Runtime Recoverable Error Categories

This section is a summary of runtime recoverable error messages that are possible when executing a CA-Clipper application using the supplied subsystems. The messages are divided into categories according to subsystem. Each category is described below, followed by a listing of all messages in each category.

BASE Errors

BASE error messages indicate errors generated by the Base system. The general format of a BASE error message is as follows:

```
Error | Warning BASE/xxxx <message text> <filename> |
   <operation>
```

TERM Errors

TERM error messages indicate errors generated by the Terminal subsystem. The general format of a TERM error message is as follows:

```
Error | Warning TERM/xxxx <message text> <filename> |
   <operation>
```

DBCMD Errors

DBCMD error messages occur in the database command set, and are unrelated to a particular driver. They occur as a result of command usage rather than from a failure of the driver itself.

```
Error | Warning DBCMD/xxxx <message text>
   <filename> | <operation>
```

DBFCDX Errors

DBFCDX error messages indicate that an error occurred during a database or index operation utilizing the DBFCDX database driver. The general format of a DBFCDX error message is as follows:

```
Error | Warning DBFCDX/xxxx <message text>
   <filename> | <operation>
```

DBFMDX Errors

DBFMDX error messages indicate that an error occurred during a database or index operation utilizing the DBFMDX database driver. The general format of a DBFMDX error message is as follows:

```
Error | Warning DBFMDX/xxxx <message text>
   <filename> | <operation>
```

DBFNDX Errors

DBFNDX error messages indicate that an error occurred during a database or index operation utilizing the DBFNDX database driver. The general format of a DBFNDX error message is as follows:

```
Error | Warning DBFNDX/xxxx <message text>
   <filename> | <operation>
```

DBFNTX Errors

DBFNTX error messages indicate that an error occurred during a database or index operation utilizing the DBFNTX database driver. The general format of a DBFNTX error message is as follows:

```
Error | Warning DBFNTX/xxxx <message text>
   <filename> | <operation>
```

Runtime Unrecoverable Errors

Unrecoverable errors are runtime errors that for some reason cannot make use of the error system. Like runtime errors, it is normal for these errors to occur. This is usually because the system is unable to execute the error block. Almost all of these errors are, therefore, related to the environment (e.g., out of memory, errors reading code to execute from disk) and can be fixed by making a change to the environment.

Unrecoverable errors always have the same format:

```
<operation> (<line number>) Unrecoverable error xxxx:
   <message text>
```

BASE Error Messages

BASE/1001 Undefined function

Explanation: You specified a function or procedure that:

1. You did not link into the current program.
2. You never directly referred to in the program.

Action: Make sure you linked the specified function or procedure. If you referred to the routine in an index key or macro expression, add a REQUEST statement to one of your program (.prg) files for each routine that was not linked.

If you are using RDDs that support tags and the error occurs when creating the index, DELETE the original index file, then recreate the index.

See Also: "Basic Concepts" chapter in the *Programming and Utilities Guide*

BASE/1002 Undefined alias

Explanation: You specified an alias not currently associated with any work area.

Action: Make sure the appropriate database file is open. Check to see if more than one work area uses the same alias at the same time. If so, designate a unique alias for each work area. The SELECT() function can be used to see if an alias is currently associated with any work area.

See Also: "Basic Concepts" chapter in the *Programming and Utilities Guide*

BASE/1003 Undefined variable

Explanation: You specified a variable that does not exist or is not visible.

Action: Some suggestions to resolve the problem are:

1. If you specified a database field, make sure you opened the appropriate database file and selected the appropriate work area. If you do not want to select another work area, preface the field variable reference with the target work area alias.
2. If you specified a private or public variable, make sure you created the variable using either a PRIVATE or PUBLIC statement.
3. If you specified a local or static variable reference within a macro variable, rearrange the code to refer to the variable directly (local and static variables are not visible within macro variables).

The /W compiler option is useful in finding misspelled or incorrectly specified variable references.

See Also: "Basic Concepts" chapter in the *Programming and Utilities Guide*

BASE/1004 No exported method

Explanation: This error can occur due to the following reasons:

1. You sent a message using the send operator (:), but the left operand was not a reference to an object, or the receiving object has no method with the specified name.
2. You called the EVAL() function and did not pass a code block as the first argument.
3. You specified a value other than a code block in a context where a code block was required.

Action: Some suggestions to resolve the problem are:

1. Check to make sure that the value being operated on is of the proper type or class.
2. If the error occurs on the send operator (:), make sure the message is one of the messages defined for the class.
3. If the missing method is "EVAL," make sure that code blocks have been supplied where required.

See Also: "Basic Concepts" chapter in the *Programming and Utilities Guide,* "Language Reference" chapter of the *Reference Guide*

BASE/1005 No exported variable

Explanation: You attempted to assign an exported instance variable using the send operator (:), but the left operand was not an object,. or the object has no exported variable with the specified name.

Action: Some suggestions to resolve the problem are:

1. Check to make sure that the value being operated on is of the proper type or class.
2. Make sure the name of the instance variable is spelled properly.

See Also: "Basic Concepts" chapter in the *Programming and Utilities Guide*, "Language Reference" chapter of the *Reference Guide*

BASE/1026 Index key expression required

Explanation: You attempted to use a function which requires an index key expression.

Action: Correct the program.

See Also: ORDCREATE()

BASE/1065 Argument error: &

Explanation: You specified an argument to the macro operator (&) that was not a character value.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1066 Argument error: conditional

Explanation: This error can occur for two reasons:

1. You specified an argument for a conditional statement such as IF or DO WHILE that did not return a logical value.
2. You specified the IF() function and the first argument did not return a logical value.

Action: Correct the program.

See Also: "Basic Concepts" chapter in the *Programming and Utilities Guide*

BASE/1067 Argument error: array dimension

Explanation: The value specifying the length of a new array was missing or non-numeric.

Action: Correct the program.

See Also: "Basic Concepts" chapter in the *Programming and Utilities Guide*

BASE/1068 Argument error: array access

Explanation: When attempting to retrieve the value of an array element, the subscript value was non-numeric, or the variable being subscripted was not an array.

Action: Correct the program.

See Also: "Basic Concepts" chapter in the *Programming and Utilities Guide*

BASE/1069 Argument error: array assign

Explanation: This error can occur for two reasons:

1. When assigning a new value to an array element, you specified a non-numeric subscript.
2. The variable you applied a subscript to was not an array.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1070 Argument error: ==

Explanation: The arguments to the == operator were of incompatible types.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1071 Argument error: =

Explanation: The arguments to the = operator were of incompatible types.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1072 Argument error: <>

Explanation: The arguments to the <>, #, or != operator were of incompatible types.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1073 Argument error: <

Explanation: The arguments to the < operator were of incompatible types.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1074 Argument error: <=

Explanation: The arguments to the <= operator were of incompatible types.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1075 Argument error: >

Explanation: The arguments to the > operator were of incompatible types.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1076 Argument error: >=

Explanation: The arguments to the >= operator were of incompatible types.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1077 Argument error: .NOT.

Explanation: The argument to the .NOT. or ! operator was not a logical value.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1078 Argument error: .AND.

Explanation: One or both of the arguments to the .AND. operator were not logical values.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1079 Argument error: .OR.

Explanation: One or both of the arguments to the .OR. operator were not logical values.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1080 Argument error: -

Explanation: The argument to the unary minus operator (-) was a non-numeric value.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1081 Argument error: +

Explanation: The arguments to the + operator were of incompatible types.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1082 Argument error: -

Explanation: The arguments to the - operator were of incompatible types.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1083 Argument error: *

Explanation: One or both of the arguments to the * operator were non-numeric values.

Action: Correct the program.

See Also "Language Reference" chapter in the *Reference Guide*

BASE/1084 Argument error: /

Explanation: One or both of the arguments to the / operator were non-numeric values.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1085 Argument error: %

Explanation: One or both of the arguments to the % operator were non-numeric values.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1086 Argument error: ++

Explanation: The ++ operator was applied to a variable or array element whose value was non-numeric.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1087 Argument error: --

Explanation: The -- operator was applied to a variable or array element whose value was non-numeric.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1088 Argument error: ^^

Explanation: One or both of the arguments to the ^^ operator were non-numeric values.

Action: Correct the program.

See Also: "Language Reference" chapter in the *Reference Guide*

BASE/1089 Argument error: ABS

Explanation: The argument to ABS() was non-numeric.

Action: Correct the program.

See Also: ABS() function

BASE/1090 Argument error: INT

Explanation: The argument to INT() was non-numeric.

Action: Correct the program.

See Also: INT() function

BASE/1091 Argument error: WORD

Explanation: The argument to WORD() was non-numeric.

Action: Correct the program.

See Also: WORD() function

BASE/1092 Argument error: MIN

Explanation: The arguments to MIN() were of incompatible types.

Action: Correct the program.

See Also: MIN() function

BASE/1093 Argument error: MAX

Explanation: The arguments to MAX() were of incompatible types.

Action: Correct the program.

See Also: MAX() function

BASE/1094 Argument error: ROUND

Explanation: One or more of the arguments to ROUND() was non-numeric.

Action: Correct the program.

See Also: ROUND() function

BASE/1095 Argument error: LOG

Explanation: The argument to LOG() was non-numeric.

Action: Correct the program.

See Also: LOG() function

BASE/1096 Argument error: EXP

Explanation: The argument to EXP() was non-numeric.

Action: Correct the program.

See Also: EXP() function

BASE/1097 Argument error: SQRT

Explanation: The argument to SQRT() was non-numeric.

Action: Correct the program.

See Also: SQRT() function

BASE/1098 Argument error: VAL

Explanation: The argument to VAL() was not a character value.

Action: Correct the program.

See Also: VAL() function

BASE/1099 Argument error: STR

Explanation: One or more of the arguments to STR() was of the wrong type.

Action: Correct the program.

See Also: STR() function

BASE/1100 Argument error: TRIM

Explanation: The argument to TRIM() was not a character value.

Action: Correct the program.

See Also: RTRIM() function

BASE/1101 Argument error: LTRIM

Explanation: The argument to LTRIM() was not a character value.

Action: Correct the program.

See Also: LTRIM() function

BASE/1102 Argument error: UPPER

Explanation: The argument to UPPER() was not a character value.

Action: Correct the program.

See Also: UPPER() function

BASE/1103 Argument error: LOWER

Explanation: The argument to LOWER() was not a character value.

Action: Correct the program.

See Also: LOWER() function

BASE/1104 Argument error: CHR

Explanation: The argument to CHR() was non-numeric.

Action: Correct the program.

See Also: CHR() function

BASE/1105 Argument error: SPACE

Explanation: The argument to SPACE() was non-numeric.

Action: Correct the program.

See Also: SPACE() function

BASE/1106 Argument error: REPLICATE

Explanation: One or more of the arguments to REPLICATE() was of the wrong type.

Action: Correct the program.

See Also: REPLICATE() function

BASE/1107 Argument error: ASC

Explanation: The argument to ASC() was not a character value.

Action: Correct the program.

See Also: ASC() function

BASE/1108 Argument error: AT

Explanation: One or more of the arguments to AT() was not a character value.

Action: Correct the program.

See Also: AT() function

BASE/1109 Argument error: $

Explanation: One of the arguments to the $ operator was not a character value.

Action: Correct the program.

See Also: "Basic Concepts" chapter of the *Programming and Utilities Guide*

BASE/1110 Argument error: SUBSTR

Explanation: One or more of the arguments to SUBSTR() was of the wrong type.

Action: Correct the program.

See Also: SUBSTR() function

BASE/1111 Argument error: LEN

Explanation: The argument to LEN() was not a character or array value.

Action: Correct the program.

See Also: LEN() function

BASE/1112 Argument error: YEAR

Explanation: The argument to YEAR() was not a date value.

Action: Correct the program.

See Also: YEAR() function

BASE/1113 Argument error: MONTH

Explanation: The argument to MONTH() was not a date value.

Action: Correct the program.

See Also: MONTH() function

BASE/1114 Argument error: DAY

Explanation: The argument to DAY() was not a date value.

Action: Correct the program.

See Also: DAY() function

BASE/1115 Argument error: DOW

Explanation: The argument to DOW() was not a date value.

Action: Correct the program.

See Also: DOW() function

BASE/1116 Argument error: CMONTH

Explanation: The argument to CMONTH() was not a date value.

Action: Correct the program.

See Also: CMONTH() function

BASE/1117 Argument error: CDOW

Explanation: The argument to CDOW() was not a date value.

Action: Correct the program.

See Also: CDOW() function

BASE/1118 Argument error: DTOC

Explanation: The argument to DTOC() was not a date value.

Action: Correct the program.

See Also: DTOC() function

BASE/1119 Argument error: CTOD

Explanation: The argument to CTOD() was not a character value.

Action: Correct the program.

See Also: CTOD() function

BASE/1120 Argument error: DTOS

Explanation: The argument to DTOS() was not a date value.

Action: Correct the program.

See Also: DTOS() function

BASE/1121 Argument error: TYPE

Explanation: The argument to TYPE() was not a character value.

Action: Correct the program. The expression supplied to TYPE() must be in textual form.

See Also: TYPE() function

BASE/1122 Argument error: TRANSFORM

Explanation: The second argument to TRANSFORM() was not a character value.

Action: Correct the program.

See Also: TRANSFORM() function

BASE/1123 Argument error: AADD

Explanation: The first argument to AADD() was not an array value.

Action: Correct the program.

See Also: AADD() function

BASE/1124 Argument error: LEFT

Explanation: One or more of the arguments to LEFT() was of the wrong type.

Action: Correct the program.

See Also: LEFT() function

BASE/1126 Argument Error: DATEFORMAT

Explanation: The date format was not legal.

Action: Correct the program.

See Also: SET()

BASE/1131 Bound error: array dimension

Explanation: An attempt was made to create an array containing more than the maximum number of elements.

Action: Correct the program. CA-Clipper allows a maximum of 4096 elements in an array.

See Also: "Basic Concepts" chapter of the *Programming and Utilities Guide*

BASE/1132 Bound error: array access

Explanation: When attempting to retrieve the value of an array element, the subscript specified was greater than the number of elements in the array.

Action: Correct the program. The LEN() function can be used to determine the number of elements in an array. The AADD() and ASIZE() functions can be used to change the size of an array.

See Also: "Basic Concepts" chapter of the *Programming and Utilities Guide*

BASE/1133 Bound error: array assign

Explanation: When attempting to assign a new value to an array element, the subscript specified was greater than the number of elements in the array.

Action: Correct the program. The LEN() function can be used to determine the number of elements in an array. The AADD() and ASIZE() functions can be used to change the size of an array.

See Also: "Basic Concepts" chapter of the *Programming and Utilities Guide*

BASE/1187 Bound error: AADD

Explanation: The array passed to AADD() already contained the maximum number of elements.

Action: Correct the program. The maximum number of elements in an array is 4096.

See Also: "Basic Concepts" chapter of the *Programming and Utilities Guide*

BASE/1209 String overflow: +

Explanation: An attempt was made to concatenate two character values whose combined length exceeded the maximum length.

Action: Correct the program. The maximum length of a character value in CA-Clipper is 65519 characters.

BASE/1210 String overflow: -

Explanation: An attempt was made to concatenate two character values whose combined length exceeded the maximum length.

Action: Correct the program. The maximum length of a character value in CA-Clipper is 65519 characters.

BASE/1233 String overflow: SPACE

Explanation: The argument to SPACE() specified the creation of a character value whose length exceeded the maximum length.

Action: Correct the program. The maximum length of a character value in CA-Clipper is 65519 characters.

BASE/1234 String overflow: REPLICATE

Explanation: The arguments to REPLICATE() specified the creation of a character value whose length exceeded the maximum length.

Action: Correct the program. The maximum length of a character value in CA-Clipper is 65519 characters.

BASE/1340 Zero divisor: /

Explanation: The right operand of the division operator (/) was zero.

Action: Correct the program. The default CA-Clipper error handler (Errorsys.prg) returns a zero result for division by zero.

BASE/1341 Zero divisor: %

Explanation: The right operand of the modulus operator (%) was zero.

Action: Correct the program. The default CA-Clipper error handler (Errorsys.prg) returns a zero result for division by zero.

BASE/1449 Syntax error: &

Explanation: The text string supplied to the macro operator (&) was not a valid CA-Clipper expression.

Action: Correct the program.

See Also: "Basic Concepts" chapter of the *Programming and Utilities Guide*

BASE/1513 Operation too complex: &

Explanation: The text string supplied to the macro operator (&) was too complex to be parsed by the macro parser.

Action: Simplify the expression or break it into several expressions and apply the macro operator to each expression separately. For filter and relational expressions, you may want to use a function to perform the operation.

See Also: "Basic Concepts" chapter of the *Programming and Utilities Guide*

BASE/2005 Open error (RESTORE command)

Default Behavior: The program will continue without restoring memory variables from the specified file. The default CA-Clipper error handler (Errorsys.prg) will set NETERR() to true (.T.) if the error was a sharing violation.

Explanation: The file referred to in a RESTORE command could not be opened.

Action: Check to make sure that the specified file exists. If the file is not in the directory where the application runs, use SET DEFAULT or SET PATH to make the file accessible, or specify the full path name where the file can be found. In a network environment, make sure the application has the necessary rights to access the file.

See Also: RESTORE command

BASE/2006 Create error (SAVE command)

Default Behavior: The program continues without saving memory variables to a file.

Explanation: The file referred to in a SAVE command could not be created.

Action: Check to make sure that sufficient disk space and directory entries are available. If the file exists, make sure it is not marked read-only. In a network environment, make sure the application has the necessary rights to create the file.

See Also: SAVE command

BASE/2011 Open error (TYPE command)

Default Behavior: The program will continue without TYPEing the file.

Explanation: The file referred to in a TYPE <*file*> command could not be opened.

Action: Check to make sure that the specified file exists. If the file is not in the directory where the application runs, use SET DEFAULT or SET PATH to make the file accessible, or specify the full path name where the file can be found. In a network environment, make sure the application has the necessary rights to access the file.

See Also: TYPE command

BASE/2012 Open error (COPY FILE command)

Default Behavior: The program will continue without copying the file.

Explanation: In a COPY FILE command, the source file could not be opened.

Action: Check to make sure that the file exists. If the file is not in the directory where the application runs, use SET DEFAULT or SET PATH to make the file accessible, or specify the full path name where the file can be found.

See Also: COPY FILE command

BASE/2016 Write error (COPY FILE command)

Default Behavior: The program continues without copying the file and the partially copied file is not erased.

Explanation: In a COPY FILE command, the source file could not be written to the destination, usually because of a full disk or insufficient rights in a network environment.

Action: Make sure there is sufficient space on the destination drive, or check network rights.

See Also: COPY FILE command

BASE/2017 Argument error (AEVAL() function)

Explanation: One or more of the arguments to AEVAL() was of the wrong type. This error can also occur when any of the required arguments are omitted.

Action: Correct the program.

BASE/2018 Open error (DISKSPACE() function)

Default Behavior: The program continues after returning an empty value.

Explanation: The disk does not exist or is not ready upon an attempt to access it with the DISKSPACE() function.

Action: Make sure to supply the correct drive number (for fixed disks) or call the DISKSPACE() function in a loop with a user warning when checking the readiness of a floppy drive.

See Also: COPY FILE command

BASE/2020 Argument error (SET() function)

Explanation: This error occurs if a negative numeric value is passed to SET() during an attempt to set the following global state variables:

```
_SET_DECIMALS    // SET DECIMALS TO xxx
_SET_EPOCH       // SET EPOCH TO xxxx
_SET_MARGIN      // SET MARGIN TO xx
_SET_MESSAGE     // SET MESSAGE TO xx
```

Action: Change your code. Do not pass negative values to SET().

See Also: COPY FILE command

BASE/2022 Argument error: ALLTRIM

Explanation: Argument to ALLTRIM() was not a character value.

Action: Correct the program.

BASE/5300 Memory low

Default Behavior: Continue with program execution. Default behavior is only available when Error:severity is set to ES_WARNING.

Explanation: The application has insufficient conventional memory. If the condition is a warning (severity 1), execution can continue but an unrecoverable error may occur without further warning. If the condition is a recoverable error (severity 2), execution cannot continue.

Action: For safety, quit the application. Make more memory available before running the application again. Remove unneeded resident utilities or device drivers. Opening fewer database or index files may make more memory available. This error may occur when manipulating large browse objects or character values, indicating that there is insufficient memory to swap in the object or character value. If many C or Assembler add-on functions are linked, overlaying some of these functions may free up memory for other uses.

See Also: For information on overlaying C and Assembler code, refer to the linker chapter in the *Programming and Utilities Guide*.

TERM Error Messages

TERM/0 Print error

Explanation: A write error has occurred on the print device or file.

Action: Check to make sure the printer is connected and online. If printer output has been redirected to a file, make sure that sufficient disk space is available. If printing to a network device, make sure that the network connection is valid and that a time-out has not occurred.

See Also: "Basic Concepts" and "Network Programming" chapters in the *Programming and Utilities Guide*

TERM/2013 Create error (SET ALTERNATE command)

Default Behavior: The program will continue without opening the file specified in the SET ALTERNATE command. If there was a file open from a previous SET ALTERNATE command, it will be closed.

Explanation: The file referred to in a SET ALTERNATE command could not be created or opened for writing.

Action: Check to make sure that sufficient disk space and directory entries are available. If the file exists, make sure it is not marked read-only. In a network environment, make sure the application has the necessary rights to create or modify the file.

See Also: SET ALTERNATE command

TERM/2014 Create error (SET PRINTER command)

Default Behavior: The program will continue without opening the file or device specified in the SET PRINTER command. If there was a file or device open from a previous SET PRINTER command, it will be closed.

Explanation: The file or device referred to in a SET PRINTER command could not be created or opened for writing.

Action: If opening a file, check to make sure that sufficient disk space and directory entries are available. If the file exists, make sure it is not marked read-only. In a network environment, make sure the application has the necessary rights to create or modify the file. If printing to a network device or queue, make sure that the designated device exists and is accessible to the workstation.

See Also: SET ALTERNATE command, "Network Programming" chapter in the *Programming and Utilities Guide*

TERM/2015 Open error (TO FILE clause)

Default Behavior: The program will continue without opening the file specified in the TO FILE clause, and the command will operate as if the TO FILE clause had not been specified.

Explanation: The file referred to in a TO FILE clause of the REPORT FORM, LABEL FORM or other command could not be created or opened for writing.

Action: Check to make sure that sufficient disk space and directory entries are available. If the file exists, make sure it is not marked read-only. In a network environment, make sure the application has the necessary rights to create or modify the file.

See Also: REPORT FORM command, LABEL FORM command

TERM/2100 Write error: <SET ALTERNATE TO file name>

TERM/2101 Write error: <SET PRINTER TO file name>

TERM/2102 Write error: <_SET_EXTRAFILE file name>

Explanation: A write error occurred on the specified file.

Action: Check to make sure that sufficient disk space and directory entries are available. In a network environment, make sure the application has the necessary rights to write to the file.

DBCMD Error Messages

DBCMD/1001 Argument error: SEEK or DBSEEK

Explanation: The argument to the SEEK or DBSEEK() was non-numeric, logical, character, or date.

Action: Correct the program.

See Also: DBSEEK() function, SEEK command

DBCMD/1003 Argument error: GOTO or DBGOTO

Explanation: Argument to GOTO or DBGOTO() was non-numeric.

Action: Correct the program.

See Also: DBGOTO() function, GOTO command

DBCMD/1004 Argument error: SET RELATION TO or DBSETRELAT

Explanation: One or more of the arguments to SET RELATION TO or DBSETRELAT() was of the wrong type, or the alias was invalid.

Action: Correct the program.

See Also: DBSETRELAT() function, SET RELATION TO command

DBCMD/1005 Argument error: USE or DBUSEAREA

Explanation: One or more of the arguments to USE or DBUSEAREA() was of the wrong type.

Action: Correct the program.

See Also: DBUSEAREA() function, USE command

DBCMD/1006 Argument error: INDEX ON or DBCREATEINDEX

Explanation: One or more of the arguments to INDEX ON or DBCREATEINDEX() was of the wrong type.

Action: Correct the program.

See Also: DBCREATEINDEX() function, INDEX ON command

DBCMD/1007 Argument error: SET ORDER TO or DBSETORDER

Explanation: The argument to SET ORDER TO or DBSETORDER() was non-numeric.

Action: Correct the program.

See Also: DBSETORDER() function, SET ORDER command

DBCMD/1008 Argument error: SET INDEX TO or DBSETINDEX

Explanation: The argument to SET INDEX TO or DBSETINDEX() was not a character value.

Action: Correct the program.

See Also: DBSETINDEX() function, SET INDEX TO command

DBCMD/1009 Argument error: FIELDNAME

Explanation: The argument FIELDNAME() was non-numeric.

Action: Correct the program.

See Also: FIELDNAME() function

DBCMD/1010 Illegal characters in alias

Explanation: An attempt was made to create an alias that was not a valid identifier. This occurs if the system is allowed to create a default alias based on a file name that is not a valid CA-Clipper identifier (e.g., Test$.dbf, 123file.dbf).

Action: If created due to the default alias, supply a valid identifier with the ALIAS clause.

DBCMD/1011 Alias already in use

Explanation: An attempt was made to create an alias that is already in use in another work area. The error is caused by:

```
USE Test NEW
USE Test NEW  // error occurs here
```

Action: If created due to the default alias, supply a valid identifier with the ALIAS clause.

DBCMD/1013 Argument error

Explanation: A database command was not used correctly.

Action: Some suggestions to resolve the problem are:

1. Check to make sure that the DBCMD is being used correctly.
2. Check that the RDD is properly linked into the application.
3. Compile with the /W option. Resolve all warning messages.

DBCMD/1014 Invalid argument for DBCREATE()

Explanation: This error occurs when the array for DBCREATE() has an invalid array element.

Action: Some suggestions to resolve the problem are:

1. Make sure that all array elements for the DBCREATE() function contain valid data.
2. Check for an additional comma after the fourth array element, as that will cause this error.

DBCMD/1015 Argument Error

Explanation: You attempted to select a linked-in driver with USE VIA but failed to REQUEST it.

Action: Correct the program.

DBCMD/1021 Argument error: ORDBAGNAME

Explanation: Argument to ORDBAGNAME() was not a character value.

Action: Correct the program.

DBCMD/1022 Argument error: ORDFOR

Explanation: One or more of the arguments to ORDFOR() was of the wrong type.

Action: Correct the program.

DBCMD/1023 Argument error: ORDKEY

Explanation: One or more of the arguments to ORDKEY() was of the wrong type.

Action: Correct the program.

DBCMD/1024 Argument error: ORDNAME

Explanation: One or more of the arguments to ORDNAME() was of the wrong type.

Action: Correct the program.

DBCMD/1025 Argument error: ORDNUMBER

Explanation: One or more of the arguments to ORDNUMBER() was of the wrong type.

Action: Correct the program.

DBCMD/1031 Argument error: DBFIELDINFO

Explanation: One or more of the arguments to DBFIELDINFO() was of the wrong type.

Action: Correct the program.

DBCMD/1032 Argument error: DBRECORDINFO

Explanation: One or more of the arguments to DBRECORDINFO() was of the wrong type.

Action: Correct the program.

DBCMD/1033 Argument error: DBORDERINFO

Explanation: One or more of the arguments to DBORDERINFO() was of the wrong type.

Action: Correct the program.

DBCMD/1034 Argument error: DBINFO

Explanation: One or more of the arguments to DBINFO() was of the wrong type.

Action: Correct the program.

DBCMD/1101 Syntax error

Explanation: This error occurs when the FIELDS clause of the TOTAL command includes an alias that does not point to the current work area:

```
TOTAL ON x TO y FIELDS badalias->f1
```

Action: With TOTAL, make sure to use an alias that refers to the current work area (if you use an alias).

DBCMD/2001 Work area not in use

Explanation: An attempt to execute a command or function that requires the use of a database failed because there was no database open in the specified work area.

Action: Correct the program. If the command is prefixed by an alias, verify that the alias is spelled correctly.

DBCMD/2019 Argument error

Explanation: One or more of the arguments to DBEVAL() was of the wrong type. This error can also occur when any of the required arguments are omitted.

Action: Correct the program. Pass a valid data type to DBEVAL().

See Also: DBEVAL() function

DBFCDX Error Messages

DBFCDX/602 Index corruption detected

Explanation: The RDD has found the index to be corrupted.

Action: Erase the index file and recreate a new index file.

DBFCDX/610 Argument error

Explanation: The index may be corrupted.

Action: Make sure that you did not create the index using a variable as in:

```
Local cDbfName := "MyDbf"
USE ( cDbfName ) NEW
INDEX ON ( cDbfName )->field TAG field TO Myindex.cdx
USE
USE ( cDbfName ) NEW
```

This will cause the DBFCDX/610 error when the database (.dbf) file is reopened using an index created like this. Also, this is very bad coding.

DBFCDX/611 Index key too long

Explanation: The index key expression is too long.

Action: Shorten the index key expression.

DBFCDX/612 Invalid key length

Explanation: This error is caused when the stored index key length and the actual length, when evaluated, are of different lengths.

Action: Some suggestions to resolve the problem are:

1. Basically, make sure that the key expression will produce a value of the same length for all records.
2. Avoid using functions like TRIM() in the key expression. If you need to restrict the length of key values, use functions like LEFT() or PADR().
3. If you are using a user-defined function (UDF), make sure that the same function is available in all applications that use the index, and that the function returns a value of the same length for all records.

DBFCDX/631 Non-compact index not supported

Explanation: The .cdx file was found to be a NON-COMPACT .cdx file.

Action: Check to see if the file was created under FOXPRO v1.0. CDX RDD only supports COMPACT version of .cdx files.

DBFCDX/650 Can't find current key in index

Explanation: The index key was not found.

Action: Try recreating the index file. Check that the index key expression is valid.

DBFCDX/1012 Corruption detected

Explanation: The .dbf file header is corrupted. This could be due to the improper use of the RDD, or there may be MEMO fields associated with the .dbf file that were not changed to .fpt format.

Action: Recreate the .dbf file header structure. Properly link in the correct RDD, or perform a COPY TO....VIA "DBFCDX" to convert the .dbt memo file format to an .fpt memo file format.

DBFCDX/1028 Create error

Explanation: The maximum number of orders per order bag was exceeded. The allowable number of orders that an order bag can contain varies by database driver.

Action: Cut down the number of tags being used.

See Also: RDD Features in the "Replaceable Database Driver Architecture" chapter of the *Driver's Guide*.

DBFCDX/1050 Invalid TAG/ORDER

Explanation: You tried to set the tag order of the order bag to a non-existing tag or an illegal (blank) tag name.

Action: Make sure that ORDSETFOCUS() is setting the tag order to a valid tag.

DBFCDX/1051 Scope Type Mismatch

Explanation: An attempt was made to use an incorrect type for a scope. For example, if you have a numeric key and attempt to use a date for a scope, you will get this error.

Action: Correct the syntax.

DBFCDX/1052 Not Custom Built Index

Explanation: Either ORDKEYADD() or ORDKEYDEL() was called for an index which is not a custom built index.

Action: Rebuild the index as a custom built index.

DBFCDX/1053 Index FOR condition didn't evaluate to a logical

Explanation: This error code indicates that you have tried to use an expression in a FOR clause of an INDEX statement which does not evaluate to a logical.

Action: Correct the syntax.

DBFCDX/1054 The type or length of the key doesn't match what's stored on disk

Explanation: This error code indicates:

1. A key expression has changed since the index was created. For example, if you had an index on LAST and changed the width of the field from 20 bytes to 25 bytes, and then tried to use the old index, you would get this error.
2. An index was created on an expression which has a variable length (e.g., TRIM(last)). The key expression must always return a fixed length.

Action: Use PADR() to ensure that the key expression is always the same length.

DBFCDX/1055 DBF Signature Invalid

Explanation: This error indicates that the signature field of the .dbf is invalid. Reasons for this error include:

1. Trying to use a non-DBF file as a .dbf.
2. Corruption of the .dbf.
3. Trying to use a file with a .dbt memo-field.

Action: Recreate the .dbf as a valid CA-Clipper .dbf

DBFCDX/1056 Memo Type Invalid

Explanation: All FPT memo entries have a type field. This error indicates that the type field is invalid. The reasons for this error include:

1. Memo file corruption.
2. Trying to use a memo-field type which CA-Clipper does not understand (e.g., FoxPro for Windows Picture Field Type).

Action: Recreate the .dbf as a valid CA-Clipper .dbf

DBFCDX/1057 Memo Too Long

Explanation: This error occurs when attempting to access a memo field which is over 64K in length. Maximum length of a memo field (with an .fpt memo file) is 65520 bytes. Reasons for this error include:

1. Memo file corruption.
2. Trying to use an .fpt created by a different product which does support >64K memo fields.

Action: Recreate the .dbf as a valid CA-Clipper .dbf

DBFMDX Error Messages

DBFMDX/1006 Exclusive required (creating index)

Explanation: The specified index (.mdx) file could not be created.

Action: This error can be corrected in the following ways:

1. Check to make sure that the .dbf file is being opened EXCLUSIVEly. DB4 requires this, and this is also required with the MDX RDD. If the file exists, make sure it is not marked read-only.
2. In a network environment, make sure the application has the necessary rights to create the file.

See Also: INDEX command, "Network Programming" chapter in the *Programming and Utilities Guide*

DBFMDX/1028 Create error

Explanation: The maximum number of orders per order bag was exceeded. The allowable number of orders that an order bag can contain varies by database driver.

Action: Cut down the number of tags being used.

See Also: RDD Features in the "Replaceable Database Driver Architecture" chapter of the *Driver's Guide*

DBFMDX/1050 Invalid TAG/ORDER

Explanation: You tried to set the tag order of the order bag to a non-existing tag or an illegal (blank) tag name.

Action: Make sure that ORDSETFOCUS() is setting the tag order to a valid tag.

DBFNDX Error Messages

DBFNDX/1001 Open error (.dbf)

Explanation: The specified database (.dbf) file could not be opened.

Action:

1. Check to make sure that the specified file exists. If the file is not in the directory where the application runs, use SET DEFAULT or SET PATH to make the file accessible, or specify the full path name where the file can be found.
2. In a network environment, make sure the application has the necessary rights to access the file. If the file is available only for read access, use the READONLY clause on the USE command.

   CA-Clipper's default error handler (Errorsys.prg) will set NETERR() to true (.T.) and will ask DBFNDX to default if the error was due to a sharing violation on the network.

See Also: USE command, "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNDX/1002 Open error (.dbt)

Explanation: The specified memo (.dbt) file could not be opened.

Action: Some suggestions to resolve the problem are:

1. Check to make sure that the specified file exists. If the file is not in the directory where the application runs, use SET DEFAULT or SET PATH to make the file accessible, or specify the full path name where the file can be found.
2. In a network environment, make sure the application has the necessary rights to access the file. If the file is available only for read access, use the READONLY clause on the USE command.
3. CA-Clipper's default error handler (Errorsys.prg) will set NETERR() to true (.T.) and will ask DBFNDX to default if the error was due to a sharing violation on the network.

See Also: USE command, "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNDX/1003 Open error (index)

Explanation: The specified index (.ndx) file could not be opened.

Action: Some suggestions to resolve the problem are:

1. Check to make sure that the specified file exists. If the file is not in the directory where the application runs, use SET DEFAULT or SET PATH to make the file accessible, or specify the full path name where the file can be found.
2. If the error occurs on SET INDEX TO, then make sure that the index file exists and that the drive and path are valid.
3. In a network environment, make sure the application has the necessary rights to access the file. If the file is available only for read access, use the READONLY clause on the USE or SET INDEX command.

   CA-Clipper's default error handler (Errorsys.prg) will set NETERR() to true (.T.) and will ask DBFNDX to default if the error was due to a sharing violation on the network.

See Also: USE command, SET INDEX command, "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNDX/1004 Open error (creating .dbf)

Explanation: The specified database (.dbf) file could not be created.

Action: Some suggestions to resolve the problem are:

1. Check to make sure that sufficient disk space and directory entries are available. If the file exists, make sure it is not marked read-only.
2. In a network environment, make sure the application has the necessary rights to create the file.

See Also: CREATE command, COPY command, "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNDX/1005 Open error (creating .dbt)

Explanation: The specified memo (.dbt) file could not be created.

Action: Some suggestions to resolve the problem are:

1. Check to make sure that sufficient disk space and directory entries are available. If the file exists, make sure it is not marked read-only.
2. In a network environment, make sure the application has the necessary rights to create the file.

See Also: CREATE command, COPY command, "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNDX/1006 Create error (creating index)

Explanation: The specified index (.ndx) file could not be created.

Action: Some suggestions to resolve the problem are:

1. Check to make sure that sufficient disk space and directory entries are available. If the file exists, make sure it is not marked read-only.
2. In a network environment, make sure the application has the necessary rights to create the file.

See Also: INDEX command, "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNDX/1010 Read error

Explanation: A read error occurred on the specified file.

Action: Some suggestions to resolve the problem are:

1. If the file is on a floppy disk, make sure the disk is properly seated in the drive.
2. In a network environment, make sure the network connection is still valid; check for problems at the server.

See Also: "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNDX/1011 Write error

Explanation: A write error occurred on the specified file.

Action: Some suggestions to resolve the problem are:

1. Check to make sure that sufficient disk space and directory entries are available. Make sure the file is not marked read-only.
2. In a network environment, make sure the application has the necessary rights to write to the file.

See Also: "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNDX/1012 Corruption detected

Explanation: File corruption has been detected in the specified file.

Action: Make sure that the file type is correct for the operation. For index files, make sure that the type of index matches the driver being used (e.g., .ndx for DBFNDX).

DBFNDX/1020 Data type error

Explanation: The value assigned to a FIELD variable was of the wrong type.

Action: Correct the program. If assigning a FIELD variable is from some other variable, make sure the other variable has been initialized (i.e., is not NIL).

See Also: "Basic Concepts" chapter in the *Programming and Utilities Guide*

DBFNDX/1021 Data width error

Explanation: The value assigned to a numeric FIELD variable could not be accurately represented in the field width specified by the database structure.

Action: Change the program to suppress invalid values or modify the structure of the database (.dbf) file to allow for larger values. DBU, the CA-Clipper Database Utility, can be used to modify the structure of a database (.dbf) file.

See Also: "Basic Concepts" and "Database Utility" chapters in the *Programming and Utilities Guide*

DBFNDX/1022 Lock required

Explanation: An attempt was made to update a record in a shared database without first obtaining a lock.

Action: Correct the program. Obtain a record lock or file lock for the work area before attempting to update a record. If shared access is not desired, use the EXCLUSIVE clause of the USE command to gain exclusive access to the database (.dbf) file.

See Also: USE command, RLOCK() function, FLOCK() function, "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNDX/1023 Exclusive required

Explanation: The operation being attempted requires exclusive use of the database (.dbf) file but the work area was opened for shared access.

Action: Correct the program. The PACK, REINDEX, and ZAP commands require exclusive access to the database (.dbf) file. To obtain exclusive access, use the EXCLUSIVE clause of the USE command.

See Also: USE command, "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNDX/1024 Append lock failed

Explanation: A new record could not be appended because a lock could not be obtained for the new record.

Action: For a shared work area, the APPEND BLANK command automatically obtains a record lock for the newly appended record. If the record cannot be locked, the APPEND fails. This generally occurs because another process has obtained a file lock on the database (.dbf) file. Change the program to handle the lock contention.

Note: The default CA-Clipper error handler (Errorsys.prg) handles this error by setting the global NETERR() status to true (.T.) and resuming execution. Thus, the error only causes an error message or alert when a custom error handler is used. For an example of how to handle this error condition, refer to Errorsys.prg.

See Also: "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNDX/1025 Write not allowed

Explanation: An attempt was made to update a record in a work area which was opened for read-only access.

Action: Correct the program. Either suppress operations which update records, or open the database (.dbf) file for read/write access. To obtain read/write access, remove the READONLY clause from the USE command.

See Also: USE command

DBFNDX/1026 Data width error

Explanation: When building an index, the initial evaluation of the key expression (on a blank record) produced a character value of zero length.

Action: Make sure the key expression will produce a value of the same length for all records. Do not use functions such as TRIM() in the key expression. To restrict the length of the key values, use the LEFT() or PADR() functions.

See Also: INDEX command

DBFNDX/1027 Limit exceeded

Explanation: Too many indexes were opened for a work area.

Action: Reduce the number of active indexes for the work area. For each work area, a maximum of 15 indexes can be active at any one time.

See Also: USE command, SET INDEX command

DBFNTX Error Messages

DBFNTX/1001 Open error (.dbf)

Explanation: The specified database (.dbf) file could not be opened.

Action: Some suggestions to resolve the problem are:

1. Check to make sure that the specified file exists. If the file is not in the directory where the application runs, use SET DEFAULT or SET PATH to make the file accessible, or specify the full path name where the file can be found.
2. In a network environment, make sure the application has the necessary rights to access the file. If the file is available only for read access, use the READONLY clause on the USE command.

   CA-Clipper's default error handler (Errorsys.prg) will set NETERR() to true (.T.) and will ask DBFNTX to default if the error was due to a sharing violation on the network.

See Also: USE command, "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNTX/1002 Open error (.dbt)

Explanation: The specified memo (.dbt) file could not be opened.

Action: Some suggestions to resolve the problem are:

1. Check to make sure that the specified file exists. If the file is not in the directory where the application runs, use SET DEFAULT or SET PATH to make the file accessible, or specify the full path name where the file can be found.
2. In a network environment, make sure the application has the necessary rights to access the file. If the file is available only for read access, use the READONLY clause on the USE command.

   CA-Clipper's default error handler (Errorsys.prg) will set NETERR() to true (.T.) and will ask DBFNTX to default if the error was due to a sharing violation on the network.

See Also: USE command, "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNTX/1003 Open error (index)

Explanation: The specified index (.ntx) file could not be opened.

Action: Some suggestions to resolve the problem are:

1. Check to make sure that the specified file exists. If the file is not in the directory where the application runs, use SET DEFAULT or SET PATH to make the file accessible, or specify the full path name where the file can be found.
2. If the file is available only for read access, use the READONLY clause on the USE or SET INDEX command.
3. In a network environment, make sure the application has the necessary rights to access the file.

   CA-Clipper's default error handler (Errorsys.prg) will set NETERR() to true (.T.) and will ask DBFNTX to default if the error was due to a sharing violation on the network.

See Also: USE command, SET INDEX command, "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNTX/1004 Open error (creating .dbf)

Explanation: The specified database (.dbf) file could not be created.

Action: Some suggestions to resolve the problem are:

1. Check to make sure that sufficient disk space and directory entries are available. If the file exists, make sure it is not marked read-only.
2. In a network environment, make sure the application has the necessary rights to create the file.

See Also: CREATE command, COPY command, "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNTX/1005 Open error (creating .dbt)

Explanation: The specified memo (.dbt) file could not be created.

Action: Some suggestions to resolve the problem are:

1. Check to make sure that sufficient disk space and directory entries are available. If the file exists, make sure it is not marked read-only.
2. In a network environment, make sure the application has the necessary rights to create the file.

See Also: CREATE command, COPY command, "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNTX/1006 Create error (creating index)

Explanation: The specified index (.ntx) file could not be created.

Action: Some suggestions to resolve the problem are:

1. Check to make sure that sufficient disk space and directory entries are available. If the file exists, make sure it is not marked read-only.
2. In a network environment, make sure the application has the necessary rights to create the file.

See Also: INDEX command, "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNTX/1010 Read error

Explanation: A read error occurred on the specified file.

Action: Some suggestions to resolve the problem are:

1. If the file is on a floppy disk, make sure the disk is properly seated in the drive. In a network environment, make sure the network connection is still valid; check for problems at the server.
2. Check to make sure that the index, memo file, and.dbf file have not been truncated by DOS's CHKDSK or a similar utility, and that there is no file corruption.

   This can also occur if the STACK space is not enough.

See Also: "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNTX/1011 Write error

Explanation: A write error occurred on the specified file.

Action: Some suggestions to resolve the problem are:

1. Check to make sure that sufficient disk space and directory entries are available. Make sure the file is not marked read-only.
2. In a network environment, make sure the application has the necessary rights to write to the file.
3. On Novell networks, make sure that the FILE OWNER has not been deleted from the user list. Make the file owner SUPERVISOR.
4. Check to see if the media has failed, or if a physical lock has been placed on the file region in question.

See Also: "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNTX/1012 Corruption detected

Explanation: File corruption has been detected in the specified file.

Action: Make sure that the file type is correct for the operation. For index files, make sure that the type of index matches the driver being used (e.g., .ntx for DBFNTX). Try recreating the database file and/or index files.

DBFNTX/1015 Request for RDD improperly linked

Explanation: The RDD was not successfully linked into the application.

Action: Make sure that the RDD used is properly linked into the application. Make sure that the RDD name is spelled correctly.

Refer to the *Drivers Guide* for proper methods of REQUESTing, USE....VIA, and/or RDDSETDEFAULT() for RDDs. There should not be any unresolved externals after linking.

DBFNTX/1020 Data type error

Explanation: The value assigned to a FIELD variable was of the wrong type.

Action: Correct the program. If assigning a FIELD variable is from some other variable, make sure the other variable has been initialized (i.e., is not NIL).

See Also: " Basic Concepts" chapter in the *Programming and Utilities Guide*

DBFNTX/1021 Data width error

Explanation: The value assigned to a numeric FIELD variable could not be accurately represented in the field width specified by the database structure.

Action: Change the program to suppress invalid values, or modify the structure of the database (.dbf) file to allow for larger values. DBU, the *CA-Clipper Database Utility*, can be used to modify the structure of a database (.dbf) file.

See Also: "Basic Concepts" and "Database Utility" chapters in the *Programming and Utilities Guide*

DBFNTX/1022 Lock required

Explanation: An attempt was made to update a record in a shared database without first obtaining a lock.

Action: Correct the program. Obtain a record lock or file lock for the work area before attempting to update a record. If shared access is not desired, use the EXCLUSIVE clause of the USE command to gain exclusive access to the database (.dbf) file.

See Also: USE command, RLOCK() function, FLOCK() function, "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNTX/1023 Exclusive required

Explanation: The operation being attempted requires exclusive use of the database (.dbf) file but the work area was opened for shared access.

Action: Correct the program. The PACK, REINDEX, and ZAP commands require exclusive access to the database (.dbf) file. To obtain exclusive access, use the EXCLUSIVE clause of the USE command.

See Also: USE command, "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNTX/1024 Append lock failed

Explanation: A new record could not be appended because a lock could not be obtained for the new record.

Action: For a shared work area, the APPEND BLANK command automatically obtains a record lock for the newly appended record. If the record cannot be locked, the APPEND fails. This generally occurs because another process has obtained a file lock on the database (.dbf) file. Change the program to handle the lock contention.

Note: The default CA-Clipper error handler (Errorsys.prg) handles this error by setting the global NETERR() status to true (.T.) and resuming execution. Thus, the error only causes an error message or alert when a custom error handler is used. For an example of how to handle this error condition, refer to Errorsys.prg.

See Also: "Network Programming" chapter in the *Programming and Utilities Guide*

DBFNTX/1025 Write not allowed

Explanation: An attempt was made to update a record in a work area which was opened for read-only access.

Action: Correct the program. Either suppress operations which update records or open the database (.dbf) file for read/write access. To obtain read/write access, remove the READONLY clause from the USE command.

See Also: USE command

DBFNTX/1026 Data width error

Explanation: When building an index, the initial evaluation of the key expression (on a blank record) produced a character value of zero length.

Action: Make sure the key expression will produce a value of the same length for all records. Do not use functions such as TRIM() in the key expression. To restrict the length of the key values, use the LEFT() or PADR() functions.

This error can also occur when creating an index on an invalid field type (e.g., trying to create an index on a MEMO FIELD). The maximum key length is 250 bytes.

See Also: INDEX command

DBFNTX/1027 Limit exceeded

Explanation: Too many indices were opened for a work area.

Action: Reduce the number of active indices for the work area. For each work area, a maximum of 15 indices can be active at any one time.

See Also: USE command, SET INDEX command

DBFNTX/1035 Record lock timeout

Explanation: The request for a record lock has timed out.

Action: Make sure that the maximum file locks for the OS has not been exceeded. This can be caused by using DBRLOCK() and not using DBRUNLOCK() to release record locks.

DBFNTX/1038 Lock failure

Explanation: CA-Clipper tried to get a lock or an unlock on an index and failed.

Action: Make sure that the Ntxerr.prg has not been modified. Check network connections, traffic, cards, etc.

DBFNTX/1201 Work area not indexed

Explanation: No index is in use when SEEK() or DBSEEK() is attempted.

Action: Make sure an index is in use.

DBFNTX/1210 Corruption detected

Explanation: The index and the database files are "out of sync" with each other. This is usually due to the index key expression evaluating to a variable length.

Action: Some suggestions to resolve the problem are:

1. Recreate the indices for the database. Ensure that all indices are open when the database is updated. Check for other causes of corruption as well (EMS memory, network problems, improper RDD, etc.).
2. Check that ALL index key expressions are a constant length. Usage of LTRIM(), RTRIM(), TRIM(), ALLTRIM(), STR(), DTOC() can all produce expressions that are not a constant length. The xTRIM() functions should all be padded out to a constant width using PADR().
3. Use all 3 arguments to STR().
4. Use DTOS() instead of DTOC() in index key expressions. DTOC() is dependent upon the SET DATE FORMAT.
5. Using a field or expression aliased into a different work area will also cause this error—e.g., area2->field1 or area2->(str(fld1, 5, 0)).

Runtime Unrecoverable Error Messages

0 **Error System Integrity Error**

Explanation: The error system has encountered an unknown error. The probable cause is memory corruption.

1 **Evaluation Stack Underflow**

Explanation: A stack error has occurred after a Return.

Action: Look for missing or additional RETURN statement.

2 **Memory Error**

Explanation: A memory error has occurred before a RUN. The system has attempted to free the Run space to DOS, and DOS reported an error. The likely cause is corruption in the DOS memory allocation system.

3 **Memory Error**

Explanation: A memory error has occurred after a RUN that prevents rebuilding of the buffer system. This occurs when the system tries to rebuild the Run space, and there is less than 16K available from DOS.

4 **Memory Error**

Explanation: A memory error has occurred corrupting an internal buffer system. This happens when an attempt is made to free an internal buffer, and it is already free.

5 **Memory Error**

Explanation: A memory error has occurred corrupting the buffer and EMM system. This occurs when attempting to map an EMM memory block into conventional address space, and the EMM driver reports an error.

6 **Buffer Error**

Explanation: This occurs when attempting to attach a buffer to a database and the database already has a buffer. The probable cause is memory corruption.

7 **Buffer Error**

Explanation: This occurs when attempting to detach a database, buffer from a database and there is no buffer to detach. The most probable cause is memory corruption.

8 **Buffer Error**

Explanation: The system ran out of database and/or index buffer handles.

9 **Buffer Error**

Explanation: The system ran out of memory when attempting to allocate a database, and/or index buffer.

10 **Too Many Nested BEGIN SEQUENCE/END Blocks**

Explanation: Too many nested BEGIN SEQUENCE/END blocks have been defined. The maximum is 16.

Note: The usual reason this occurs is a LOOP statement within a BEGIN SEQUENCE/END structure like the following:

```
DO WHILE <condition>
   <statements>...
   BEGIN SEQUENCE
       <statements>...
       LOOP
   END <statements>...
ENDDO
```

11 **BEGIN SEQUENCE/END Integrity Error**

Explanation: An underflow of the BEGIN SEQUENCE stack has occurred. This only happens if there is a pending END without a BEGIN SEQUENCE.

12 **Evaluation stack underflow after BEGIN SEQUENCE/END**

Explanation: This error occurs after a BREAK.

14 **SORT Error**

Explanation: The system ran out of memory during a SORT operation while attempting to allocate a database buffer.

15 **SORT Error**

Explanation: Memory corruption has occurred during a SORT operation.

16 **Database Not Open**

Explanation: This occurs at the end of a large block database operation (such as, APPEND, JOIN, UPDATE, or TOTAL) and one of the database files used in the operation is no longer open.

Action: Check the user-defined function used within the erroneous statement for any statements that close database files.

17 **NTX File Corrupted**

Explanation: This occurs when an index buffer is found to be corrupted when attempting to update an index page.

18 **NTX File Corrupted**

Explanation: This occurs when an index buffer is found to be corrupted when attempting to update an index page, and the index is UNIQUE.

19 **NTX File Corrupted**

Explanation: This occurs when an index buffer is found to be corrupted when attempting to update an index page, and the index is non-UNIQUE.

20 **NDX File Key Type Error**

Explanation: An evaluation of a key in a .ndx file does not result in a character or number. This can only occur if the key is a logical value.

21 **NDX File Key Type Error**

Explanation: A SEEK expression evaluates to a different data type than the index expression.

22 **NTX File Key Type Error**

Explanation: A SEEK expression in a .ntx file evaluates to a logical, or the system runs out of memory when evaluating the key expression.

24 **Write error**

Explanation: This error occurs when a CA-Clipper application is unable to write to a database file or to an index file.

Action: Check that sufficient disk space and directory entries are available. Make sure the file is not marked read-only. In a network environment, make sure the application has the necessary rights to write to the file and all network connections are correct.

See Also: "Network Programming" chapter in the *Programming and Utilities Guide*.

37 Too many symbols in symbol table

Explanation: This error occurs with an application linked with EXOSPACE. The application will terminate immediately upon start up. This is caused by CODE GENERATORS that generate an excessive amount of symbols.

Action: Some suggestions to resolve the problem are:

1. Decrease that amount and/or reuse symbols, declare as local.
2. List .PRGs in .CLP files to compile fewer and larger .OBJ files.

See Also: "Compiling" chapter in the *Programming and Utilities Guide* and the "CA-Clipper Technical Specifications" appendix.

92 SORT/INDEX ON Error

Explanation: The system is unable to create a temporary file during a SORT or INDEX operation. This can occur for one of the following reasons:

1. No disk space.
2. Disk is write-protected.
3. No more directory entries.
4. The file already exists and is read-only.
5. There are not enough file handles available.

331 String/Array buffer/memory overflow

Explanation: In the worst case, this error will occur when slightly over a megabyte of strings and/or arrays are in use; the best case is in excess of 16 megabytes. Probably the most common cause of this error is the declaration of extremely large arrays (e.g., LOCAL aArray[500][300]). Every array element requires memory to store (even if its value is NIL). The number of array elements in an array is determined by multiplying the number of elements in every dimension and adding the sum of all dimensions except for the last. For example, a 500 by 300 array has (500 x 300) + 500, or 150,500 elements. As every array element in CA-Clipper requires 14 bytes, this amounts to 150,500 x 14 or 2,107,000 bytes—well in excess of one megabyte and, therefore, potentially dangerous.

Action: Reduce the size and/or number of strings and arrays that are active at any one time. Declare as many string variables and arrays LOCAL as possible.

Note: There is no benefit gained by reusing arrays. CA-Clipper is much more efficient when strings and arrays are thrown away and rebuilt often rather than kept around unnecessarily for long periods.

332 String/Array memory overflow

Explanation: The maximum capacity of the Segment Virtual Object Store (SVOS) system has been exceeded. Due to the dynamic nature of SVOS, it is impossible to state exactly when this error will occur. In the worst case, this error will occur when slightly over a megabyte of strings and/or arrays are in use; the best case is in excess of 16 megabytes. Probably the most common cause of this error is the declaration of extremely large arrays (e.g., LOCAL aArray[500][300]). Every array element requires memory to store (even if its value is NIL). The number of array elements in an array is determined by multiplying the number of elements in every dimension and adding the sum of all dimensions except for the last. For example, a 500 by 300 array has (500 x 300) + 500, or 150,500 elements. As every array element in CA-Clipper requires 14 bytes, this amounts to 150,500 x 14 or 2,107,000 bytes—well in excess of one megabyte and, therefore, potentially dangerous.

Action: Reduce the size and/or number of strings and arrays that are active at any one time. Declare as many string variables and arrays LOCAL as possible.

Note: There is no benefit gained by reusing arrays. CA-Clipper is much more efficient when strings and arrays are thrown away and rebuilt often rather than kept around unnecessarily for long periods.

335 String/Array invalid pointer

Explanation: This results from an error accessing the memory address for a string value during an element assignment while declaring a large multidimensional array.

Action: Try declaring an empty array and then build the array using AADD(). Check string assignments for a NULL character.

336 String/Array memory overflow

Explanation: This results from trying to a declare a large multidimensional array.

Action: Try declaring an empty array, and then build the array using AADD().

415 Cannot open overlay file

Explanation: This error occurs when a CA-Clipper application cannot find or open an overlay file. Overlay files include executable (.EXE) files (in the case of dynamic overlays), and static overlay (.ovl) files. Probably the most common cause of this error is insufficient file handles available to the CA-Clipper application.

Action: Some suggestions to resolve the problem are:

1. The first step in trying to solve this problem is to increase the number of file handles available to the application. Refer to the Files and Buffers section in "The Runtime Environment" chapter of the *Programming and Utilities Guide.*
2. If an insufficient number of file handles is not the problem, it is possible that the CA-Clipper application cannot find the file it is trying to open. It is possible that another file of the same name as the executable is in the DOS path and is being searched for the overlay. Try renaming the executable.

See Also: Specifying the Location of Executable Files section in "The Runtime Environment" chapter of the *Programming and Utilities Guide.*

416 Read error on overlay

Explanation: This error indicates that the runtime manager could not READ from overlay.

Action: Some suggestions to resolve the problem are:

1. Make sure you are not deleting or FCLOSE()ing any file handles that you did not specifically create.
2. If you are on a network, make sure that the server did not lock up. Also make sure that the workstation has not been disconnected. Check the validity of the network cards, cabling, drivers, etc.
3. Increase the number of file handles available to the application.
4. Check that the application's EXE drive and directory are still valid.
5. Make sure the .EXE is marked READONLY (dynamic manager always opens page files SHARED and READONLY).
6. Make sure there are no duplicates of the .EXE file available through any MAP, PATH or SEARCH drive. ERASE duplicate .EXE files.
7. Do not rename the .EXE. Recreate it with an alternative OUTPUT link command.

520 **Attempt to get value for an invalid field type**

Explanation: This error may indicate a corrupted or non CA-Clipper compatible database file.

Action: Repair the database header and field structure.

521 **Replacement of field with invalid data type**

Self-explanatory.

612 **EVAL() given something other than codeblock**

Explanation: A data type other than a code block was supplied as the first argument to EVAL().

Note: A class object is not a code block.

See Also: "Basic Concepts" chapter in the *Programming and Utilities Guide.*

650 **Out of stack space**

Explanation: Stack space is exhausted.

Action: Some suggestions to resolve the problem are:

1. Use STACK or PROCEDURE DEPTH command to instruct the linker to increase the stack space at link time.
2. Check for recursion. Some modifications to the CA-Clipper Errorsys.prg may cause a recursive call, which will exhaust the call stack.
3. Try using the default CA-Clipper Errorsys.prg if the Errorsys.prg was modified to test for runaway recursion caused by an error occurring while attempting to handle an Error Object.
4. Check for UDFs/.PRGs that have the same name as an internal CA-Clipper function.
5. Check for use of an incorrect or invalid version of the Errorsys.prg, i.e., the Summer '87 CA-Clipper Errorsys.prg does use an Error object.

666 **C Function Attempt to Free Invalid Pointer**

Explanation: This error indicates that a C function attempted to free an invalid pointer.

667 Eval stack crashed

Explanation: The EVALUATION STACK (containing LOCALs, etc.) and/or MEMVAR table has been exceeded, causing an attempt to access a locked VM segment in near memory.

Action: Some suggestions to resolve this problem are:

1. Decrease the STACK or PROCEDURE DEPTH setting for the linker.
2. Decrease the number of LOCAL, STATIC or PRIVATE variables in use at one time. i.e., use arrays instead of individual variables.
3. Decrease the number of ITEMS allocated if using ITEM.API of CA-Clipper.
4. Free more conventional memory for use by the VM System.

Note: There could be an error in the runtime error handler.

See Also: Error 650.

668 EVAL STACK FAULT

Explanation: The EVALUATION STACK expanded into a locked VMM segment. The CA-Clipper VM System will use the DGROUP Free Space when conventional DOS memory is very low.

Action: Free up conventional DOS memory by reducing the application's load size and/or increasing free DOS memory as reported by DOS's MEM.EXE command.

Note: There could be an error in the runtime error handler.

See Also: Error 650.

669 EVAL STACK FAULT

Explanation: The MEMVAR table expanded into a locked VM Segment. The CA-Clipper VM System will use the DGROUP free space when conventional DOS memory is low.

Action: Free up conventional DOS memory by reducing the application's load size and/or increasing free DOS memory as reported by DOS's MEM.EXE command.

Note: There could be an error in the runtime error handler.

See Also: Error 650.

670 Memory initialization error

Explanation: An error has occurred during the initialization or re-initialization of the memory system. This error usually indicates an extremely low memory condition at startup, or that an application that was RUN from within CA-Clipper allocated DOS memory without freeing it.

Action: If the error occurred at startup, more conventional memory should be made available for the application. If it occurred immediately following the RUN command, the application that was run should be eliminated to see if this solves the problem. This problem may also occur with an improperly linked or corrupted application.

701 Unable to locate keyboard driver upon startup

Action: Make sure the latest CA-Clipper libraries are used, and that no third-party libraries are linked. Determine that the .EXE is not corrupted.

702 Keyboard driver I/O error upon startup

Explanation: The keyboard detected was not compatible for I/O access.

Action: Make sure the latest CA-Clipper libraries are used, and that no third-party libraries are linked. Determine that the .EXE is not corrupted. Determine the keyboard may be accessed from the DOS prompt.

703 Unable to locate display driver upon startup

Action: Make sure the latest CA-Clipper libraries are used, and that no third-party libraries are linked. Determine that the .EXE is not corrupted.

704 Screen display driver I/O error upon startup

Explanation: The CRT detected was not compatible for I/O access.

Action: Make sure the latest CA-Clipper libraries are used, and that no third-party libraries are linked. Determine that the .EXE is not corrupted. Determine the screen may be accessed from the DOS prompt.

705 Unable to determine disk drivetype upon startup

Action: Make sure the latest CA-Clipper libraries are used, and that no third-party libraries are linked. Determine that the .EXE is not corrupted.

706 Disk drivetype I/O error upon startup

Explanation: The disk detected was not compatible for I/O access.

Action: Make sure the latest libraries are used, and that no third-party libraries are linked. Determine that the .EXE is not corrupted. Determine the disk may be accessed from the DOS prompt.

715 Printer not ready

Explanation: Self-explanatory.

Action: Make sure the latest libraries are used, and that no third-party libraries are linked. Determine that the link instructions are valid and the .EXE is not corrupted. Determine the printer may be accessed from the DOS prompt.

773 Dynamic overlay system not linked in

Explanation: This may indicate an invalid link line or a corrupted executable file.

Action: Make sure the latest libraries are used, and that no third-party libraries are linked. Determine that the link instructions are valid and the .EXE is not corrupted. Confirm that the memory configuration did not affect linking.

998 Missing function

Explanation: CA-Clipper has attempted to execute a function or UDF that is not recognized as valid CA-Clipper code. This error can occur if an attempt is made to execute code compiled with a new version of CA-Clipper under an older version, or because of a corrupted or invalid executable.

Action: Make sure that all code is compiled under the same version of CA-Clipper, and is being linked with the correct version of the libraries. If a third-party linker is used, see if the error occurs when using the real mode linker for CA-Clipper. If a corrupted executable is suspected, recompile all .OBJ files and relink. Declare UDFs or built-in functions hidden from the compiler in macro or INDEXing commands with REQUEST if they are not called elsewhere in the code. Link in missing .OBJs. Make sure all linker warnings and errors are resolved.

999 Attempt to execute invalid code

Explanation: CA-Clipper has attempted to execute something that it does not recognize as valid CA-Clipper code. This error can occur if an attempt is made to execute code compiled with a new version of CA-Clipper under an older version, or because of a corrupted or invalid executable.

Action: Make sure that all code is compiled under the same version of CA-Clipper and is being linked with the correct version of the libraries. If a third-party linker is used, see if the error occurs when using CA-Clipper/Exospace or Blinker. If a corrupted executable is suspected, recompile all .OBJ files and relink.

1010 Read/write in index page failed

Action: Check for memory conflicts that could cause a corruption of the index buffer. Try disabling the EMS with the //E:0 in SET CLIPPER. If using third-party RDDs, try increasing the stack space. If using structural indexes (.CDX, .MDX), delete the index file and recreate.

1011 Read/write access in index page failed

Explanation: CA-Clipper has attempted to read or write to an index page. This error can occur if a the stack has become corrupted or there is insufficient stack space. This may be due to a corrupted or invalid .EXE.

Action: Try increasing the STACK or PROCEDURE DEPTH. Check for memory conflicts that could cause a corruption of the index buffer. Try disabling the EMS with the //E:0 in SET CLIPPER. If using third-party RDDs, try increasing the stack space. If using structural indexes (.CDX, .MDX), delete the index file and recreate. Resolve all linker warnings and errors. Recompile all .OBJs and relink. Check all third party RDD version compatibility.

1020 Record not found

Explanation: This error indicates one of the following conditions:

1. Invalid record number data type specified in DBGOTO().
2. Invalid field data type encountered when loading a field from disk to memory.
3. Invalid field data type encountered when storing a field from memory to disk.

Action: Delete indices and recreate. Check for memory conflicts that could cause a corruption of the index buffer. Try disabling the EMS with the //E:0 in SET CLIPPER. If using third-party RDDs, try increasing the stack space. If using structural indexes (.CDX, .MDX), delete the index file and recreate.

Note: This error is applicable to the DBFNTX replaceable database driver (RDD), and may not be applicable to other RDDs.

1101 Maximum workareas exceeded

Explanation: CA-Clipper supports 250 work areas. This error can be produced if this limit is exceeded. This usually indicates that an internal process tried to access a field which does not exist, or that an internal field structure is not valid for the work area. This means the runtime system is out of work areas for work area index strings.

Action: A good place to start looking is in the third-party RDD or any "in house" C/ASM code. Make sure the database driver was REQUESTed and linked. Check the .LIBrary link order.

1102 Requested RDD not linked

Explanation: The database driver that was specified is not linked into the application.

Action: Make sure that the correct database driver is specified.

1112 Read value larger than 64Kb

Explanation: A read of a variable with length larger than 64Kb was attempted. The limit for character variables and memo fields is 64Kb.

Action: This read error could indicate an index access failure indicating a bad FIELD type, i.e., INDEXing on a MEMO or large character field. Look into third-party RDD or any "in house" C/ASM code. Make sure the REQUESTed database driver was linked and check the .LIBrary link order.

1201 No master index in use

Explanation: An operation, such as a SEEK, was performed in the work area but there was no master or controlling index order.

Action: Make sure that the index ORDER is not ZERO, or nullified, or that the TAG that had been SET ORDER TO was not deleted within a multi-tag index.

1210 Data and Index files out of sync

Explanation: Database (.dbf) and index (.ntx) files are not synchronized, or there is a stack corruption or access problem related to the RDD loaded.

Action: Some suggestions to resolve the problem are:

1. If the cause of this problem is that the database is being updated through a utility external to the application, simply recreate the index (.ntx) file and restart the application.
2. If the database was not updated outside of the application, check the code for the application to ensure that all indexes are open when updates are being done. Correct the code, recreate the index files (.ntx), and restart the application.
3. Check that ALL index key expressions are a constant length. Usage of LTRIM(), RTRIM(), TRIM(), ALLTRIM(), STR(), and DTOC() can all produce expressions that are not a constant length. The TRIM() functions should all be padded out to a constant width using PADR().
4. Use all three arguments to STR().
5. Use DTOS() instead of DTOC() in index key expressions. DTOC() is dependent upon the SET DATE FORMAT.

Note: This error is applicable to the third party replaceable database drivers (RDDs), and may not be applicable to CA-Clipper RDDs. This may mean the stack was corrupted, preventing the application from accessing the RDD's index expression. Increase the STACK or PROCEDURE DEPTH. Look for missing or incorrectly ordered .LIBraries, or loading a mismatched RDD.

See Also: Errors 1010, 1020, and DBFNTX/1210 in this chapter.

1240 Index key evaluation error

Explanation: An error occurred during index key calculation. This can also be caused by a corrupted header in the .dbf file can cause this. INDEX ON with an EVAL clause not returning a logical true (.T.) will also cause this error.

Action: Check the index key to ensure it is not on a logical field. Rebuild the .dbf file in the DBU utility, or with DBCREATE(), not COPY STRUCTURE, then append records.

1242 Data type mismatch on key replacement

Explanation: An attempt was made to replace a key field in the database with an invalid data type. Indexing on a logical field may also cause this error.

Action: Test the index keys and the replacement values in the debugger.

1255 Damaged index header

Explanation: This is caused by an index key expression greater than 255 characters.

Action: Recreate the index file, not exceeding the key expression limit.

See Also: "CA-Clipper Technical Specifications" appendix, check third party documentation.

2155 Read error on index heading page

Explanation: A read error occurred when reading the heading page of the index file, or the header did not contain the .ntx signature, or the macro compiler returned a syntax error when attempting to compile the key expression.

Action: Recreate the index file and check the key expression. Make sure the database header is updated before index recreation.

4001 Number of METHODs exceeded

Explanation: An error occurred when accessing more than 32 methods for an instantiation of a class object at runtime.

Action: CA-Clipper allows no more than 32 methods for any class object at runtime. Check for third party Class .LIBraries and creation of new methods for additional classes. A typographical error or using the SEND operator may mistakenly cause identification of an erroneous method.

4406 Temp file read error

Explanation: An error occurred when reading from a temporary file. This is most likely to occur when the disk where the temporary files are written is full.

Action: Free up the necessary space on the drive for the temporary file, or redirect the temporary file to a different drive via the TEMPPATH option of the CA-Clipper environment variable. Check file ownership and rights if on a network.

4412 Indexing miscalculation

Explanation: This error occurs during indexing if there has been a miscalculation of how indexing should occur.

Action: Some suggestions to resolve the problem are:

1. Try increasing available conventional memory for indexing operation.
2. Try indexing using a small test program.
3. Test recreating the index with one more or one less database record.
4. Test recreating the index with one more or one less byte in the key expression.
5. Test adding CHR(0) to the key.
6. Test with the E:0 parameter in the SET CLIPPER environment variable.

Note: The LEN(CHR(0)) is one byte, the LEN("") is 0 bytes. Test with combinations of the above suggestions.

See Also: "The Runtime Environment" chapter of the *Programming and Utilities Guide.*

4414 Indexing miscalculation

Explanation: This error occurs during indexing or sorting if there has been a miscalculation of how indexing should occur. It will occur if the free conventional memory is insufficient to contain the .dbf record information. It usually occurs when indexing or sorting long field length .dbf files because there is not enough memory to contain the information for the record.

Action: Some suggestions to resolve the problem are:

1. Increase available conventional memory for indexing or sorting operation.
2. Try a protected mode version of application.

See Also: Error 4412.

4424 Temp file creation error

Explanation: An error occurred when creating a temporary file for indexing or sorting. This error can occur for several reasons: insufficient file handles, an invalid TEMPPATH or insufficient network rights for the directory where temporary files are created. Note that temporary files are created in the current directory if no TEMPPATH is specified.

Action: Correct the environment. If insufficient network rights are the problem, either give the user the necessary rights, or redirect the temporary file to a more appropriate directory. Check file ownership and network rights.

See Also: "The Runtime Environment" chapter of the *Programming and Utilities Guide.*

5302 Conventional memory exhausted

Explanation: This is an "out of memory" message that indicates that the CA-Clipper application no longer has sufficient conventional memory available to continue.

Specifically, this error occurs when the virtual memory system attempts and fails to bring a VM segment into conventional memory that had been swapped out to disk or expanded memory.

Action: Make more conventional memory available to the VM system. This may be done in several ways:

1. Increase the amount of conventional memory available before running the application (by removing TSRs, making use of a 386 memory manager, etc.).
2. Dynamically overlay C and ASM code.
3. Ensure that any C or assembly language code in use is making use of the virtual memory system if it is doing dynamic memory allocation. Any memory allocated using the Fixed Memory Allocator functions (_xalloc() and _xgrab()) reduces the amount of conventional memory available if it is not immediately freed.

See Also: Virtual Memory chapter of the *Technical Reference Guide.*

5304 Conventional memory exhausted

Explanation: This is an "out of memory" message that indicates that the CA-Clipper application no longer has sufficient conventional memory available to continue.

Specifically, this error occurs when a function attempts to allocate conventional memory and fails. This error can be received as a result of the failure of _xgrab() in Extend System functions.

Action: Make more conventional memory available to the VM system. This can be done in several ways:

1. Increase the amount of conventional memory available before running the application (by removing TSRs, making use of a 386 memory manager, etc.).
2. Dynamically overlay C and ASM code.
3. Ensure that any C or assembly language code in use is making use of the virtual memory system if it is doing dynamic memory allocation. Any memory allocated using the Fixed Memory Allocator functions (_xalloc() and _xgrab()) reduces the amount of conventional memory available if it is not immediately freed.

See Also: Error 5302 in this chapter, Virtual Memory chapter of the *Technical Reference Guide*.

5305 VM Swap Space Exhausted

Explanation: There is insufficient room in the VM swap area in real memory to load virtualized data.

See Also: Error 5302 in this chapter, Virtual Memory chapter of the *Technical Reference Guide*.

5306 Conventional memory exhausted

Explanation: This is an "out of memory" message that indicates that the CA-Clipper application no longer has sufficient conventional memory available to continue.

Specifically, this error occurs when the virtual memory system attempts and fails to bring a VM segment memory into conventional memory that has been swapped out to disk or expanded memory.

Action: Make more conventional memory available to the VM system. This can be done in several ways:

1. Increase the amount of conventional memory available before running the application (by removing TSRs, making use of a 386 memory manager, etc.).
2. Dynamically overlay C and ASM code.
3. Ensure that any C or assembly language code in use is making use of the virtual memory system if it is doing dynamic memory allocation. Any memory allocated using the Fixed Memory Allocator functions (_xalloc() and _xgrab()) reduces the amount of conventional memory available if it is not immediately freed.

See Also: Error 5302 in this chapter, Virtual Memory chapter of the *Technical Reference Guide*.

5311 VMM Unable to Create Swap File

Explanation: The virtual memory (VM) subsystem is unable to create a swap file on disk. This error occurs for several reasons:

1. The target disk is full.
2. The target directory is full.
3. Insufficient file handles are available.
4. An invalid path is specified in the SWAPPATH parameter of the CLIPPER environment variable, or on the command line used to start the application.
5. The user has insufficient rights on a network drive to create the file.

See Also: "The Runtime Environment" chapter of the *Programming and Utilities Guide*.

5312 VM Swap File Overallocated

Explanation: This is an "out of memory" message that indicates that the CA-Clipper application no longer has sufficient conventional memory available to continue.

Specifically, this error occurs when the virtual memory system needs to swap a virtual memory segment out of conventional memory and it has used all expanded memory and disk space that has been made available to it.

Action: Make more virtual memory available to the application. This can be done by making more expanded memory available by increasing the E setting of the CLIPPER environment variable, or by making more disk space available by increasing the SWAPK setting of the CLIPPER environment variable.

This problem can also be resolved by reducing the size and/or number of strings and arrays that are active at any one time. Probably the most common cause of this error is the declaration of extremely large arrays (e.g., LOCAL aArray[4096][4096]). Note that every array element requires memory to store (even if its value is NIL), and that the number of array elements in an array is determined by multiplying the number of elements in every dimension and adding the sum of all dimensions except for the last. For example, a 4096 by 4096 array has (4096 x 4096) + 4096 or 16,781,312 elements. As every array element in CA-Clipper requires 14 bytes, this amounts to 16,781,312 x 14 or 234,938,368 bytes—well in excess of the theoretical capacity of the virtual memory system.

Note: There is no benefit gained by reusing arrays. CA-Clipper is much more efficient when strings and arrays are thrown away and rebuilt often rather than kept around unnecessarily for long periods.

5313 VMM Write Error on Swap File

Explanation: Write errors typically occur for one of the following reasons:

1. Disk full.
2. Bad sector encountered.
3. The swap file was deleted by another process within a multitasking environment on a workstation in which the SHARE utility has not been loaded.

See Also: Error 5311 in this chapter, "The Runtime Environment" chapter of the *Programming and Utilities Guide.*

5320 VMM IAMBS Manager Error

Explanation: An "out of memory" error has occurred because the VM system is unable to provide more memory, or it encountered an error.

Action: Make more conventional memory available to the VM system and test with a clean environment after a cold boot.

See Also: Error 5302, "The Runtime Environment" chapter of the *Programming and Utilities Guide.*

5321 VMM system unable to free EMS page

Explanation: An error occurred during deallocating EMS page frames.

Action: Some suggestions to resolve the problem are:

1. Use the SET CLIPPER = //BADCACHE and/or unload TSRs and device drivers from the upper memory area.
2. Try a clean boot with a basic CONFIG.SYS and AUTOEXEC.BAT. To find the conflict, add TSRs/device drivers back in until the problem recurs.

See Also: Error 5302 in this chapter, "The Runtime Environment" chapter of the *Programming and Utilities Guide.*

9001 Database RDD failed to load

Explanation: The runtime system has failed to detect an RDD loaded in an application that requires one.

Action: Refer to Actions and Notes under 999 and 1011. Check third party .LIBraries for compatibility and version. Check link script and order. Look for missing RDD .LIBrary or .OBJ files. The .EXE may be invalid.

See Also: Linker and debugger chapters of the *Programming and Utilities Guide.*

9002 RDD invalid or not linked

Explanation: The runtime system has failed to detect a matching RDD loaded for the database and indices used in the application.

Action: Make sure the database header is correct. Recreate the indices. Refer to Actions and Notes under 999 and 1011. Check third party .LIBraries for compatibility and version. Check link script and order. Look for missing RDD .LIBrary or .OBJ files. The .EXE may be invalid.

See Also: Linker and debugger chapters of the *Programming and Utilities Guide*.

9005 Unable to load database table

Explanation: The runtime system has failed to load the database table for use with an RDD, or the RDD failed to load or is unavailable.

Action: Use a utility to test the database header integrity. Refer to Actions and Notes under 999 and 1011. Check third party .LIBrary compatibility and version. Check link script and order. Look for missing RDD .LIBrary or .OBJ files. The .EXE may be invalid.

See Also: Linker and debugger chapters of the *Programming and Utilities Guide*.

9006 DBFNTX RDD index buffer access error

Explanation: The runtime system has failed to detect an RDD loaded in an application that requires one.

Action: Check third party .LIBraries for compatibility and version. Check link script and order. Look for missing RDD .LIBrary or .OBJ files. The .EXE may be invalid.

See Also: Errors 999 and 1011 in this chapter, linker and debugger chapters of the *Programming and Utilities Guide*.

9999 Invalid .EXE caused by RDD unavailable at startup

Explanation: The runtime system has failed to detect an RDD loaded in an application that requires one.

Action: Check third party .LIBraries for compatibility and version. Check link script and order. Look for missing RDD .LIBrary or .OBJ files. The .EXE may be invalid.

See Also: Errors 999 and 1011 in this chapter, linker and debugger chapters of the *Programming and Utilities Guide*.

VM Integrity failure

Explanation: This error indicates a misuse of the VM System has occurred by either an internal or external (third-party) subsystem.

Action: Test the following suggestions to help eliminate the problem:

1. Check for memory conflicts that could result in virtual memory corruption.
2. Check for incorrect data type passed to the IAMBS (VMM) or a GET instance variable.
3. Check third-party .LIBraries for compatibility and version.
4. Test using a simple .LNK script to create the .EXE.
5. Resolve 100% of all warnings and errors occurring at link time. Link with VERBOSE while redirecting to an error output file.
6. Attempt to extract a test without using built-in, add-on, or third-party .LIBrary functions, and see if the error still occurs.
7. Create a protected mode version of the application linking with MAP S, N, A. Identify the runtime address reported by comparing it with the named module for the most closely matched address listed in the .MAP.

See Also: Error 999 and 5302, linker and debugger chapters of the *Programming and Utilities Guide*, Virtual Memory chapter in the *Technical Reference Guide*.

Chapter 6
RMAKE Errors

This section is a summary of error messages possible when using RMAKE, the CA-Clipper make utility. The messages are divided into categories according to severity and the point in the make process where the error occurs. Each category is described below, followed by a listing of all messages in each category. For specific information on the syntax and operation of RMAKE, refer to the "Program Maintenance—RMAKE.EXE" chapter in the *Programming and Utilities Guide.*

RMAKE Warnings

RMAKE warnings indicate potential errors in the make file but do not prevent the make process from continuing. The DOS return code is unaffected by warnings.

The general format of an RMAKE warning message is as follows:

```
<filename>(<line>) Warning RMAKE/R1xxx <message text>
   [: <symbol>]
```

If the warning is not associated with a particular line of a make file, no filename or line number is displayed. Unless the /W option is specified, RMAKE does not display any warning messages.

RMAKE Execution Errors

RMAKE results in an execution error if any program executed as an action in a make file results in a DOS return code other than zero. RMAKE sets the DOS return code to 1 after an execution error and, unless the /I option was specified, terminates the make run.

The general format of an RMAKE execution error message is as follows:

```
<filename>(<line>) Error RMAKE/R2xxx Exit <n>:
   <command line text>
```

The exit number indicates the DOS return code of the indicated command line in the make file.

RMAKE Fatal Errors

Fatal error messages identify serious make file problems from which RMAKE cannot recover. Fatal errors may occur during either the parse or the make phase. After a fatal error, RMAKE terminates the make run and sets the DOS return code to 1.

The general format of a fatal error message is as follows:

```
<filename>(<line>) Fatal RMAKE/R3xxx <message text>
   [: <symbol>]
```

If the error is not associated with a particular line of a make file, no file name or line number is displayed.

RMAKE Warning Messages

R1001 **Ignoring redefinition of command-line macro**

Explanation: A make file attempted to assign a new value to a macro defined on the command line or in the RMAKE environment variable. The redefinition is ignored, and the macro retains its previous value.

Action: Specify an #undef directive to clear the definition of a command-line macro, allowing a new value to be assigned.

R1002 **Target does not exist: '<file>'**

Explanation: The indicated target file is used in a dependency rule and does not exist.

Action: RMAKE will build the target.

R1003 **Ignoring text: '<text>'**

Explanation: A statement in a make file contains extraneous characters. The characters are ignored.

Action: Examine the make file to ensure that the extraneous text is not the result of some other error in the file. If not, delete the characters from the make file.

RMAKE Execution Error Messages

R2001 **Exit n: '<action line>'**

Explanation: A program executed as one of the actions in a dependency or inference rule returned a nonzero DOS return code. This means the executable program terminated abnormally for some reason. The make run terminates unless the /I option was used.

Action: Correct the problem before attempting the make again.

RMAKE Fatal Error Messages

R3001 **Too many make files**

Explanation: You specified more than 16 make files on the command line or in the RMAKE environment variable.

Action: Reduce the number of make files to 16 or fewer, and run the make again.

R3002 **Can't open: '<file>'**

Explanation: RMAKE was unable to open a make file specified on the command line, in the RMAKE environment variable, or in a #include directive.

Action: A full or partial path may be specified as part of any make filename in order to force RMAKE to look for the file in a particular drive and/or directory. If no path is specified, RMAKE searches only the current directory. The /S option causes RMAKE to search subdirectories as well.

R3003 **Invalid option: '<option>'**

Explanation: An invalid option was specified on the command line or in the RMAKE environment variable. The available options can be found in the "Program Maintenance—RMAKE.EXE" chapter in the *Programming and Utilities Guide.*

Action: Correct and run the make again.

R3004 **Out of memory**

Explanation: There is not enough memory for more definitions.

Action: Simplify the make by breaking it into two separate runs.

R3005 **Internal workspace exhausted**

Explanation: The internal workspace was exhausted.

Action: The workspace can be enlarged using the /XW option.

R3006 **Symbol table exhausted**

Explanation: The internal symbol table was exhausted.

Action: The table can be enlarged using the /XS option.

R3007 String too large

Explanation: A single string in the make file exceeded 1024 bytes.

Action: Simplify the offending line in the make file.

R3008 String table exhausted

Explanation: No more memory is available for definitions.

Action: The make should be simplified by breaking it into two separate runs.

R3009 File table exhausted

Explanation: The table used to track filenames and time stamps is exhausted.

Action: The make should be simplified by breaking it into two separate runs.

R3010 Too many actions

Explanation: A dependency or inference rule specified more than 32 action lines, or the total of the action lines exceeded 1024 bytes.

Action: Simplify the offending definition.

R3011 Too many dependencies

Explanation: A dependency rule specified more than 128 dependent files.

Action: Simplify the rule.

R3012 Syntax error: '<token>'

Explanation: A syntax error was detected in the indicated rule or directive. The proper syntax for all make file rules and directives can be found in the "Program Maintenance—RMAKE.EXE" chapter in the *Programming and Utilities Guide*.

Action: Correct and run the make again.

R3013 Unbalanced parentheses

Explanation: You specified a rule or directive with unbalanced parentheses. Each opening parenthesis must have a corresponding closing parenthesis.

Action: Examine the offending line in the make file and correct the problem.

R3014 #else without if

Explanation: A #else directive was encountered, but no corresponding #if directive preceded it. #else is illegal in a make file unless it falls between a #if directive and a #endif directive.

Action: Correct and run the make again.

R3015 #endif without if

Explanation: You specified a #endif directive with no #if directive preceding it. #endif is illegal in a make file unless there is a corresponding #if directive preceding it.

Action: Correct and run the make again.

R3016 Open conditionals

Explanation: You specified one or more #if unterminated directives at the end of a make file. All #if directives must have corresponding #endif directives.

Action: Check the make file carefully and close all conditional constructs with a #endif directive. Using indentation in the make file will make problems of this nature easier to spot and avoid.

R3017 Unrecognized directive: '<directive>'

Explanation: You specified an invalid directive. A complete list of make file directives can be found in the "Program Maintenance—RMAKE.EXE" chapter in the *Programming and Utilities Guide.*

Action: Correct and run the make again.

R3018 Dependency does not exist: '<file>'

Explanation: You specified a dependency rule containing a nonexistent file as a dependency. RMAKE is, therefore, unable to determine whether the target file is out of date. RMAKE searches the current directory only for dependency files unless you define a special macro to specify where dependency files are to be found. See the "Program Maintenance—RMAKE.EXE" chapter in the *Programming and Utilities Guide* for more information on these special search macros.

Action: Correct and run the make again.

R3019 Circular dependency

Explanation: You specified a target file that depends on itself, either directly or through a series of dependency rules. Circular dependencies are not allowed.

Action: Check the make file resolving any such dependencies, and run the make again.

R3020 Environment overflows workspace

Explanation: You specified one or more DOS SET commands as actions, and the text of the commands has overflowed the internal workspace.

Action: Expand the workspace with the /XW option and run the make again.

R3021 Error in redirection

Explanation: An error occurred when RMAKE attempted to redirect the standard input or output files as specified in an action line. This usually means the output file could not be created or opened. There may not be enough room to create another file in the directory specified, or on a network the file was inadvertently deleted by another user.

Action: Correct and run the make again.

R3022 Can't execute '<action line>'

Explanation: A failure occurred when trying to execute an action line.

Action: Check the action line to make sure the syntax used to execute the specified program is correct.

Chapter 7
Build and Debugger Error Messages

This chapter describes error messages that you may receive while building or debugging an application from within the CA-Clipper Workbench.

Build Error Messages

Build Fatal Error Messages are presented in a STOP message box titled "Fatal error during build." The following messages indicate various file errors encountered during a build.

File Errors

The WORKING and BACKUP directories specified in the [AutoMake] section of CACI.INI (written to the Windows directory during installation) have not been properly defined. They must exist and be distinct. A file may have been deleted, renamed, moved or write-protected. Less probable—but still possible—hard disk, directory structure, or file integrity may be compromised.

"Backup directory not defined or not useable"

"Backup source & target filespecs are identical"

"Backup source not found"

"Error creating backup file"

"Error creating compile script" <name>

"Error opening application file" <name>

"Restore source & target filespecs are identical"

"Restore target is invalid"

Other File Errors

Below are listed file errors caused by reasons other than poorly defined WORKING and BACKUP directories. These errors can be received due to the following reasons: the file is in use by another process or user; the specified directory does not exist; an illegal DOS file name has been specified; or a hardware problem with the network or hard drive has occurred.

"A directory may have been deleted, renamed or moved"

"An erroneous file specification has been entered under compile or link options"

"A file is currently being used by another process"

"Bad filename" <name>

"Error opening backup source"

"Error opening backup target"

"File specification invalid" <name>

"Invalid application filename" <name>

CLIPPERCMD Error Message

Some information is passed to the CA-Clipper compiler through a temporary CLIPPERCMD environment variable. If the length of CLIPPERCMD exceeds 127 characters, a fatal error is generated:

"Exception parameter" "CLIPPERCMD Variable would be too long"

Information passed through CLIPPERCMD includes error status switch, default libraries, temporary file specification, command definition file, and include file override.

Link Template Processing Messages

During link template processing, an attempt to operate on a permanent symbol will evoke one of the following:

"Protected variable may not be deleted" <name>

"Protected variable may not be modified" <name>

"Protected variable may not be replaced" <name>

The following message indicates that link template processing could not be completed. The "Use link template" box had been checked but no template had been provided.

"Exception parameter" "Fatal error processing link template"

Miscellaneous Error Messages

Other messages are listed for completeness but should be evoked only during debugging. Call Computer Associates Technical Support should one occur.

"Error opening restore source"

"Error opening restore target"

"Exception parameter" "Application not open"

"Exception parameter" "Database not opened"

"Exception parameter" "Error deleting" <name>

"Exception parameter" "Fatal error processing compile log"

"Exception parameter" "Internal AutoMake Error"

"Exception parameter" "Multiple instances not permitted"

"Invalid signature" <name>

Miscellaneous Linker Error Messages

The following messages are echoed from the normal compile and link batch files when there is no requirement to compile or link. They are found in the compile and link output (log) files, respectively.

"All object files are up to date" <name>" is up to date."

The following messages are inserted into a build log file or created, if necessary, to indicate why no output file create/update was done. Any one will cause the linker (or other object file processor) diagnostics to be displayed in an edit window.

"MESSAGE - Build canceled"

"MESSAGE - Build log missing"

"MESSAGE Syntax-only specified"

"MESSAGE Compile-only specified"

The following messages are echoed from a template-directed build batch file for appearance in the build log file. Any one will cause the linker (or other object file processor) diagnostics to be displayed in an edit window. Warnings, errors and severe errors are issued upon detection of syntax errors in the link template.

MESSAGE "<template supplied string>"

Explantion: This message is created by a template #message command.

WARNING Line <n> "Command out of context"

WARNING Line <n> "Unexpected statement"

ERROR Line <n> "Invalid <more> string"

ERROR Line <n> "Invalid <sep> string"

ERROR Line <n> "Invalid <write> string"

ERROR Line <n> "Invalid assign operator"

ERROR Line <n> "Invalid assign operand"

ERROR Line <n> "Invalid assign target"

ERROR Line <n> "Invalid comment string"

ERROR Line <n> "Invalid escape specification"

ERROR Line <n> "Invalid generator symbol"

FATAL ERROR Line <n> "Unclosed conditional block"

FATAL ERROR Line <n> "Unterminated batch"

Debugger Error Messages

The following error messages can occur when attempting to debug an application from within the CA-Clipper Workbench.

DBG0001 Unable to communicate with CA-Clipper application

Explanation: The debugger cannot send or receive an internal message to the CA-Clipper application.

Action: Close the CA-Clipper application window, close the debugger window, and restart the debug session.

DBG0002 The Windows/DOS communications server can operate only in Windows Enhanced mode.

Explanation: An attempt was made to run a debug session under Windows Standard or Real mode.

Action: Restart Windows in Enhanced mode.

DBG0003 Unable to execute Windows/DOS communications client program. Program not found.

Explanation: CACI.BAT could not be found.

Action: Be sure that the directory which contains CACI.BAT (normally C:\CLIP53\BIN) is included in the current PATH.

DBG0004 Unable to execute Windows/DOS communications client program. Path not found.

Explanation: CACI.BAT could not be found.

Action: Be sure that the directory which contains CACI.BAT (normally C:\CLIP53\BIN) is included in the current PATH.

DBG0005 Unable to execute Windows/DOS communications client program. Insufficient Memory.

Explanation: There is not enough available memory to start a DOS session.

Action: Shut down all other Windows applications that are currently running and retry the operation.

DBG0006 Unable to execute Windows/DOS communications client program. Execution error number n.

Explanation: An error has occurred while attempting to create a DOS session.

Action: Contact Computer Associates Technical Support. Please make a note of the execution error number.

DBG0100 Unable to connect to Windows/DOS communications Pipe.

Explanation: CA-Clipper Workbench could not establish contact with the CA-Clipper program.

Action: Place the following line in the [386Enh] section of the SYSTEM.INI file:

```
Device = CAWDCI.386
```

The file CAWDCI.386 has been installed to the Windows system directory upon installation of the CA-Clipper Workbench. Be sure that this file exists in this directory. Be sure that an environment variable named PIPEHANDLE exists. If it does not, add the following line to the AUTOEXEC.BAT file:

```
SET PIPEHANDLE=0
```

DBG0101 Unable to disconnect from Windows/DOS communications Pipe.

Explanation: CA-Clipper Workbench could not close the CA-Clipper application DOS window.

Action: The DOS window must be closed by using the window's control box and selecting the CLOSE menu item.

DBG0102 Send message over pipe failed.

Explanation: The debugger cannot send or receive an internal message to the CA-Clipper application.

Action: Close the CA-Clipper application window, close the debugger window, and restart the debug session.

DBG0103 Unable to communicate with CA-Clipper application.

Explanation: The debugger cannot send or receive an internal message to the CA-Clipper application.

Action: Close the CA-Clipper application window, close the debugger window, and restart the debug session.

DBG0104 Unable to create Windows/DOS Communications Server window.

Explanation: The creation of an internal CA-Clipper Workbench window has failed.

Action: Close the CA-Clipper Workbench and restart Windows.

DBG0105 Error requesting DOS client failure notification.

Explanation: The debugger cannot send or receive an internal message to the CA-Clipper application.

Action: Close the CA-Clipper application window, close the debugger window, and restart the debug session.

DBG0106 The Clipper program has abnormally terminated.

Explanation: The CA-Clipper program has ended unexpectedly.

Action: Correct the code in the CA-Clipper program.

DBG0200 Unable to obtain local memory for DosBoxData.

Explanation: There is not enough available memory to start a DOS session.

Action: Shut down all other Windows applications that are currently running and retry the operation.

DBG0201 Unable to create DosBoxData index entry.

Explanation: There is not enough available memory to start a DOS session.

Action: Shut down all other Windows applications that are currently running and retry the operation.

DBG0202 Unable to obtain address of DosBoxData.

Explanation: There is not enough available memory to start a DOS session.

Action: Shut down all other Windows applications that are currently running and retry the operation.

DBG0203 Unable to remove DosBoxData.

Explanation: There is not enough available memory to start a DOS session.

Action: Shut down all other Windows applications that are currently running and retry the operation.

DBG0204 Unable to obtain memory for WinOldApIndex.

Explanation: There is not enough available memory to start a DOS session.

Action: Shut down all other Windows applications that are currently running and retry the operation.

Chapter 8
DOS Error Messages

The following table provides a complete listing of DOS error numbers and their descriptions.

DOS Errors

Error Number	Description	Error Number	Description
1	Invalid function number	19	Attempt to write on write-protected diskette
2	File not found	20	Unknown unit
3	Path not found	21	Drive not ready
4	Too many open files (no handles left)	22	Unknown command
5	Access denied	23	Data error (CRC)
6	Invalid handle	24	Bad request structure length
7	Memory control blocks destroyed	25	Seek error
8	Insufficient memory	26	Unknown media type
9	Invalid memory block address	27	Sector not found
10	Invalid environment	28	Printer out of paper
11	Invalid format	29	Write fault
12	Invalid access code	30	Read fault
13	Invalid data	31	General failure
14	Reserved	32	Sharing violation
15	Invalid drive was specified	33	Lock violation
16	Attempt to remove the current directory	34	Invalid disk change
17	Not same device	35	FCB unavailable
18	No more files	36	Sharing buffer overflow

DOS Errors (cont.)

Error Number	Description	Error Number	Description
37-49	Reserved	66	Network device type incorrect
50	Network request not supported	67	Network name not found
51	Remote computer not listening	68	Network name limit exceeded
52	Duplicate name on network	69	Network BIOS session limit exceeded
53	Network name not found	70	Temporarily paused
54	Network busy	71	Network request not accepted
55	Network device no longer exists	72	Print or disk redirection paused
56	Network BIOS command limit exceeded	73-79	Reserved
57	Network adapter hardware error	80	File already exists
58	Incorrect response from network	81	Reserved
59	Unexpected network error	82	Cannot make directory entry
60	Incompatible remote adapter	83	Fail on INT 24H
61	Print queue full	84	Too many redirections
62	Not enough space for print file	85	Duplicate redirection
63	Print file deleted (not enough space)	86	Invalid password
64	Network name deleted	87	Invalid parameter
65	Access denied	88	Network device fault

Appendix A
ASCII Character Chart

This appendix shows the ASCII characters and their decimal and hexadecimal values.

ASCII Character Table

DEC	HEX	CHAR	DEC	HEX	CHAR
0	00		18	12	↕
1	01	☺	19	13	!!
2	02	☻	20	14	¶
3	03	♥	21	15	§
4	04	♦	22	16	▬
5	05	♣	23	17	↨
6	06	♠	24	18	↑
7	07	•	25	19	↓
8	08	◘	26	1A	→
9	09	○	27	1B	←
10	0A	◙	28	1C	∟
11	0B	♂	29	1D	↔
12	0C	♀	30	1E	▲
13	0D	♪	31	1F	▼
14	0E	♫	32	20	
15	0F	☼	33	21	!
16	10	►	34	22	"
17	11	◄	35	23	#

ASCII Character Table (cont.)

DEC	HEX	CHAR	DEC	HEX	CHAR
36	24	$	70	46	F
37	25	%	71	47	G
38	26	&	72	48	H
39	27	'	73	49	I
40	28	(	74	4A	J
41	29	)	75	4B	K
42	2A	*	76	4C	L
43	2B	+	77	4D	M
44	2C	,	78	4E	N
45	2D	-	79	4F	O
46	2E	.	80	50	P
47	2F	/	81	51	Q
48	30	0	82	52	R
49	31	1	83	53	S
50	32	2	84	54	T
51	33	3	85	55	U
52	34	4	86	56	V
53	35	5	87	57	W
54	36	6	88	58	X
55	37	7	89	59	Y
56	38	8	90	5A	Z
57	39	9	91	5B	[
58	3A	:	92	5C	\
59	3B	;	93	5D	]
60	3C	<	94	5E	^
61	3D	=	95	5F	_
62	3E	>	96	60	`
63	3F	?	97	61	a
64	40	@	98	62	b
65	41	A	99	63	c
66	42	B	100	64	d
67	43	C	101	65	e
68	44	D	102	66	f
69	45	E	103	67	g

ASCII Character Table (cont.)

DEC	HEX	CHAR	DEC	HEX	CHAR
104	68	h	133	85	à
105	69	i	134	86	å
106	6A	j	135	87	ç
107	6B	k	136	88	ê
108	6C	l	137	89	ë
109	6D	m	138	8A	è
110	6E	n	139	8B	ï
111	6F	o	140	8C	î
112	70	p	141	8D	ì
113	71	q	142	8E	Ä
114	72	r	143	8F	Å
115	73	s	144	90	É
116	74	t	145	91	æ
117	75	u	146	92	Æ
118	76	v	147	93	ô
119	77	w	148	94	ö
120	78	x	149	95	ò
121	79	y	150	96	û
122	7A	z	151	97	ù
123	7B	{	152	98	ÿ
124	7C	|	153	99	Ö
125	7D	}	154	9A	Ü
126	7E	~	155	9B	¢
127	7F	⌂	156	9C	£
128	80	Ç	157	9D	¥
129	81	ü	158	9E	Pt
130	82	é	159	9F	*f*
131	83	â	160	A0	á
132	84	ä	161	A1	í

ASCII Character Table (cont.)

DEC	HEX	CHAR	DEC	HEX	CHAR
162	A2	ó	190	BE	╛
163	A3	ú	191	BF	┐
164	A4	ñ	192	C0	└
165	A5	Ñ	193	C1	┴
166	A6	ª	194	C2	┬
167	A7	º	195	C3	├
168	A8	¿	196	C4	─
169	A9	⌐	197	C5	┼
170	AA	¬	198	C6	╞
171	AB	½	199	C7	╟
172	AC	¼	200	C8	╚
173	AD	¡	201	C9	╔
174	AE	«	202	CA	╩
175	AF	»	203	CB	╦
176	B0	▓	204	CC	╠
177	B1	▒	205	CD	═
178	B2	▓	206	CE	╬
179	B3	│	207	CF	╧
180	B4	┤	208	D0	╨
181	B5	╡	209	D1	╤
182	B6	╢	210	D2	╥
183	B7	╖	211	D3	╙
184	B8	╕	212	D4	╘
185	B9	╣	213	D5	╒
186	BA	║	214	D6	╓
187	BB	╗	215	D7	╫
188	BC	╝	216	D8	╪
189	BD	╜	217	D9	┘

ASCII Character Table (cont.)

DEC	HEX	CHAR	DEC	HEX	CHAR
218	DA	┌	237	ED	ϕ
219	DB	▉	238	EE	∈
220	DC	▄	239	EF	∩
221	DD	▌	240	F0	≡
222	DE	▐	241	F1	±
223	DF	▀	242	F2	≥
224	E0	∝	243	F3	≤
225	E1	β	244	F4	⌠
226	E2	Γ	245	F5	⌡
227	E3	π	246	F6	÷
228	E4	Σ	247	F7	≈
229	E5	σ	248	F8	°
230	E6	μ	249	F9	•
231	E7	T	250	FA	•
232	E8	Φ	251	FB	√
233	E9	θ	252	FC	n
234	EA	Ω	253	FD	2
235	EB	δ	254	FE	■
236	EC	∞	255	FF	

Note: In Microsoft Windows, character sets differ based on the font being used. For a character map of each font, use the Windows Character Map application.

Appendix B
Color Table

The following table lists the color codes that can be used with the SET COLOR command and the SETCOLOR() function. Note, however, that SETCOLOR() does not support the numeric color codes.

Color Table

Color	Letter	Number	Monochrome
Black	N, Space	0	Black
Blue	B	1	Underline
Green	G	2	White
Cyan	BG	3	White
Red	R	4	White
Magenta	RB	5	White
Brown	GR	6	White
White	W	7	White
Gray	N+	8	Black
Bright Blue	B+	9	Bright Underline
Bright Green	G+	10	Bright White
Bright Cyan	BG+	11	Bright White
Bright Red	R+	12	Bright White
Bright Magenta	RB+	13	Bright White
Yellow	GR+	14	Bright White
Bright White	W+	15	Bright White
Black	U		Underline
Inverse video	I		Inverse Video
Blank	X		Blank

Appendix C
CA-Clipper INKEY() Codes

The following tables illustrate the Key Name, Key Code, and Constant definitions for CA-Clipper supported key strokes and Control, Alt and Shift key combinations as they are declared in the header file Inkey.ch. The Inkey tables cover Cursor Movement keys, Editing keys, Control keys, Alt keys, Function keys, Shift+Function keys, Control+Functions keys, Alt+Function keys, and Mouse Events.

Cursor Movement Keys

Key Name	Key Code	Constant
Up arrow, Ctrl+E	5	K_UP
Down arrow, Ctrl+X	24	K_DOWN
Left arrow, Ctrl+S	19	K_LEFT
Right arrow, Ctrl+D	4	K_RIGHT
Home, Ctrl+A	1	K_HOME
End, Ctrl+F	6	K_END
PgUp, Ctrl+R	18	K_PGUP
PgDn, Ctrl+C	3	K_PGDN
Ctrl+Left arrow, Ctrl+Z	26	K_CTRL_LEFT
Ctrl+Right arrow, Ctrl+B	2	K_CTRL_RIGHT
Ctrl+Home, Ctrl+]	29	K_CTRL_HOME
Ctrl+End, Ctrl+W	23	K_CTRL_END
Ctrl+PgUp, Ctrl+Hyphen	31	K_CTRL_PGUP
Ctrl+PgDown, Ctrl^	30	K_CTRL_PGDN
Ctrl+Return	10	K_CTRL_RET
Esc, Ctrl+[	27	K_ESC
Return, Ctrl+M	13	K_RETURN
Enter, Ctrl+M	13	K_ENTER

Editing Keys

Key Name	Key Code	Constant
Del, Ctrl+G	7	K_DEL
Tab, Ctrl+I	9	K_TAB
Shift+Tab	271	K_SH_TAB
Ins, Ctrl+V	22	K_INS
Ctrl+Backspace	127	K_CTRL_BS
Backspace, Ctrl+H	8	K_BS

Control Keys

Key Name	Key Code	Constant
Ctrl+A, Home	1	K_CTRL_A
Ctrl+B, Ctrl+Right arrow	2	K_CTRL_B
Ctrl+C, PgDn, Ctrl+ScrollLock	3	K_CTRL_C
Ctrl+D, Right arrow	4	K_CTRL_D
Ctrl+E, Up arrow	5	K_CTRL_E
Ctrl+F, End	6	K_CTRL_F
Ctrl+G, Del	7	K_CTRL_G
Ctrl+H, Backspace	8	K_CTRL_H
Ctrl+I, Tab	9	K_CTRL_I
Ctrl+J	10	K_CTRL_J
Ctrl+K	11	K_CTRL_K
Ctrl+L	12	K_CTRL_L
Ctrl+M, Return	13	K_CTRL_M
Ctrl+N	14	K_CTRL_N
Ctrl+O	15	K_CTRL_O
Ctrl+P	16	K_CTRL_P
Ctrl+Q	17	K_CTRL_Q
Ctrl+R, PgUp	18	K_CTRL_R
Ctrl+S, Left arrow	19	K_CTRL_S
Ctrl+T	20	K_CTRL_T
Ctrl+U	21	K_CTRL_U
Ctrl+V, Ins	22	K_CTRL_V
Ctrl+W, Ctrl+End	23	K_CTRL_W

Control Keys(cont.)

Key Name	Key Code	Constant
Ctrl+X, Down arrow	24	K_CTRL_X
Ctrl+Y	25	K_CTRL_Y
Ctrl+Z, Ctrl+Left arrow	26	K_CTRL_Z

Alt Keys

Key Name	Key Code	Constant	Key Name	Key Code	Constant
Alt+A	286	K_ALT_A	Alt+N	305	K_ALT_N
Alt+B	304	K_ALT_B	Alt+O	280	K_ALT_O
Alt+C	302	K_ALT_C	Alt+P	281	K_ALT_P
Alt+D	288	K_ALT_D	Alt+Q	272	K_ALT_Q
Alt+E	274	K_ALT_E	Alt+R	275	K_ALT_R
Alt+F	289	K_ALT_F	Alt+S	287	K_ALT_S
Alt+G	290	K_ALT_G	Alt+T	276	K_ALT_T
Alt+H	291	K_ALT_H	Alt+U	278	K_ALT_U
Alt+I	279	K_ALT_I	Alt+V	303	K_ALT_V
Alt+J	292	K_ALT_J	Alt+W	273	K_ALT_W
Alt+K	293	K_ALT_K	Alt+X	301	K_ALT_X
Alt+L	294	K_ALT_L	Alt+Y	277	K_ALT_Y
Alt+M	306	K_ALT_M	Alt+Z	300	K_ALT_Z

Function Keys

Key Name	Key Code	Constant
F1, Ctrl+\	28	K_F1
F2	-1	K_F2
F3	-2	K_F3
F4	-3	K_F4
F5	-4	K_F5
F6	-5	K_F6
F7	-6	K_F7
F8	-7	K_F8
F9	-8	K_F9
F10	-9	K_F10

Shift+Function Keys

Key Name	Key Code	Constant
Shift+F1	-10	K_SH_F1
Shift+F2	-11	K_SH_F2
Shift+F3	-12	K_SH_F3
Shift+F4	-13	K_SH_F4
Shift+F5	-14	K_SH_F5
Shift+F6	-15	K_SH_F6
Shift+F7	-16	K_SH_F7
Shift+F8	-17	K_SH_F8
Shift+F9	-18	K_SH_F9
Shift+F10	-19	K_SH_F10

Control+Function Keys

Key Name	Key Code	Constant
Ctrl+F1	-20	K_CTRL_F1
Ctrl+F2	-21	K_CTRL_F2
Ctrl+F3	-22	K_CTRL_F3
Ctrl+F4	-23	K_CTRL_F4
Ctrl+F5	-24	K_CTRL_F5
Ctrl+F6	-25	K_CTRL_F6
Ctrl+F7	-26	K_CTRL_F7
Ctrl+F8	-27	K_CTRL_F8
Ctrl+F9	-28	K_CTRL_F9
Ctrl+F10	-29	K_CTRL_F10

Alt+Function Keys

Key Name	Key Code	Constant
Alt+F1	-30	K_ALT_F1
Alt+F2	-31	K_ALT_F2
Alt+F3	-32	K_ALT_F3
Alt+F4	-33	K_ALT_F4
Alt+F5	-34	K_ALT_F5
Alt+F6	-35	K_ALT_F6
Alt+F7	-36	K_ALT_F7

Alt+Function Keys (cont.)

Key Name	Key Code	Constant
Alt+F8	-37	K_ALT_F8
Alt+F9	-38	K_ALT_F9
Alt+F10	-39	K_ALT_F10

Mouse Events

Event Name	Key Code	Constant
Mouse Events	1001	K_MOUSEMOVE
Mouse Left Click Down	1002	K_LBUTTONDOWN
Mouse Left Click Up	1003	K_LBUTTONUP
Mouse Right Click Down	1004	K_RBUTTONDOWN
Mouse Right Click Up	1005	K_RBUTTONUP
Mouse Left Double Click	1006	K_LBUTTONDBLCLK
Mouse Right Double Click	1007	K_RBUTTONDBLCLK

Appendix D
Character Tables

The following tables illustrate the Currency Symbols, Bullet Characters, Mathematical Symbols, Foreign Language Characters, Greek Characters, and Other Characters together with their corresponding codes and meanings.

Note: In Microsoft Windows, character sets differ based on the font being used. For a character map of each font, use the Windows Character Map application.

Currency Symbols

Symbol	Code	Meaning
¢	155	Cents
£	156	Pounds
¥	157	Yen
ƒ	159	Francs
$	36	Dollars

Bullet Characters

Symbol	Code	Meaning
•	7	BEL
♦	4	EOT
●	249	
•	250	
♥	3	ETX
♣	5	ENQ
♠	6	ACK
♂	11	VT
♀	12	FF
*	42	

Mathematical Symbols

Symbol	Code	Meaning
Å	143	
|	179	Pipe
∈	238	Is an element of
∩	239	Intersection
≡	240	Identical to, congruent
±	241	Plus or minus
≥	242	Greater than or equal to
≤	243	Less than or equal to
⌠	244	Integral - top half
⌡	245	Integral - bottom half
÷	246	Division
≈	247	Nearly equal
°	248	Degree
•	249	Multiplied by
•	250	
√	251	Square root
n	252	Nth power
2	253	Squared (second power)
∞	236	Infinity

Foreign Language Characters

Code	Letter	Code	Letter	Code	Letter	Code	Letter
131	â	136	ê	161	í	225	ß
132	ä Ä 142	137	ë	147	ô	150	û
133	à	138	è	148	ö Ö 153	129	ü Ü 154
160	á	130	é É 144	149	ò	151	ù
134	å Å 143	140	î	162	ó	163	ú
145	æ Æ 146	139	ï	164	ñ Ñ 165	152	ÿ
135	ç Ç 128	141	ì				

Greek Characters

Code	Letter			Name	Transliteration
224	α			Alpha	a
225	β			Beta	b
226	Γ			Gamma	g
235	δ			Delta	d
233	θ			Theta	th
230	μ			Mu	m
227	π			Pi	p
229	σ	Σ	228	Sigma	s
231	τ			Tau	t
237	φ	Φ	232	Phi	ph
234	Ω			Omega	

Other Characters

Symbol	Code	Meaning
ª	166	
º	167	
¿	168	
¡	173	
«	174	
»	175	
‼	19	
¶	20	Paragraph or Return Character
§	21	Section Number
Pt	158	

Appendix E
CA-Clipper Reserved Words

The following tables illustrate CA-Clipper reserved *words* and *functions*. *Reserved words* cannot be used for the names of variables, procedures, or user-defined functions. *Reserved functions* are built into the compiler and cannot be redefined by an application. In addition, any abbreviations of reserved words or functions of four or more characters are also reserved. All identifiers that begin with one or more underscore characters (_) are reserved.

Reserved Words

AADD	ELSEIF	LOCK	SELECT
ABS	EMPTY	LOG	SETPOS
ASC	ENDCASE	LOWER	SPACE
AT	ENDDO	LTRIM	SQRT
BOF	ENDIF	MAX	STR
BREAK	EOF	MIN	SUBSTR
CDOW	EXP	MONTH	TEXTBLOCK
CHR	FCOUNT	PCOL	TIME
CMONTH	FIELDNAME	PCOUNT	TRANSFORM
COL	FILE	PROCEDURE	TRIM
CTOD	FLOCK	PROW	TYPE
DATE	FOUND	RECCOUNT	UPPER
DAY	FUNCTION	RECNO	VAL
DELETED	IF	REPLICATE	VALTYPE
DEVPOS	IIF	RLOCK	WHILE
DOW	INKEY	ROUND	WORD
DTOC	INT	ROW	YEAR
DTOS	LASTREC	RTRIM	
ELSE	LEN	SECONDS	

Appendix F
dBASE Commands and Functions Not Supported by CA-Clipper

Some dBASE commands and functions are not supported by CA-Clipper. CA-Clipper does not support any of the commands that are used primarily in the interactive or "dot prompt" mode. In the interactive mode, you may instantly query the various databases without writing a program, however, CA-Clipper has been designed to compile and execute programs significantly faster than can be accomplished in the interactive mode.

The dBASE commands and functions that are not supported by CA-Clipper are listed in the table below.

CA-Clipper Equivalents of dBASE Commands & Functions

dBASE Command/Function	CA-Clipper Equivalent
APPEND	DBU.EXE
ASSIST	DBU.EXE
BROWSE	BROWSE(), DBEDIT(), TBrowse Class
CHANGE	DBU.EXE
CLEAR FIELDS	n/a
CREATE/MODIFY LABEL	RL.EXE
CREATE/MODIFY QUERY	n/a
CREATE/MODIFY REPORT	RL.EXE
CREATE/MODIFY SCREEN	n/a
CREATE/MODIFY STRUCTURE	DBU.EXE
CREATE/MODIFY VIEW	DBU.EXE
EDIT	DBU.EXE
ERROR()	Error:genCode, Error:osCode, Error:SubCode messages

CA-Clipper Equivalents of dBASE Commands & Functions (cont.)

dBASE Command/Function	CA-Clipper Equivalent
EXPORT TO	n/a
HELP	The Guide To CA-Clipper
IMPORT FROM	n/a
INSERT	n/a
LIST/DISPLAY FILES	DBU.EXE
LIST/DISPLAY HISTORY	The CA-Clipper debugger
LIST/DISPLAY MEMORY	The CA-Clipper debugger
LIST/DISPLAY STATUS	The CA-Clipper debugger
LIST/DISPLAY STRUCTURE	The CA-Clipper debugger
LOAD	BLINKER.EXE
LOGOUT	n/a
MESSAGE()	Error:description message
MODIFY COMMAND	PE.EXE
ON ERROR	ERRORBLOCK()
ON ESCAPE	SET KEY User function
ON KEY	SET KEY User function
RESUME	RETURN false (.F.) from an error handling block if Error:canDefault contains true (.T.)
RETRY	RETURN true (.T.) from an error handling block if Error:canRetry contains true (.T.)
RETURN TO MASTER	BEGIN SEQUENCE...BREAK...END
SET	The CA-Clipper debugger
SET CARRY	n/a
SET CATALOG	n/a
SET COLOR ON | OFF	n/a
SET DEBUG	ALTD()
SET DOHISTORY	The CA-Clipper debugger
SET ECHO	The CA-Clipper debugger
SET ENCRYPTION	n/a

CA-Clipper Equivalents of dBASE Commands & Functions (cont.)

dBASE Command/Function	CA-Clipper Equivalent
SET FIELDS	DBU.EXE
SET HEADING	n/a
SET HELP	n/a
SET HISTORY	n/a
SET MEMOWIDTH	MEMOLINE(), MEMOEDIT(), MLCOUNT()
SET MENUS	n/a
SET MESSAGE	n/a
SET SAFETY	n/a
SET STATUS	n/a
SET TALK	The CA-Clipper debugger
SET TITLE	n/a
SET VIEW	DBU.EXE

Appendix G
Categorized Language Tables

The following tables illustrate CA-Clipper commands, functions, preprocessor directives, and statements categorized by subject.

Character Data Manipulation

Reference Item	Description
ALLTRIM()	Remove leading and trailing spaces from character string
ASC()	Convert a character to its ASCII value
AT()	Return the position of a substring within a string
CHR()	Convert an ASCII code to a character value
CTOD()	Convert a date string to a date value
DBFILTER()	Return the current filter expression as a character string
DELETE TAG	Delete a tag
DESCEND()	Create a descending index key value
DEVOUTPICT()	Write the value of an expression at the current cursor position
DISPBOX()	Define the box border characters
DISPOUT()	Define the display color of an expression
DTOC()	Convert a date value to a character string
EMPTY()	Determine if the result of an expression is empty
HARDCR()	Replace all soft CRs with hard CRs
ISALPHA()	Determine if the leftmost character is alphabetic
ISDISK	Check if a disk drive is available
ISLOWER()	Determine if the leftmost character is a lower case letter
ISUPPER()	Determine if the leftmost character is upper case
LEFT()	Extract a substring beginning with the first character
LEN()	Return the length of a character string or array size
LOWER()	Convert uppercase characters to lowercase
LTRIM()	Remove leading spaces from a character string

Character Data Manipulations (cont.)

Reference Item	Description
MEMOLINE()	Extract a line of text from character string or memo field
MEMOREAD()	Return the contents of a disk file as a character string
MEMOTRAN()	Replace carriage return/line feeds in character strings
ORDBAGNAME()	Return a character string of a specific order
ORDDESTROY()	Remove a specified order from multiple order bags
ORDFOR()	Returns the FOR expression of an order
PAD()	Pad character, date or numeric values with a fill character
RAT()	Return the position of the last occurrence of a substring
RDDSETDEFAULT	Set or return the default RDD driver
REPLICATE()	Return a string repeated a specified number of times
RIGHT()	Return a substring beginning with rightmost character
RTRIM()	Remove trailing spaces from a character string
SET EXACT	Toggle exact matches for character strings
SOUNDEX()	Convert a character string to soundex form
SPACE()	Return a string of spaces
STR()	Convert a numeric expression to a character string
STRTRAN()	Search and replace characters within a character string
STUFF()	Delete and insert characters in a string
SUBSTR()	Extract a substring from a character string
TRANSFORM()	Convert any value into a formatted character string
TRIM()	Remove trailing spaces from a character string
TYPE()	Determine the type of an expression
UPPER()	Convert lowercase characters to uppercase
VAL()	Convert a character number to numeric type
VALTYPE()	Determine the data type returned by an expression

Memo Field Manipulation

Reference Item	Description
HARDCR()	Replace all soft CRs with hard CRs
MEMOEDIT()	Display or edit character strings and memo fields
MEMOLINE()	Extract a line of text from character string or memo field
MEMOREAD()	Return the contents of a disk file as a character string
MEMOTRAN()	Replace carriage return/line feeds in character strings
MEMOWRIT()	Write a character string or memo field to a disk file

Memo Field Manipulation (cont.)

Reference Item	Description
MLCOUNT()	Count the lines in a character string or memo field
MLCTOPOS()	Return byte position based on line and column position
MLPOS()	Determine the position of a line in a memo field
MPOSTOLC()	Return line and column position based on byte position
READINSERT()	Toggle the current insert mode for MEMOEDIT()
SET SCOREBOARD	Toggle the message display from MEMOEDIT()
STRTRAN()	Search and replace characters within a memo field
TYPE()	Determine the type of an expression
VALTYPE()	Determine the data type returned by an expression

Numeric Data Manipulation

Reference Item	Description
ABS()	Return the absolute value of a numeric expression
BIN2I()	Convert a 16-bit signed integer to a numeric value
BIN2L()	Convert a 32-bit signed integer to a numeric value
BIN2W()	Convert a 16-bit unsigned integer to a numeric value
CHR()	Convert an ASCII code to a character value
DESCEND()	Create a descending index key value
DEVPOS()	Move the printhead to a new position
DISPCOUNT()	Return the number of bytes available on a specified disk
EMPTY()	Determine if the result of an expression is empty
EXP()	Calculate e**x
INT()	Convert a numeric value to an integer
I2BIN()	Convert a CA-Clipper numeric to a 16-bit binary integer
LOG()	Calculate the natural logarithm of a numeric value
L2BIN()	Convert a numeric value to a 32-bit binary integer
MAX()	Return the larger of two numeric or date values
MIN()	Return the smaller of two numeric or date values
MOD()	Return dBASE III PLUS modulus of two numbers
ORDCOND()	Specify conditions for Ordering or rebuilding an existing order
ROUND()	Return a value rounded to a specified number of digits
SET EVENTMASK	Return an integer value for keyboard and mouse events

Numeric Data Manipulation (cont.)

Reference Item	Description
SQRT()	Return the square root of a positive number
STR()	Convert a numeric expression to a character string
TRANSFORM()	Convert any value into a formatted character string
TYPE()	Determine the type of an expression
VAL()	Convert a character number to a numeric type
VALTYPE()	Determine the data type returned by an expression
WORD()	Convert CALL command numeric parameters from double to int

Date and Time Data Manipulation

Reference Item	Description
CDOW()	Convert a date value to a character day of the week
CMONTH()	Convert a date to a character month name
CTOD()	Convert a date string to a date value
DATE()	Return the system date as a date value
DAY()	Return the day of the month as a numeric value
DOW()	Convert a date value to a numeric day of the week
DTOC()	Convert a date value to a character string
DTOS()	Convert a date value to a string formatted as yyyymmdd
DESCEND()	Create a descending index key value
EMPTY()	Determine if the result of an expression is empty
MAX()	Return the larger of two numeric or date values
MIN()	Return the smaller of two numeric or date values
MONTH()	Convert a date value to the number of the month
SECONDS()	Return the number of seconds elapsed since midnight
TIME()	Return the system time
TRANSFORM()	Convert any value into a formatted character string
TYPE()	Determine the type of an expression
VALTYPE()	Determine the data type returned by an expression
YEAR()	Convert a date value to the year as a numeric value

Array Manipulation

Reference Item	Description
AADD()	Add a new element to the end of an array
ACHOICE()	Execute a pop-up menu

Array Manipulation (cont.)

Reference Item	Description
ACLONE()	Duplicate a nested or multidimensional array
ACOPY()	Copy elements from one array to another
ADEL()	Delete an array element
ADIR()	Fill a series of arrays with directory information
AEVAL()	Execute a code block for each element in an array
AFIELDS()	Fill arrays with the structure of the current database file
AFILL()	Fill an array with a specified value
AINS()	Insert a NIL element into an array
ARRAY()	Create an uninitialized array of specified length
ASCAN()	Scan an array for a value or until a block returns true (.T.)
ASIZE()	Grow or shrink an array
ASORT()	Sort an array
ATAIL()	Return the highest numbered element of an array
DBRLOCKLIST	Return an array of the current lock list
DBSTRUCT()	Create and initialize private memory variables and arrays
DECLARE	Create an array of directory and file information
DESCEND()	Create a descending index key value
DIRECTORY()	Determine if the result of an expression is empty
EMPTY()	Return the length of a character string or array size
LEN()	Determine the type of an expression
RDDLIST()	Return a one-dimensional array that list the available RDDs
TYPE()	Determine the data type returned by an expression
VALTYPE()	Determine the data type returned by an expression

Code Block Manipulation

Reference Item	Description
AEVAL()	Execute a code block for each element in an array
ASCAN()	Scan an array for a value or until a block returns true
DBEVAL()	Evaluate a code block for each record matching a scope and condition
DESCEND()	Create a descending index key value
EMPTY()	Determine if the result of an expression is empty
ERRORBLOCK()	Post a code block to execute when a runtime error occurs

Code Block Manipulation (cont.)

Reference Item	Description
EVAL()	Evaluate a code block
FIELDBLOCK()	Return a set-get code block for a field variable
FIELDWBLOCK()	Return a set-get block for a field in a given work area
GETDOSETKEY	Process SET KEY code block during Get editing
MEMVARBLOCK()	Return a set-get code block for a given memory variable
READFORMAT()	Return the code block that implements a format
SETKEY()	Assign an action block to a key
TYPE()	Determine the type of an expression
VALTYPE()	Determine the data type returned by an expression

Databases

Reference Item	Description
AFIELDS()	Fill arrays with the structure of the current database file
ALIAS()	Return a specified work area alias
APPEND BLANK	Add a new record to current database file
APPEND FROM	Import records from a database or ASCII file
AVERAGE	Average numeric expressions in the current work area
BOF()	Determine when beginning of file is encountered
CLEAR ALL	Close files and release public and private variables
CLOSE	Close a specific set of files
COMMIT	Perform a solid-disk write for all active work areas
CONTINUE	Resume a pending LOCATE
COPY STRUCTURE	Copy the current (.dbf) structure to a new database file
COPY STRUCTURE EXTENDED	Copy field definitions to a database file
COPY TO	Export records to a new database (.dbf) or ASCII file
COUNT	Tally records to a variable
CREATE	Create an empty structure extended database file
CREATE FROM	Create a new database file from a structure extended file
DBAPPEND()	Add a new record
DBCLEARFILTER()	Clear a logical filter condition
DBCLEARINDEX()	Close any active indices for the current work area

Databases (cont.)

Reference Item	Description
DBCLEARRELATION()	Clear any active relations for the active work area
DBCLOSEALL()	Close all occupied work areas
DBCLOSEAREA()	Close a work area
DBCOMMIT()	Flush pending updates
DBCOMMITALL()	Flush pending updates in all work areas
DBCREATE()	Create a database file from a database structure array
DBCREATEINDEX()	Create index for the database file associated with current work area
DBDELETE()	Mark a record for deletion
DBEVAL()	Evaluate a code block for each record matching a scope and condition
DBF()	Return current alias name
DBFILTER()	Return the current filter expression as a character string
DBGOBOTTOM()	Move to the last logical record
DBGOTO()	Move to a particular record
DBGOTOP()	Move to the first logical record
DBRECALL()	Reinstate a record marked for deletion
DBREINDEX()	Rebuild all active indices associated with the current work area
DBRELATION()	Return the linking expression of a specified relation
DBRLOCK()	Lock the record at the current or specified identity
DBRLOCKLIST()	Return an array of the current lock list
DBRSELECT()	Return the target work area number of a relation
DBRUNLOCK()	Release all or specified record locks
DBSEEK()	Search for a key value
DBSELECTAR()	Change the current work area
DBSETDRIVER()	Set the default database driver
DBSETFILTER()	Set a filter condition
DBSETINDEX()	Open an index
DBSETORDER()	Set the controlling order
DBSETRELATION()	Relate two work areas
DBSKIP()	Move relative to the current record
DBSTRUCT()	Create an array containing the structure of a database file
DBUNLOCK()	Release all locks for the current work area

Databases (cont.)

Reference Item	Description
DBUSEAREA()	Use a database file in a work area
DELETE	Mark records for deletion
DELETED()	Return the deleted status of the current record
DELETE TAG	Delete a tag
DISKNAME	Return the current DOS drive
EMPTY()	Determine if the result of an expression is empty
EOF()	Determine when end of file is encountered
FCOUNT()	Return the number of fields in the current database file
FIELD	Declare database field names
FIELDBLOCK()	Return a set-get code block for a field variable
FIELDGET()	Retrieve the value for a field variable
FIELDNAME()	Return a field name from the current database file
FIELDPOS()	Return the position of a field in a work area
FIELDPUT()	Set the value of a field variable
FIELDWBLOCK()	Return a set-get block for a field in a given work area
FIND	Search an index for a specified key value
FLOCK()	Lock an open and shared database file
FOUND()	Determine if the previous search operation succeeded
GO	Move the record pointer to a specific record
HEADER()	Return the current database file header length
JOIN	Create a new database file by merging records/fields from two work areas
LASTREC()	Determine the number of records in the current database file
LOCATE	Search sequentially for a record matching a condition
LUPDATE()	Return the last modification date of a database file
MEMOSETSUPER()	Set a RDD inheritance chain for the DBFMEMO database driver
ORDBAGEXT()	Return the default order Bag RDD extension of the current work area
ORDBAGNAME()	Return a character string of a specific order
ORDCREATE()	Create an order in an order bag
ORDDESTROY()	Remove a specified order from multiple-order bags
ORDFOR()	Return the FOR expression of an order
ORDKEY()	Return the key expression of a specified order
ORDLISTADD	Add the contents of an order bag to the order list

Databases (cont.)

Reference Item	Description
ORDLISTCLEAR	Clear the current order list
ORDLISTREBUILD()	Rebuild all orders in the order list of the current work area
ORDNAME()	Return the name of an order in the order list
ORDNUMBER()	Return the position of an order in the current order list
ORDSETFOCUS()	Return the order name of the previous controlling order
PACK	Remove deleted records from a database file
RDDLIST()	Return a one-dimensional array that lists the available RDDs
RDDNAME()	Return the name of the RDD active in the current work area
RDDSETDEFAULT()	Set or return the default RDD driver
RECALL	Restore records marked for deletion
RECCOUNT()	Determine the number of records in the current database file
RECNO()	Return the current record number of a work area
RECSIZE()	Determine the record length of a database file
REPLACE	Assign new values to field variables
SEEK	Search an index for a specified key value
SELECT	Change the current work area
SELECT()	Determine the work area number of a specified alias
SET DELETED	Toggle filtering of deleted records
SET EXCLUSIVE	Establish shared or exclusive USE of database files
SET FILTER	Hide records not meeting a condition
SET PATH	Specify the CA-Clipper search path for opening files
SET SOFTSEEK	Toggle relative SEEKing
SET UNIQUE	Toggle the inclusion of nonunique keys into an index
SKIP	Move the record pointer to a new position
SORT	Copy to a database file in sorted order
SUM	Sum numeric expressions to variables
TOTAL	Summarize records by key value to a database file
UPDATE	Update current database file from another database file
USE	Open an existing database and its associated files
USED()	Determine if a database file is in USE
ZAP	Remove all records from the current database file

Indices

Reference Item	Description
DBCLEARIND()	Close indices
DBCREATEIND()	Create an index file
DBCLOSEALL()	Close all occupied work areas
DBCLOSEAREA()	Close a work area
DBREINDEX()	Recreate all active indices
DBSEEK()	Search for a key value
DBSETINDEX()	Open an index
DBSETORDER()	Set the controlling order
DESCEND()	Create a descending index key value
FOUND()	Determine if the previous search operation succeeded
INDEX	Create an index file
INDEXEXT()	Return the default index extension based on the database driver currently linked
INDEXKEY()	Return the key expression of a specified index
INDEXORD()	Return the order position of the controlling index
REINDEX	Rebuild open indices in the current work area
SET INDEX	Open index file(s) in the current work area
SET ORDER	Set a new controlling index
SET UNIQUE	Toggle the inclusion of nonunique keys into an index

Relations

Reference Item	Description
DBCLEAREL()	Clear active relations
DBRELATION()	Return the linking expression of a specified relation
DBRSELECT()	Return the target work area number of a relation
DBSETRELATION()	Relate two work areas
SET RELATION	Relate two work areas by a key value or record number

Program Execution

Reference Item	Description
BEGIN SEQUENCE	Define a sequence of statements for a BREAK
BREAK()	Branch out of a BEGIN SEQUENCE...END construct
CALL	Execute a C or Assembler procedure

Program Execution (cont.)

Reference Item	Description
CANCEL	Terminate program processing
DO	Call a procedure
DO CASE	Execute one of several alternative blocks of statements
DO WHILE	Execute a loop while a condition is true (.T.)
EXTERNAL	Declare a list of procedure or user-defined function names to the linker
FOR	Execute a block of statements a specified number of times
FUNCTION	Declare a user-defined function name and formal parameters
IIF	Return the result of an expression based on a condition
IF	Execute one of several alternative blocks of statements
IF()	Return the result of an expression based on a condition
NOTE	Place a single-line comment in a program file
PARAMETERS	Create private parameter variables
PCOUNT()	Determine the position of the last actual parameter passed
QUIT	Terminate program processing
RETURN	Terminate a procedure, user-defined function or program
RUN	Execute a DOS command or program
SET KEY	Assign a procedure invocation to a key
SET PROCEDURE	Compile procedures/functions into the current .OBJ file
SETKEY()	Assign an action block to a key
SETCANCEL()	Toggle Alt-C as a program termination key
WAIT	Suspend program processing until a key is pressed
WORD()	Convert CALL command numeric parameters from double to int

Memory Variable Handling

Reference Item	Description
ACCEPT	Place keyboard input into a memory variable
CLEAR ALL	Close files and release public and private variables
CLEAR MEMORY	Release all public and private variables
DECLARE	Create and initialize private memory variables and arrays
INPUT	Enter the result of an expression into a variable
LOCAL	Declare and initialize local variables and arrays

Memory Variable Handling (cont.)

Reference Item	Description
MEMVAR	Declare private and public variable names
MEMVARBLOCK()	Return a set-get code block for a given memory variable
PARAMETERS	Create private parameter variables
PRIVATE	Create and initialize private memory variables and arrays
PUBLIC	Create and initialize public memory variables and arrays
RELEASE	Delete public and private memory variables
RESTORE	Retrieve memory variables from a memory (.mem) file
SAVE	Save variables to a memory (.mem) file
STATIC	Declare and initialize static variables and arrays
STORE	Assign a value to one or more variables

Debugging and Error Handling

Reference Item	Description
ALTD()	Invoke the CA-Clipper debugger
BREAK()	Branch out of a BEGIN SEQUENCE...END construct
DOSERROR()	Return the DOS error number
ERRORBLOCK()	Post a code block to execute when a runtime error occurs
ERRORLEVEL()	Set the CA-Clipper return code
FERROR()	Test for errors after a binary file operation
NETERR()	Determine if a network command has failed
OUTERR()	Write a list of values to the standard error device
PROCLINE()	Return the source line number of the current or previous activation
PROCNAME()	Return the name of the current or previous procedure or user-defined function
READKILL()	Determine if the current READ should be exited

Low-level File Handling

Reference Item	Description
DIRCHANGE	Change the current DOS directory
DIRMAKE	Create a specified directory
DIRREMOVE	Remove a specified directory
DISKCHANGE	Change the current DOS disk drive

Low-level File Handling (cont.)

Reference Item	Description
DISKNAME	Return the current DOS drive
DISKSPACE()	Return the number of bytes available on a specified disk
DOSERROR()	Return the DOS error number
FCREATE()	Return the file handle as a numeric value
FCLOSE()	Close an open binary file and write DOS buffers to disk
FCREATE()	Create and/or truncate a binary file to zero length
FERASE()	Delete a file from disk
FERROR()	Test for errors after a binary file operation
FOPEN()	Open a binary file
FREAD()	Read characters from a binary file into a buffer variable
FREADSTR()	Read characters from a binary file
FSEEK()	Set a binary file pointer to a new position
FWRITE()	Write to an open binary file
HEADER()	Return the current database file header length

Data Input and Output

Reference Item	Description
?/??	Display one or more values to the console
@...BOX	Draw a box on the screen
@...CHECKBOX	Create a new checkbox Get object and display it to the screen
@...CLEAR	Clear a rectangular region of the screen
@...GET	Create a new Get object and display it on the screen
@...LISTBOX	Create a new list box Get object and display it to the screen
@...PROMPT	Paint a menu item and define a message
@...PUSHBUTTON	Create a new pushbutton Get object and display it to the screen
@...RADIOGROUP	Create a new radio button group and display it to the screen
@...SAY	Display data at a specified screen or printer row and column
@...TBROWSE	Create a new TBrowse Get object and display it to the screen
@...TO	Draw a single- or double-line box
ACHOICE()	Execute a pop-up menu
ALERT()	Display a simple modal dialog box
BROWSE()	Browse records within a window
CLEAR GETS	Release Get objects from the current *GetList* array

Data Input and Output (cont.)

Reference Item	Description
CLEAR SCREEN	Clear the screen and home the cursor
COL()	Return the screen cursor column position
DBEDIT()	Browse records in a table format
DEVOUT()	Write a value to the current device
DISPBEGIN()	Begin buffering screen output
DISPBOX()	Display a box on the screen
DISPEND()	Display buffered screen updates
DISPLAY	Display records to the console
DISPOUT()	Write a value to the display
GETACTIVE()	Return the currently active Get object
GETAPPLYKEY	Apply a key to a Get object
GETPOSTVALIDATE	Postvalidate the current Get object
GETPREVALIDATE	Prevalidate a Get object
GETREADER	Execute standard READ behavior for a Get object
GBMPLOAD()	Allow you to load one or more BMP or ICO files into memory
GELLIPSE()	Draw an ellipse or circle
GFNTERASE()	Erase a specified font from memory
GFNTLOAD()	Load a font file into memory
GFNTSET()	Set an already loaded font as active
GFRAME()	Draw box frames with a 3D look
GGETPIXEL()	Get the color information for a specific pixel
GLINE()	Draw lines on the screen
GMODE()	Change the video mode or retrieve information about the current video mode
GPOLYGON()	Create a polygon if you pass an array of coordinates which make up the polygon
GPUTPIXEL()	Draw and changes the color of a specific pixel on the screen
GRECT()	Draw filled or empty rectangles on the screen
GSETCLIP()	Limit the active display to a portion of the screen
GSETEXCL()	Prevent output from being displayed in a defined region of the screen
GSETPAL()	Change a color's basic component values
GWRITEAT()	Display text in graphic mode without affecting the background

Data Input and Output (cont.)

Reference Item	Description
LABEL FORM	Display labels to the console
LIST	List records to the console
MCOL	Determine the mouse cursor's screen column position
MDBLCLK	Determine and optionally change the mouse's double-click speed threshold
MENU TO	Execute a lightbar menu for defined PROMPTs
MHIDE()	Hide the mouse pointer
MLEFTDOWN	Determine the press status of the left mouse button
MPRESENT	Determine if a mouse is present
MRESTSTATE	Re-establish the previous state of the mouse
MRIGHTDOWN	Determine the status of the right mouse button
MROW	Determine a mouse cusor's screen row position
MSAVESTATE	Save the current state of the mouse
MSETBOUNDS	Set screen boundaries for the mouse cursor
MSETCURSOR	Determine a mouse's visibility
MSETPOS	Set a new position for the mouse cursor
MSETCLIP()	Control mouse pointer movements
MSHOW()	Display the mouse pointer
MSTATE()	Return information on the mouse state
OUTSTD()	Write a list of values to the standard output device
QOUT()	Display a list of expressions to the console
READ	Activate full-screen editing mode using Get objects
READEXIT()	Toggle Up arrow and Down arrow as READ exit keys
READINSERT()	Toggle the current insert mode for READ and MEMOEDIT()
READKEY()	Determine what key was used to terminate a READ
READMODAL()	Activate a full-screen editing mode for a *GetList*
READUPDATED()	Determine if any Get changed during a READ
READVAR()	Return the current GET/MENU variable name
REPORT FORM	Display a report to the console
RESTORE SCREEN	Display a saved screen
RESTSCREEN()	Display a saved screen region to a specified location
ROW()	Return the screen row position of the cursor

Data Input and Output (cont.)

Reference Item	Description
SAVE SCREEN	Save current screen to a buffer or variable
SAVESCREEN()	Save a screen region for later display
SCROLL()	Scroll a screen region up or down
SET CENTURY	Modify the date format to include or omit century digits
SET COLOR	Define screen colors
SET CONFIRM	Toggle required exit key to terminate GETs
SET CONSOLE	Toggle console display to the screen
SET CURSOR	Toggle the screen cursor on or off
SET DATE	Set the date format for input and display
SET DECIMALS	Set the number of decimal places displayed
SET DELIMITERS	Toggle or define GET delimiters
SET DEVICE	Direct @...SAYs to the screen or printer
SET EPOCH	Control the interpretation of dates with no century digits
SET ESCAPE	Toggle Esc as a READ exit key
SET EVENTMASK	Return an integer value for keyboard events
SET FIXED	Toggle fixing of the number of decimal digits displayed
SET FORMAT	Activate a format when READ is executed
SET INTENSITY	Toggle enhanced display of GETs and PROMPTs
SET MESSAGE	Set the @...PROMPT message line row
SET SCOREBOARD	Toggle the message display from READ or MEMOEDIT()
SET WRAP	Toggle wrapping of the highlights in MENUs
SETBLINK()	Toggle asterisk (*) interpretation in SET COLOR
SETCOLOR()	Return the current colors and optionally set new colors
SETCURSOR()	Set the cursor shape
SETMODE()	Change display mode to specified number of rows and columns
SETPOS()	Move the cursor to a new position
SET VIDEOMODE	Changes the current display to text mode and different graphic modes
TEXT	Display a literal block of text
TRANSFORM()	Convert any value into a formatted character string
UPDATED()	Determine if any GET changed during a READ

Printing

Reference Item	Description
?/??	Display one or more values to the console
@...SAY	Display data at a specified screen or printer row and column
DEVOUTPICT()	Write the value of an expression at the current printhead position
DEVPOS()	Move the printhead to a new position
DISPLAY	Display records to the console
EJECT	Advance the print head to top of form
ISPRINTER()	Determine whether the LPT1 is ready
LABEL FORM	Display labels to the console
LIST	List records to the console
OUTSTD()	Write a list of values to the standard output device
PCOL()	Return the current column position of the printhead
PROW()	Return the current row position of the printhead
QOUT()	Display a list of expressions to the console
REPORT FORM	Display a report to the console
SET MARGIN	Set the page offset for all printed pages
SET PRINTER	Toggle echo of console output to the printer or set the destination of printed output
SETPRC()	Set PROW() and PCOL() values
TYPE	Display the contents of a text file

Keyboard Entry

Reference Item	Description
CLEAR TYPEAHEAD	Empty the keyboard buffer
INKEY()	Extract a character from the keyboard buffer
KEYBOARD	Stuff a string into the keyboard buffer
LASTKEY()	Return the INKEY() value of the last key extracted from the keyboard buffer
NEXTKEY()	Read the pending key in the keyboard buffer
SET EVENTMASK	Return an integer value for keyboard events
SET TYPEAHEAD	Set the size of the keyboard buffer

Environment

Reference Item	Description
CURDIR()	Return the current DOS directory
DISKSPACE()	Return the space available on a specified disk
FKLABEL()	Return function key name
FKMAX()	Return number of function keys as a constant
GETENV()	Retrieve the contents of a DOS environment variable
ISCOLOR()	Determine if the current computer has color compatibility
MAXCOL()	Determine the maximum visible screen column
MAXROW()	Determine the maximum visible screen row
MEMORY()	Determine the amount of available free pool memory
MENUMODAL	Activate a top bar menu
NETERR()	Determine if a network command has failed
NETNAME()	Return the current workstation identification
NOSNOW()	Toggle snow position
OS()	Return the operating system name
RDDSETDEFAULT	Set or return the default RDD driver
SET()	Inspect or change a global setting
SET BELL	Toggle automatic sounding of the bell during full-screen operations
SET COLOR	Define screen colors
SET DEFAULT	Set the CA-Clipper default drive and directory
SET FUNCTION	Assign a character string to a function key
SET TYPEAHEAD	Set the size of the keyboard buffer
TONE()	Sound a speaker tone for a specified frequency and duration
VERSION()	Returns CA-Clipper version

File Management

Reference Item	Description
ADIR()	Fill a series of arrays with directory information
CLOSE	Close a specific set of files
COPY FILE	Copy a file to a new file
CURDIR()	Return the current DOS directory
DBCREATEIND()	Create an index file

File Management (cont.)

Reference Item	Description
DELETE FILE	Remove a file from disk
DIR	Display a listing of files from a specified path
DIRECTORY()	Create an array of directory and file information
DIRMAKE	Create a specified directory
DIRREMOVE	Remove a specified directory
DISKCHANGE	Change the current DOS disk drive
DISKSPACE()	Return the space available on a specified disk
ERASE	Remove a file from disk
EOF()	Determine when end of file is encountered
FILE()	Determine if files exist in the CA-Clipper default directory or path
FLOCK()	Lock an open and shared database file
FRENAME()	Change the name of a file
INDEX	Create an index file
RENAME	Change the name of a file
SET ALTERNATE	Echo console output to a text file
SET DEFAULT	Set the CA-Clipper default drive and directory
SET PATH	Specify the CA-Clipper search path for opening files
TYPE	Display the contents of a text file

Networking

Reference Item	Description
DBRLOCK	Lock the record at the current or specified identity
DBRUNLOCK	Release all or specified record locks
DBSETRELATION()	Relate two work areas
DBUNLOCK()	Release all locks for the current work area
DBUNLOCKALL()	Release all locks for all work areas
FLOCK()	Lock an open and shared database file
NETERR()	Determine if a network command has failed
NETNAME()	Return the current workstation identification
ORDBAGEXT()	Return the default order bag extension of the current work area
ORDLISTREBUILD()	Rebuild all orders in the order list of the current work area

Networking (cont.)

Reference Item	Description
RLOCK()	Lock the current record in the active work area
SET EXCLUSIVE	Establish shared or exclusive USE of database files
UNLOCK	Release file record locks set by the current user

Preprocessor Directives

Directive	Description
#command/#translate	Specify a user-defined command or translation directive
#define	Define a manifest constant or pseudofunction
#error	Generate a compiler error and display a message
#ifdef	Compile a section of code if an identifier is defined
#ifndef	Compile a section of code if an identifier is undefined
#include	Include a file into the current source file
#stdout	Outputs the literal text to the standard output device
#undef	Remove a #define definition
#xcommand/#xtranslate	Specify a user-defined command or translation directive

String Operators

Operator	Description
+	Concatenate
-	Concatenate without intervening spaces

Mathematical Operators

Operator	Description
+	Addition or Unary Positive
-	Subtraction or Unary Negative
*	Multiplication
/	Division
%	Modulus
** or ^	Exponentiation

Relational Operators

Operator	Description
<	Less than
>	Greater than
=	Equal
==	Exactly equal for character; equal for other data types
<>, #, or !=	Not equal
<=	Less than or equal
>=	Greater than or equal
$	Is contained in the set or is a subset of

Logical Operators

Operator	Description
.AND.	Logical and
.OR.	Logical or
.NOT. or !	Logical negate

Assignment Operators

Operator	Description
=	Assign
:=	Inline assign
+=	Inline add (or concatenate) and assign
-=	Inline subtract (or concatenate) and assign
*=	Inline multiply and assign
/=	Inline divide and assign
**= or ^=	Inline exponentiate and assign
%=	Inline modulus and assign

Increment and Decrement Operators

Operator	Description
++	Prefix or postfix increment
--	Prefix or postfix decrement

Special Operators

Operator	Description
:	Send a message to an object
()	Function or grouping
[]	Array element
{}	Constant array definition
->	Alias identifier
&	Compile and execute
@	Pass by reference

Appendix H
Obsolete Language Items

In the *Reference Guide*, the asterisk symbol is used to indicate items in the language that are obsolete or that exist for compatibility with previous releases of CA-Clipper. As you begin to incorporate new product features into your existing applications, it would be advisable to review your code to determine any obsolete items that you might be using. This table lists each obsolete language item with its recommended replacement and should be helpful to you in updating your programs.

Warning: *Obsolete items are not in keeping with the current CA-Clipper programming philosophy, and we strongly discourage their use as they may not be supported in future releases of CA-Clipper.*

Obsolete Language Items

Obsolete	Recommended
ADIR() function	DIRECTORY() function
AFIELDS() function	DBSTRUCT() function
CALL command	Extend API
CANCEL command	QUIT command
CLEAR ALL command	CLOSE or RELEASE command
DBEDIT() function	TBrowse class
DBF() function	ALIAS() function
DECLARE statement	PRIVATE statement
DIR command	DIRECTORY() function
DO statement	Use function-calling syntax
EXTERNAL statement	REQUEST statement
FIND command	SEEK command
FKLABEL() function	Use constant value F*n* where *n* ranges from one to 40
FKMAX() function	Use constant value of 40
MOD() function	Modulus operator (%)

Obsolete Language Items (cont.)

Obsolete	Recommended
NOTE command	C-style comment indicators, /*...*/ and //
READKEY() function	LASTKEY() function
RECCOUNT() function	LASTREC() function
RESTORE SCREEN command	RESTSCREEN() function
SAVE SCREEN command	SAVESCREEN() function
SET COLOR command	SETCOLOR() function
SET EXACT command	Not recommended
SET EXCLUSIVE command	USE command with EXCLUSIVE and SHARED clauses
SET FORMAT command	Not recommended
SET PROCEDURE command	Compiler script files (.clp)
SET UNIQUE command	INDEX command with UNIQUE clause
STORE command	Inline assignment operator (:=)
TEXT command	? or @...SAY command
WAIT command	@...GET...READ command or INKEY() function
WORD() function	Extend API

Appendix I
CA-Clipper Technical Specifications

The following table specifies parameter limits imposed by the CA-Clipper compiler, benchmarks, and definitions, together with their values.

Parameter Limits and Their Values

Description	Specification
Significant number of bytes in a variable name	10
Significant number of bytes in FUNCTION name	10
Significant number of bytes in PROCEDURE name	10
Significant number of bytes in TAG name	10
Significant number of NUMERIC decimal places	16
Significant overall length of a NUMERIC	32
NUMERIC data type value range	10^{-308} to 10^{308}
Maximum number of bytes allowed in a FIELD name	10
Maximum number of FIELDS in a .dbf (DBFNTX RDD)	1024
Maximum number of FIELDS in a .dbf (DBFNDX RDD)	1024
Maximum number of FIELDS in a .dbf (DBFMDX RDD)	1024
Maximum number of FIELDS in a .dbf (DBFCDX RDD)	1024
Maximum number of records in a .dbf	1 billion
Maximum size of a memo file for .dbt (DBFNTX RDD)	32 megabytes
Maximum size of a memo file for .fpt (DBFCDX RDD)	4.2 gigabytes[1]

Parameter Limits and Their Values (cont.)

Description	Specification
Maximum size of a BLOB file (DBFMEMO RDD)	(storage space)[2]
Maximum length of a CHARACTER string	65,517 bytes
Maximum length of a CHARACTER string for a macro	254 bytes[3]
Maximum size string, or CHARACTER or MEMO FIELD	65,517 bytes
Default DATE FIELD size	8 bytes
Default LOGIC FIELD size	1 byte
Default database MEMO FIELD pointer size	10 bytes
Maximum NUMERIC overall length	32
Maximum NUMERIC decimal precision	16
Default DATE() range: lowest date allowed	01/01/0100
Default DATE() range: highest date allowed	12/31/2999
Maximum number of elements in an array	4096
Maximum number of elements in a subarray	4096
Maximum number of METHODs in class instantiation	32
Maximum symbol table size in an .OBJ file	64K bytes
Maximum buffer size: TBrowse Class Object table	2048 bytes
Maximum buffer size: TRANSFORM(), REPLICATE()	2048 bytes
Maximum key expression length: .ntx (DBFNTX RDD)	256 bytes
Maximum key expression length: .ndx (DBFNDX RDD)	256 bytes
Maximum key expression length: .mdx (DBFMDX RDD)	220 bytes
Maximum key expression length: .cdx (DBFCDX RDD)	255 bytes
Maximum number of order bags/area: (DBFNTX RDD)	15 files
Maximum number of order bags/area: (DBFNDX RDD)	15 files
Maximum number of order bags/area: (DBFMDX RDD)	15 files
Maximum number of order bags/area: (DBFCDX RDD)	Unlimited[4]
Maximum number of tags/order bags: (DBFNTX RDD)	1 tag
Maximum number of tags/order bags: (DBFNDX RDD)	1 tag
Maximum number of tags/order bags: (DBFMDX RDD)	47 tags
Maximum number of tags/order bags: (DBFCDX RDD)	Unlimited[5]

Parameter Limits and Their Values (cont.)

Description	Specification
Maximum number of file handles in an application	250[6]
Maximum number of work areas in an application	250[7]
Maximum number of record locks in a work area	4096[8]
Maximum number of concurrent record locks/work area	1
Maximum FOR expression length: .ntx (DBFNTX RDD)	256 bytes
Maximum FOR expression length: .ndx (DBFNDX RDD)	N/A
Maximum FOR expression length: .mdx (DBFMDX RDD)	261 bytes
Maximum FOR expression length: .cdx (DBFCDX RDD)	255 bytes

Floating Point Specifications

- Floating point based on MSC 8.0 LLIBCE.LIB math emulation
- Floating point operations are in MSC 8.0 IEEE format
- Floating point NUMERIC overflow value displayed by asterisks
- Floating point errors truncated by conversion to STR()
- Floating point error cascade produces math runtime errors

NOTES

1 Limited to available disk space and FoxPro 2.6 compatibility
2 Limited to available disk space and maximum size supported by DOS (4.2 gigabytes)
3 Greater complexity will reduce the overall supportable length
4 Limited to buffers, disk storage space, and memory
5 Limited to buffers, disk storage space, 249 file handles (1 database must be open), and memory
6 DOS 3.30 and higher permit 255 file handles; 5 are used by DOS
7 Limited to available file handles
8 DBRLOCKLIST() and DIRECTORY() arrays limited to 4096 elements

Index

D

E

N

O

P

R

S

T

U

V

W