Symantec C++

Appendixes

Part Seven

Expression Evaluation

A

An expression comprises operands and operators, such as constants, variables, and functions. You can specify variables and functions using their symbolic names defined in your program. Symantec C++ supports standard language operators. This manual does not provide a complete discussion of language expressions.

Entering Expressions

The following IDDE operations prompt you to enter an expression:

- Modifying a variable
- Modifying a memory location
- Modifying a CPU register
- Setting a conditional breakpoint
- Evaluating an expression
- Specifying an array index
- Specifying a memory address
- Specifying a live memory expression

When one of these operations is executed, the debugger displays the **Expression** dialog box, shown in Figure A-1. For example, to modify a variable in the Data/Object window, you can input a new value by entering an expression. The debugger evaluates this expression and assigns the result to the variable.

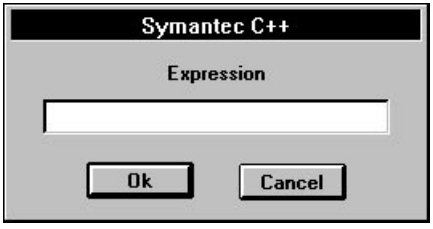


Figure A-1 Expression dialog box

Symbols and Their Scope

A symbol is the name of a variable, procedure, module, or enumerated symbol in your program. You declare symbols in the scope of a procedure or module. When you use a symbol in an expression, the debugger determines its scope based on the module, procedure, and line where the current instruction is located.

The IDDE expression evaluator tries to match an entered symbol against:

- A symbol in the current procedure
- The current procedure's name
- A global symbol in the current module
- The current module's name
- Any other module's name

If you want the debugger to search for a symbol in other procedures or modules, you must qualify the symbol by using a scope override (described below).

Scope override

You can override the current scope where the debugger looks for a symbol by qualifying the symbol with a module or a procedure name. To override the current scope, use the syntax:

[*ModuleName.*] [*ProcName.*] *SymbolName*

The debugger looks for the symbol *SymbolName* in the procedure *ProcName* declared in the module *ModuleName*. For example, if you enter:

```
InOut.WriteString.i
```

the debugger tries to find the symbol `i` in the scope of the procedure `WriteString` declared in the module `InOut`.

If you do not include the module name, the debugger uses the current module (the module containing the current instruction).

For example, if you enter:

```
WriteString.i
```

the debugger tries to find the symbol `i` in the scope of the procedure `WriteString` declared in the module displayed in the Source window.

If you specify a module name but not a procedure name, the debugger uses the global scope of the module specified. For example, if you enter:

```
InOut.i
```

the debugger tries to find the symbol `i` in the global scope of the module `InOut`.

Register symbols

To evaluate processor register values, use the symbols listed in the following tables:

Table A-1 Processor registers, 16-bit

Symbol	Register
`AX` or `ax`	AX
`BX` or `bx`	BX
`CX` or `cx`	CX
`DX` or `dx`	DX
`SI` or `si`	SI
`DI` or `di`	DI
`SS` or `ss`	SS
`DS` or `ds`	DS
`CS` or `cs`	CS
`ES` or `es`	ES
`SP` or `sp`	SP
`BP` or `bp`	BP
`IP` or `ip`	IP
`_F` or `_f`	Flags
`FS`	FS available only when debugging in 32-bit mode
`GS`	GS available only when debugging in 32-bit mode

Table A-2 Processor registers, 32-bit (available only when debugging in 32-bit mode)

Symbol	Register
`EAX` or `eax`	EAX
`EBX` or `ebx`	EBX

Table A-2 Processor registers, 32-bit (available only when debugging in 32-bit mode) *(Continued)*

Symbol	Register
`ECX` or `ecx`	ECX
`EDX` or `edx`	EDX
`ESI` or `esi`	ESI
`EDI` or `edi`	EDI
`ESP` or `esp`	ESP
`EBP` or `ebp`	EBP
`EIP` or `eip`	EIP

Table A-3 Floating point stack registers

Symbol	Floating point stack
`FP0` or `fp0`	ST(0)
`FP1` or `fp1`	ST(1)
`FP2` or `fp2`	ST(2)
`FP3` or `fp3`	ST(3)
`FP4` or `fp4`	ST(4)
`FP5` or `fp5`	ST(5)
`FP6` or `fp6`	ST(6)
`FP7` or `fp7`	ST(7)

Operators

The IDDE supports standard C and C++ operators in expressions. These operators, described in the following sections, have the same precedence within the debugger's expression evaluator as they do in C and C++.

In addition to the standard operators in C and C++, the IDDE supports the colon operator (`:`). The colon operator joins a *segment*`:`*offset* pair of unsigned integers to specify an address value. This operator has the same priority as the unary operators.

The IDDE supports the standard C and C++ operators listed below, in descending order of precedence:

Primary

```
()  []  ->  . this  ::
```

Unary

```
*  &  -  !  ~  ++  --  sizeof
```

Binary

```
.*  ->*
*  /  %
+  -
>>  <<
>  <  >=  <=
==  !=
&
^
|
&&
||
```

Assignment

```
=  +=  -=  *=  /=  %=  >>=  <<=
&=  ^=  |=
```

C expressions in the IDDE also may include typecasts of the form:

(*type-name*) *expression*

For C++, the above typecast is valid only for built-in types.

Because the debugging information does not associate line numbers with local scopes, the IDDE cannot distinguish variables declared in a local scope. For example, in line 7 of the following source code:

```
1    int i;
2    proc()
3    {
4          int i;
5          if  (i){
6              int i;
7              i=5;
8           }
9    }
```

The intent is for the variable `i` in `i=5` to refer to the `i` in line 6, but the IDDE will associate it with the `i` in line 4.

Considerations When Using C++ Expressions

This section describes considerations for working with the IDDE expression evaluator and C++ expressions.

The expression evaluator generally expects the same syntax as the compiler.

Access to class members

All members of a class object are accessible, no matter which type of access control is imposed (public, protected, or private), or if the object is a member of a base class (embedded object).

For example, if class `Customer` has a private member `name`, enter the following into the IDDE expression evaluator:

```
Customer::name
```

The expression evaluator provides the value of `name` in an output dialog box. You also can access members of an object using a pointer to the object.

For example, if the `Salesperson` class defines a virtual function named `totalSales`, redefined in the class inherited from `Salesperson`, `totalSales` can be called using a pointer to `Salesperson`:

```
salePtr->totalSales()
```

Ambiguous references

When an expression makes an ambiguous reference to a member name, you must qualify it with the class name. For example, the class `Resistor` is defined as follows:

```
1  class Parts
2  {
3     unsigned int specs;
4  }  resistorParts;
5
6  class Components
7  {
8     unsigned int specs;
9  }  resistorComponents;
10
11 class Resistor:
12 public Parts, public Components;
13 {
14    int name;
15 }  resistor;
16    ...
```

Assume that class `Resistor` inherits from the `Parts` and `Components` classes. Both `Parts` and `Components` define a

member item called `specs`. If `largeResistor` is an instance of class `Resistor`, the following expression is ambiguous:

```
largeResistor.specs
```

To resolve this problem, use either of the following expressions:

- `largeResistor.Parts::specs`
- `largeResistor.Components::specs`

Constructors and destructors

The IDDE expression evaluator calls constructor or destructor functions just as it calls normal functions. Functions that declare or return local objects are valid expressions, and they return the address of the resulting object.

Note

The IDDE expression evaluator does not let you call the `new` and `delete` operators.

Overloaded functions

The IDDE expression evaluator supports calling overloaded functions only if an exact match exists or if the match does not need a conversion involving the construction of an object. For example, the overloaded function `Print` is defined as below:

```
1  Print(int x)
2  {
3     ...
4  }
5
6  Print(float y)
7  {
8     ...
9  }
```

In this case, both of the following expressions for the IDDE expression evaluator are valid:

- `Print(5.5);`
- `Print(5);`

Overloaded operators

IDDE's expression evaluator lets you call an overloaded operator for user-defined types. For example, suppose you define a class that represents arrays as follows:

```
class Array array1, array2, array3, array4
```

If this class has a member function that overloads the + operator, then you can evaluate the following:

```
array1 = array2 + array3 + array4
```

Make sure that no variables overflow during evaluation. The expression evaluator automatically creates temporary objects as needed to store the intermediate values and discards them after it performs the evaluation.

Function and Procedure Calls

The IDDE lets you execute function and procedure calls defined in your program when evaluating an expression. Use this feature to:

- Insert a call in your program.
- Call a procedure that you define to display your program's data in a customized format, such as a graph, a table, or a statistical chart.
- Use a function call as a condition for a breakpoint. When the breakpoint is reached, the function is called. Depending on the return value, the debugger stops your program's execution.

To include a function call in an expression, use the syntax:

```
procedureName([param1[,param2]...])
```

You can specify the procedure parameters as expressions. The IDDE passes all parameters by value.

Evaluating expressions with function calls

The IDDE evaluates expressions, except for function and procedure calls, by interpretation. When the interpreter encounters a function call, it saves the application's registers and pushes its own evaluation stack onto the application program's stack. Next, the debugger orders the application to begin executing at the function's entry

point. If the **Flip Screen** command is on, the application's screen comes to the foreground. When the application's procedure returns, the debugger takes control and restores the application's register state. During the evaluation of a procedure or a function, the debugger ignores breakpoints and watchpoints. It takes the return value and continues evaluating the expression, if necessary.

Side effects of expression evaluation

When including function or procedure calls in the IDDE expressions, beware of possible side effects caused when you evaluate a function that results in changes to your program's data. Such changes could alter the behavior of your program after it resumes execution.

Expression Evaluation Errors

The IDDE normally evaluates an expression and displays the result after you press Enter. However, with the **Set Conditional Breakpoint** command, the IDDE does not evaluate an expression until it reaches the breakpoint. If a run-time error occurs during the evaluation of a conditional breakpoint expression, the IDDE assumes that the expression is false and does not display an error message.

When the IDDE finds a syntax or semantic error in an expression, it displays the error message in the title bar of the debugger's main window.

A Expression Evaluation

IDDE Settings and Command-Line Options B

This appendix provides a series of figures that show the relationship between the settings on the Build page of the IDDE **Project Settings** dialog box and the command line options that you pass to the command line utilities. To opens the **Project Settings** dialog box, select **Project Settings** from the IDDE **Project** menu. Then click on the Build tab to select the Build page.

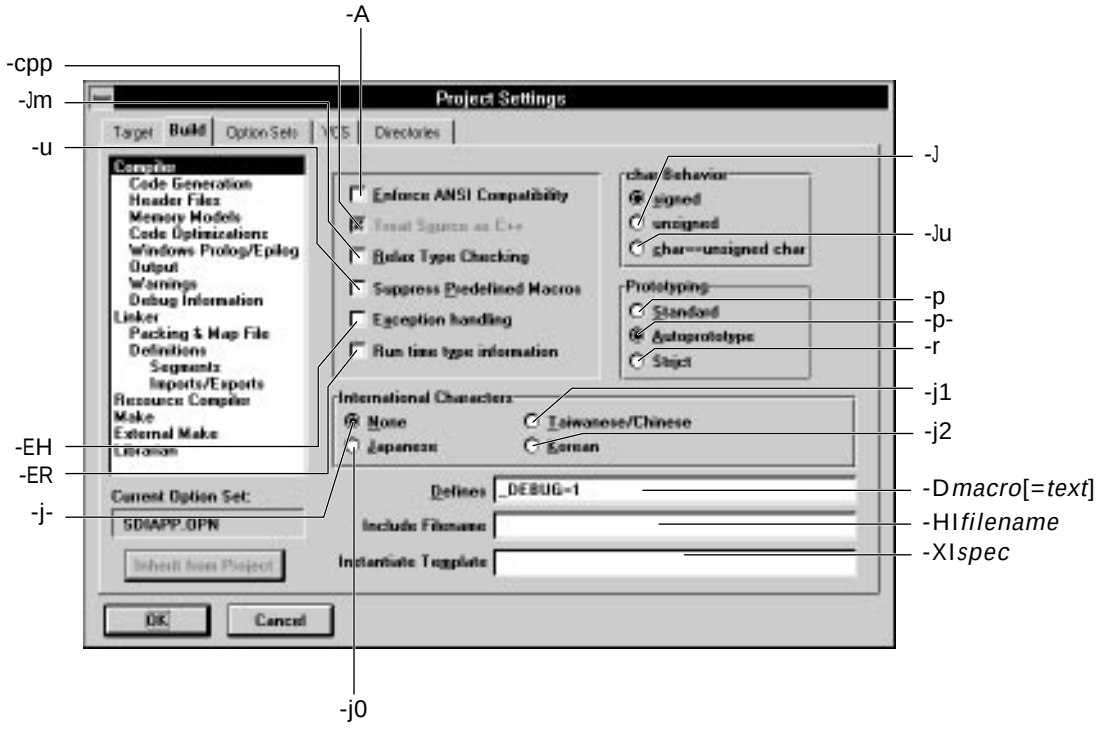
The Build page comprises different subpages, each one containing specific options for the compiler, linker, `.def` file, resource compiler, and librarian utility. Chapter 16, "More about Project Build Settings," describes in detail each of the subpages of the Build page.

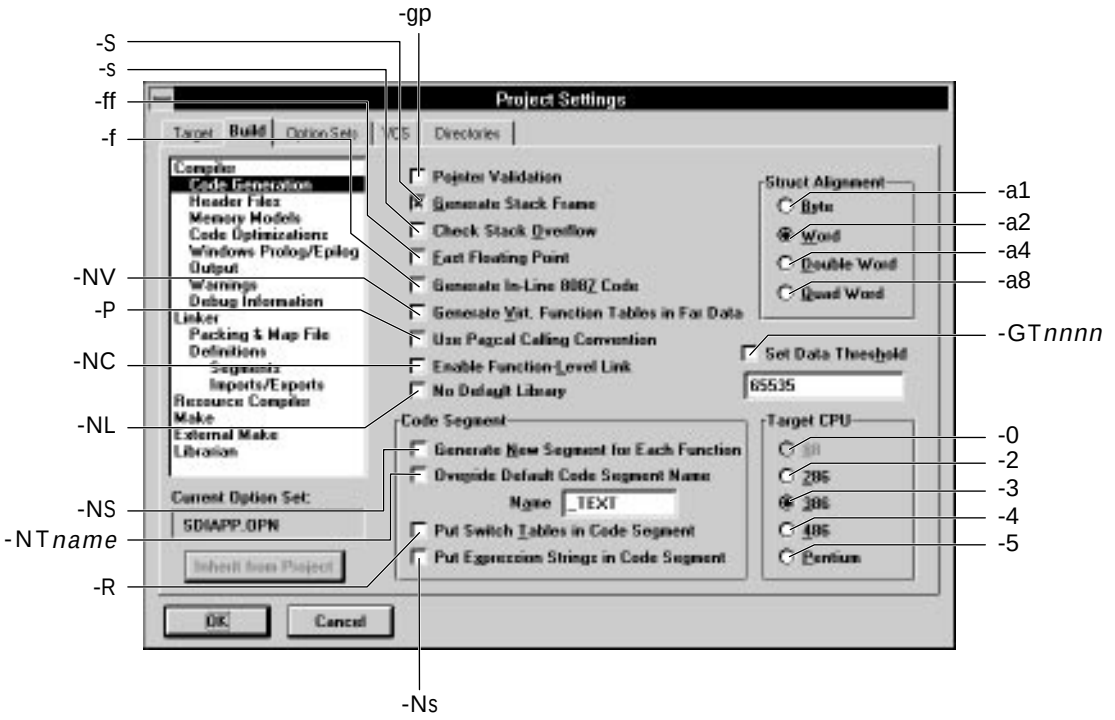
For detailed information on how these options affect the compilation and linking of your code, refer to the *Symantec C++ Compiler and Tools Guide*.

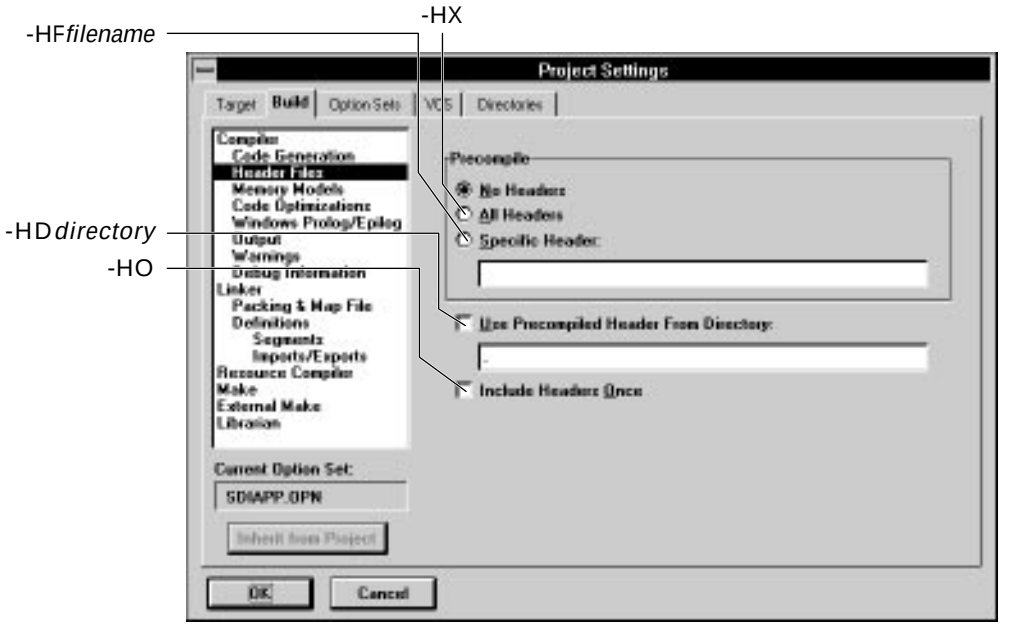
This chapter consists of a series of figures showing subpages of the Build page. Each figure has callouts to each option that has a command-line equivalent.

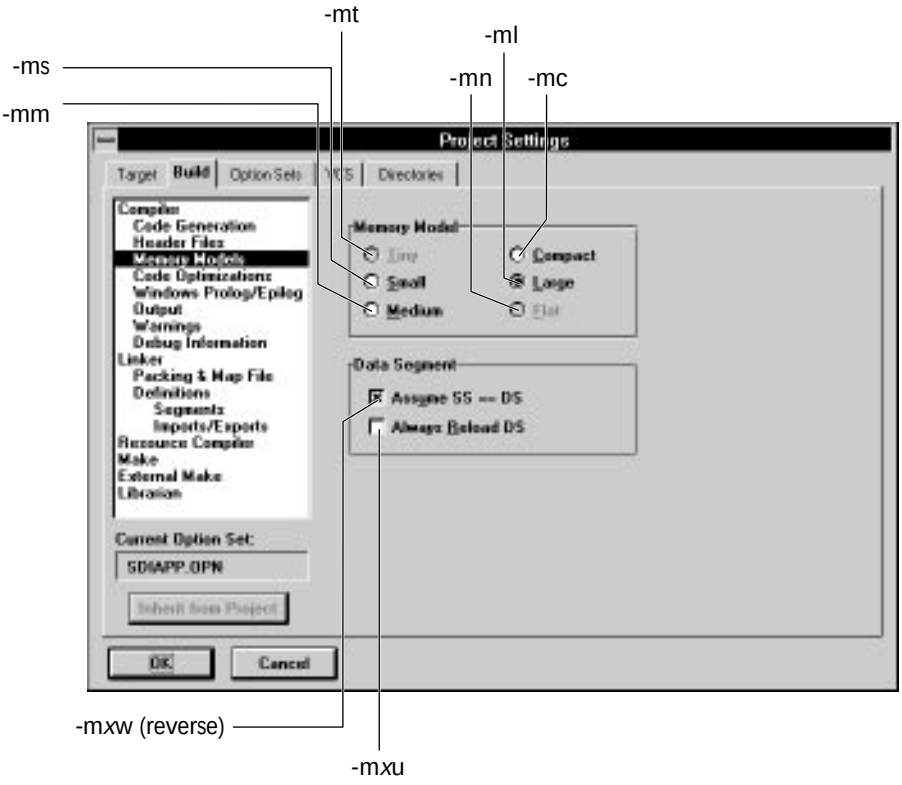
Mapping IDDE Options to Command-Line Parameters

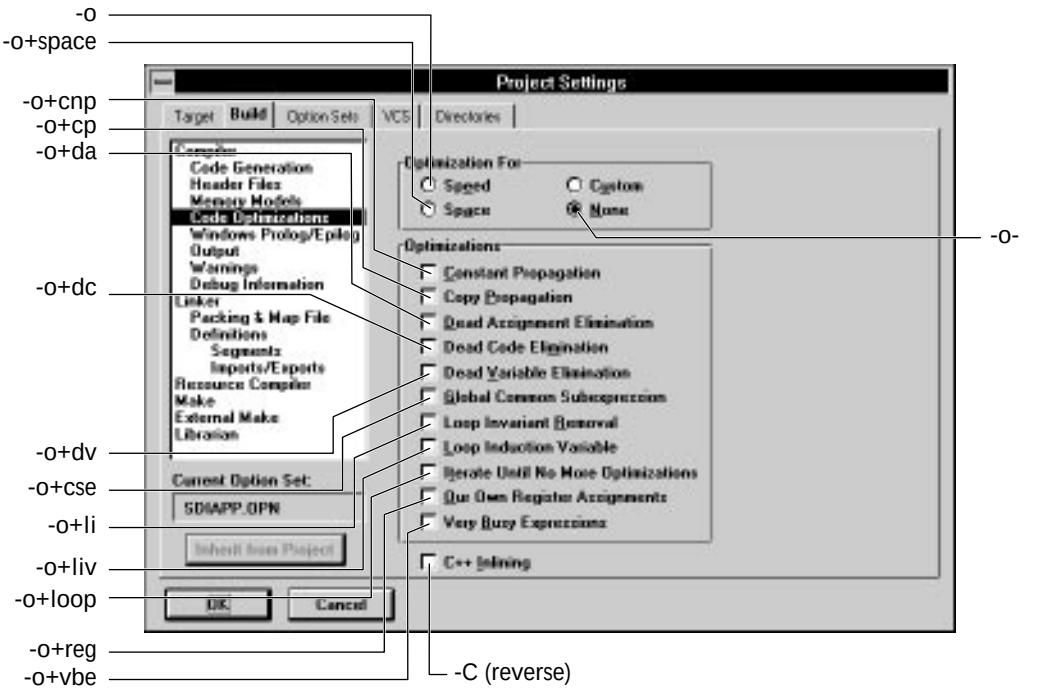
Options in the callouts to the Compiler pages are passed to the `sc.exe` compiler. Options in the callouts to the Linker subpage are passed to the `optlink.exe` linker. Definition file options (shown in the Definition subpages callouts) are placed in your project's `.def` file. Options for the resource compiler (Resource Compiler subpage) are passed to the Symantec resource compiler and linker utility, `rcc.exe`. Options for the librarian utility are passed to `lib.exe`.

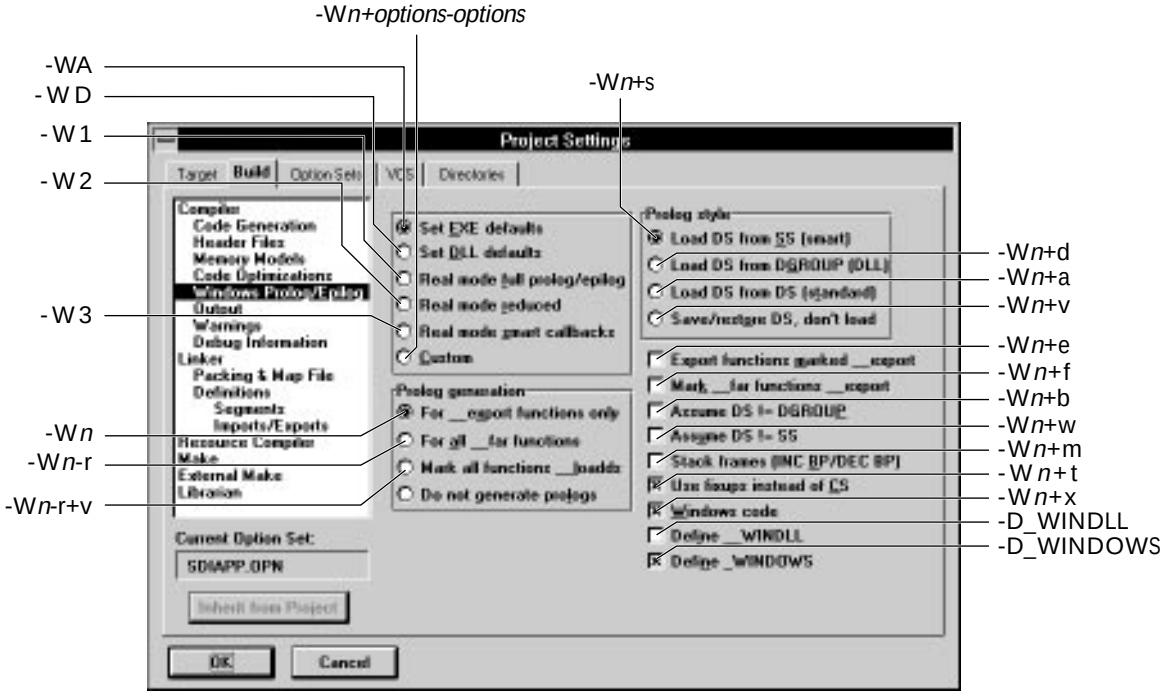

-A
-cpp
-Jm
-u
-J
-Ju
-p
-p-
-r
-j1
-j2
-EH
-ER
-j-
-D macro[=text]
-HI filename
-XI spec
-j0
Project Settings
Target
Build
Option Sets
VCS
Directories
Compiler
Code Generation
Header Files
Memory Models
Code Optimizations
Windows Prolog/Epilog
Output
Warnings
Debug Information
Linker
Packing & Map File
Definitions
Segments
Imports/Exports
Resource Compiler
Make
External Make
Librarian
Enforce ANSI Compatibility
Relax Type Checking
Suppress Predefined Macros
Exception Handling
Run time type information
char Behavior
signed
unsigned
char==unsigned char
Prototyping
Standard
Autoprototype
Strict
International Characters
None
Japanese
Taiwanese/Chinese
Korean
Current Option Set:
SDMPP.OPN
Defines
_DEBUG=1
Include Filename
Instantiate Template
Inherit from Project
OK
Cancel

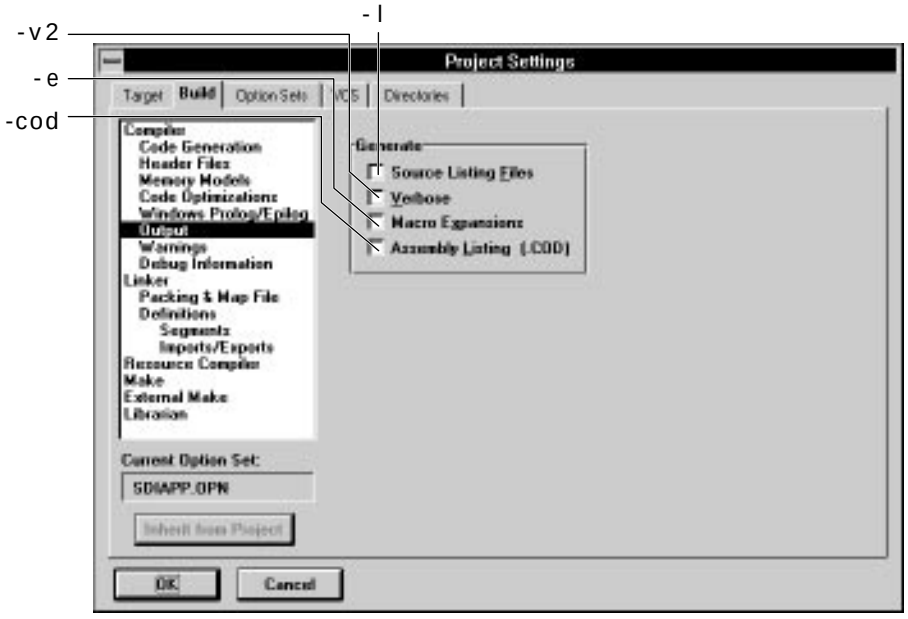

-gp
-S
-s
-ff
-f
-NV
-P
-NC
-NL
-NS
-NTname
-R
-Ns
-a1
-a2
-a4
-a8
-GTnnnn
-0
-2
-3
-4
-5
Project Settings
Target
Build
Option Sets
VCS
Directories
Compiler
Code Generation
Header Files
Memory Models
Code Optimizations
Windows Prolog/Epilog
Output
Warnings
Debug Information
Linker
Packing & Map File
Definitions
Segments
Imports/Exports
Resource Compiler
Make
External Make
Librarian
Pointer Validation
Generate Stack Frame
Check Stack Overflow
Fast Floating Point
Generate In-Line 8087 Code
Generate Virt. Function Tables in Far Data
Use Pascal Calling Convention
Enable Function-Level Link
No Default Library
Struct Alignment
Byte
Word
Double Word
Quad Word
Set Data Threshold
65535
Code Segment
Generate New Segment for Each Function
Override Default Code Segment Name
Name _TEXT
Put Switch Tables in Code Segment
Put Expression Strings in Code Segment
Target CPU
186
286
386
486
Pentium
Current Option Set:
SDIAPP.OPN
Inherit from Project
OK
Cancel

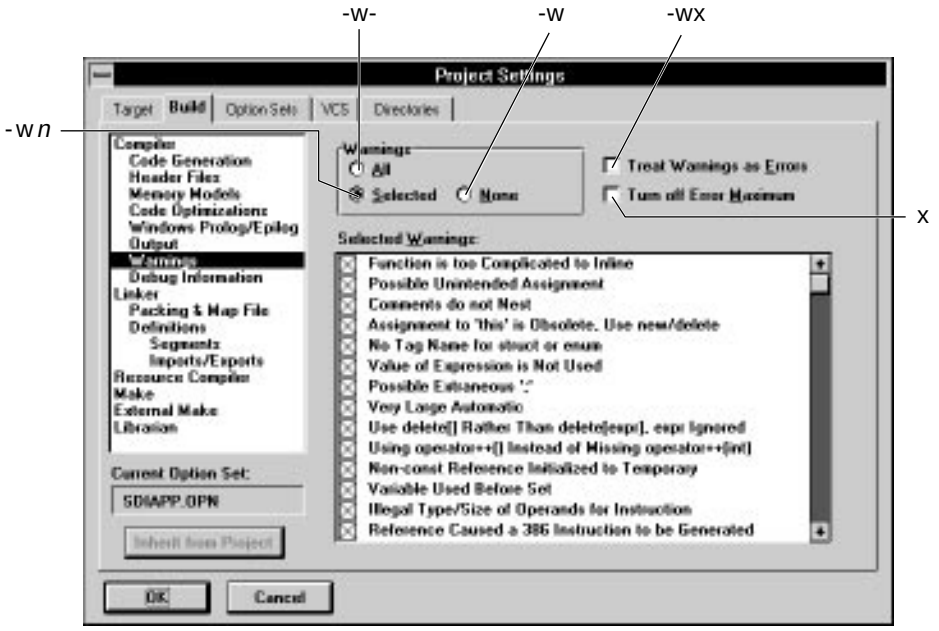

-HX
-HF*filename*
-HD*directory*
-HO
Project Settings
Target
Build
Option Sets
VCS
Directories
Compiler
Code Generation
Header Files
Memory Models
Code Optimizations
Windows Prolog/Epilog
Output
Warnings
Debug Information
Linker
Packing & Map File
Definitions
Segments
Imports/Exports
Resource Compiler
Make
External Make
Librarian
Precompile
No Headers
All Headers
Specific Header:
Use Precompiled Header From Directory:
Include Headers Once
Current Option Set:
SDIAPP.OPN
Inherit from Project
OK
Cancel

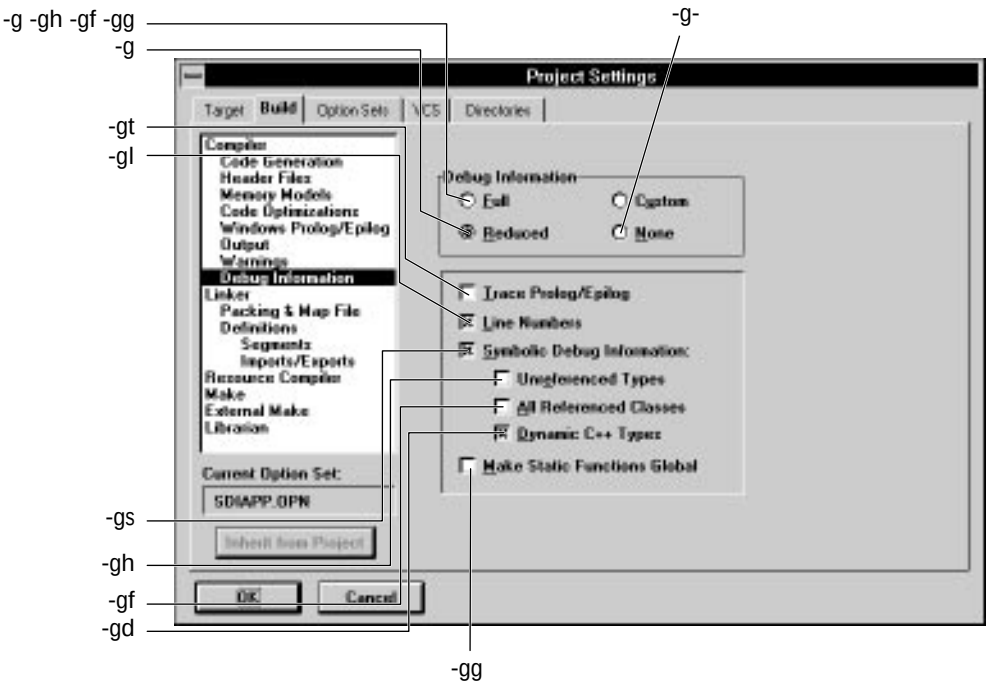

-mt
-ml
-ms
-mn
-mc
-mm
Project Settings
Target
Build
Option Sets
VCS
Directories
Compiler
Code Generation
Header Files
Memory Models
Code Optimizations
Windows Prolog/Eplog
Output
Warnings
Debug Information
Linker
Packing & Map File
Definitions
Segments
Imports/Exports
Resource Compiler
Make
External Make
Librarian
Memory Model
Tiny
Small
Medium
Compact
Large
Flat
Data Segment
Assume SS == DS
Always Reload DS
Current Option Set:
SDIAPP.OPN
Inherit from Project
OK
Cancel
-mxw (reverse)
-mxu

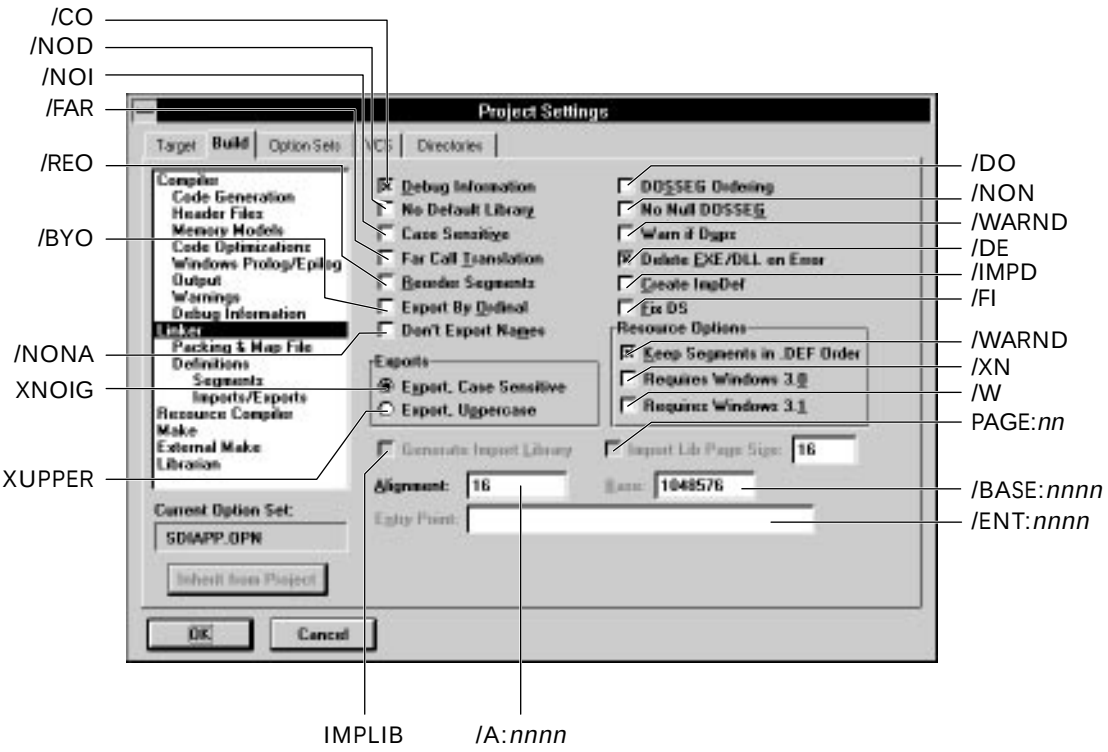

-o
-o+space
-o+cnp
-o+cp
-o+da
-o+dc
-o+dv
-o+cse
-o+li
-o+liv
-o+loop
-o+reg
-o+vbe
-o-
-C (reverse)
Project Settings
Target
Build
Option Sets
VCS
Directories
Code Generation
Header Files
Memory Models
Code Optimizations
Windows Prolog/Epilog
Output
Warnings
Debug Information
Linker
Packing & Map File
Definitions
Segments
Imports/Exports
Resource Compiler
Make
External Make
Librarian
Optimization For:
Speed
Space
Custom
None
Optimizations
Constant Propagation
Copy Propagation
Dead Assignment Elimination
Dead Code Elimination
Dead Variable Elimination
Global Common Subexpression
Loop Invariant Removal
Loop Induction Variable
Iterate Until No More Optimizations
Our Own Register Assignments
Very Busy Expressions
C++ Inlining
Current Option Set:
SDMPP.OPN
OK
Cancel

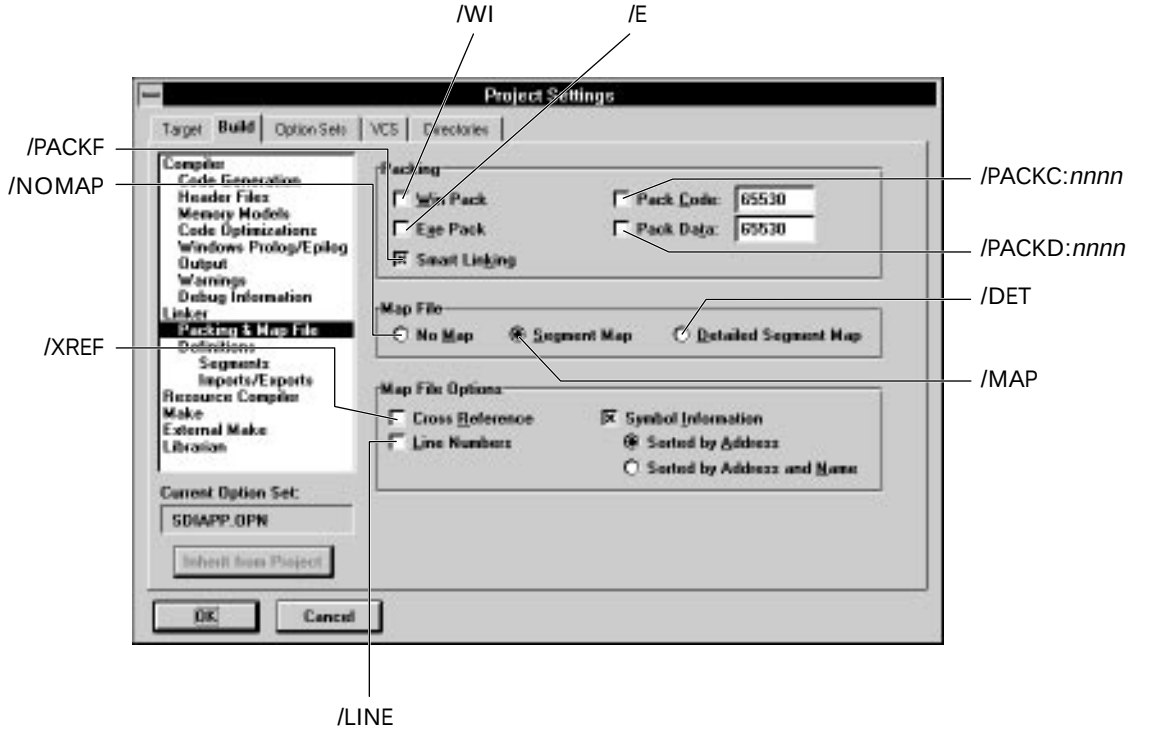

-Wn+options-options
-WA
-W D
-W 1
-W 2
-W 3
-Wn
-Wn-r
-Wn-r+v
-Wn+s
-Wn+d
-Wn+a
-Wn+v
-Wn+e
-Wn+f
-Wn+b
-Wn+w
-Wn+m
-W n+t
-Wn+x
-D_WINDLL
-D_WINDOWS
Project Settings
Target
Build
Option Sets
VCS
Directories
Compiler
Code Generation
Header Files
Memory Models
Code Optimizations
Windows Prolog/Epilog
Output
Warnings
Debug Information
Linker
Packing & Map File
Definitions
Segments
Imports/Exports
Resource Compiler
Make
External Make
Librarian
Set EXE defaults
Set DLL defaults
Real mode full prolog/epilog
Real mode reduced
Real mode smart callbacks
Custom
Prolog generation
For __export functions only
For all __far functions
Mark all functions __loadds
Do not generate prologs
Prolog style
Load DS from SS (smart)
Load DS from DGROUP (DLL)
Load DS from DS (standard)
Save/restore DS, don't load
Export functions marked __export
Mark __far functions __export
Assume DS != DGROUP
Assume DS != SS
Stack frames (INC BP/DEC BP)
Use floops instead of CS
Windows code
Define __WINDLL
Define _WINDOWS
Current Option Set:
SDIAPP.OPN
Inherit from Project
OK
Cancel

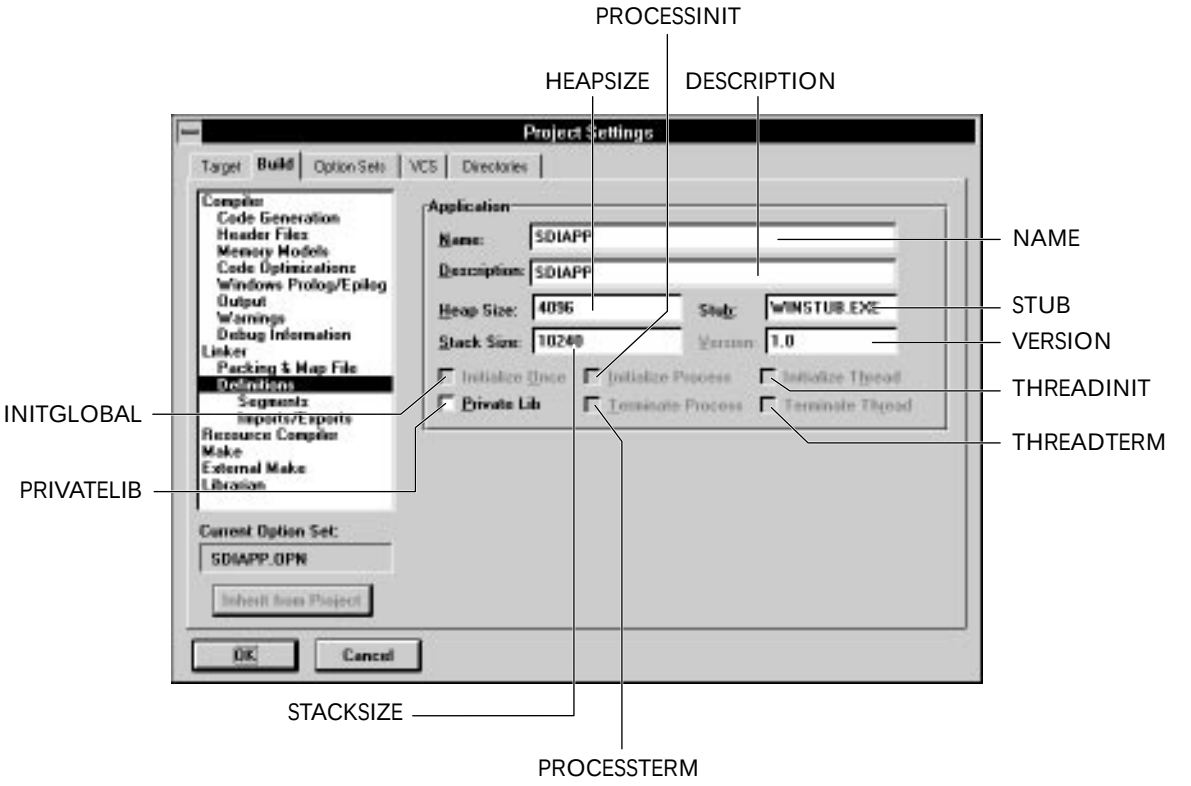

-v2
-e
-cod
-l
Project Settings
Target
Build
Option Sets
VCS
Directories
Compiler
Code Generation
Header Files
Memory Models
Code Optimizations
Windows Prolog/Epilog
Output
Warnings
Debug Information
Linker
Packing & Map File
Definitions
Segments
Imports/Exports
Resource Compiler
Make
External Make
Librarian
Generate
Source Listing Files
Verbose
Macro Expansions
Assembly Listing (.COD)
Current Option Set:
SDIAPP.OPN
Inherit from Project
OK
Cancel

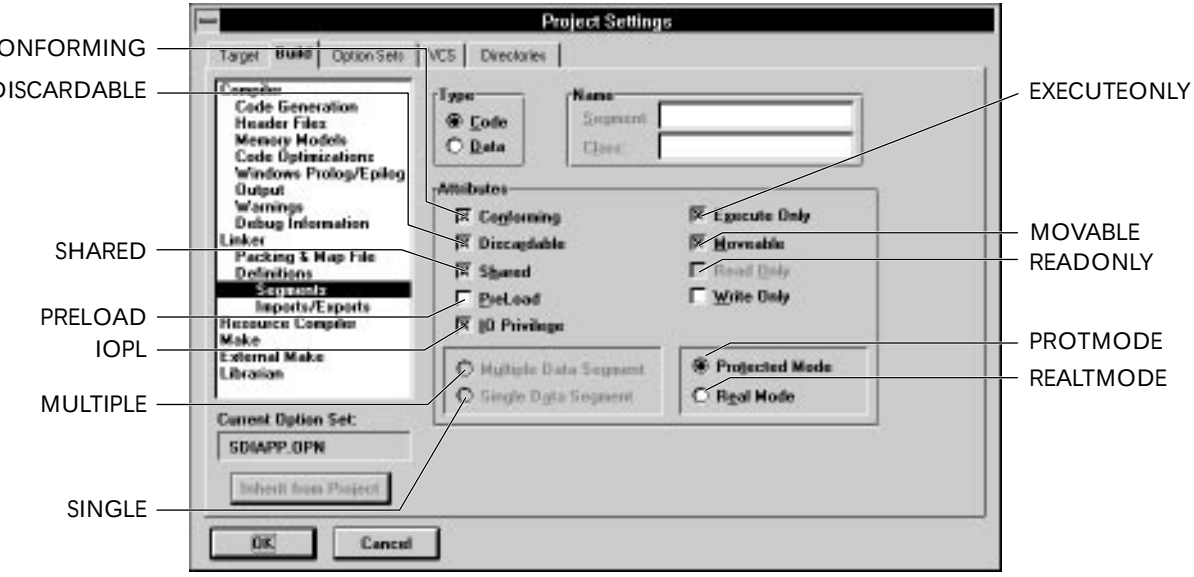

-w-
-w
-wx
-wn
x
Project Settings
Target
Build
Option Sets
VCS
Directories
Compiler
Code Generation
Header Files
Memory Models
Code Optimizations
Windows Prolog/Epilog
Output
Warnings
Debug Information
Linker
Packing & Map File
Definitions
Segments
Imports/Exports
Resource Compiler
Make
External Make
Librarian
Warnings:
All
Selected
None
Treat Warnings as Errors
Turn off Error Maximum
Selected Warnings:
Function is too Complicated to Inline
Possible Unintended Assignment
Comments do not Nest
Assignment to 'this' is Obsolete. Use new/delete
No Tag Name for struct or enum
Value of Expression is Not Used
Possible Extraneous ';'
Very Large Automatic
Use delete[] Rather Than delete[exp]. exp Ignored
Using operator++() Instead of Missing operator++(int)
Non-const Reference Initialized to Temporary
Variable Used Before Set
Illegal Type/Size of Operands for Instruction
Reference Caused a 386 Instruction to be Generated
Current Option Set:
SDIAPP.OPN
Inherit from Project
OK
Cancel

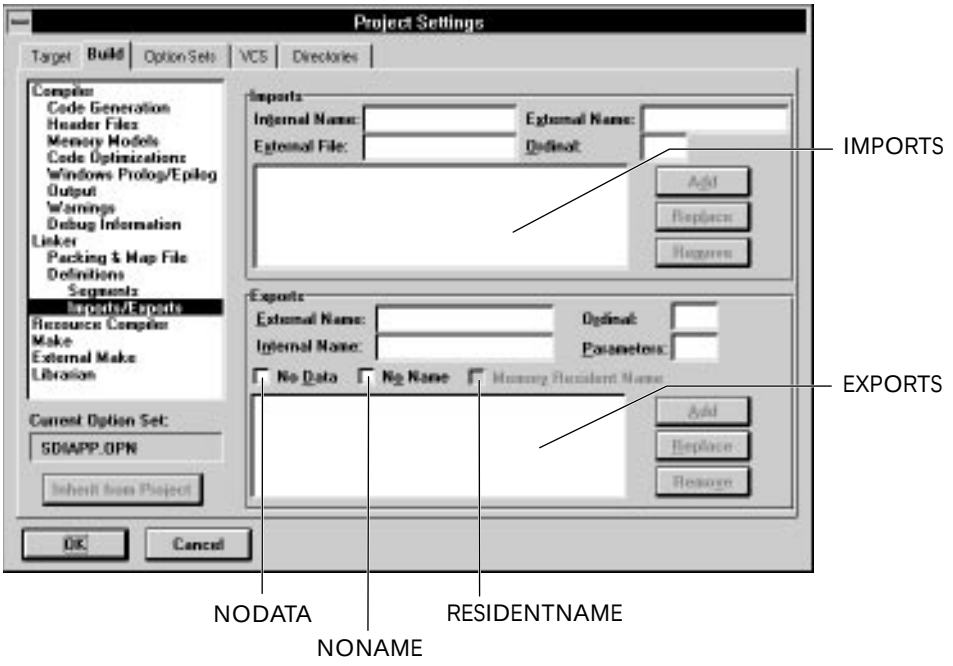

-g -gh -gf -gg
-g
-g-
-gt
-gl
-gs
-gh
-gf
-gd
-gg
Project Settings
Target
Build
Option Sets
VCS
Directories
Compiler
Code Generation
Header Files
Memory Models
Code Optimizations
Windows Prolog/Epilog
Output
Warnings
Debug Information
Linker
Packing & Map File
Definitions
Segments
Imports/Exports
Resource Compiler
Make
External Make
Librarian
Debug Information:
Full
Custom
Reduced
None
Trace Prolog/Epilog
Line Numbers
Symbolic Debug Information
Unreferenced Types
All Referenced Classes
Dynamic C++ Types
Make Static Functions Global
Current Option Set:
SDIAPP.OPN
Inherit from Project
OK
Cancel

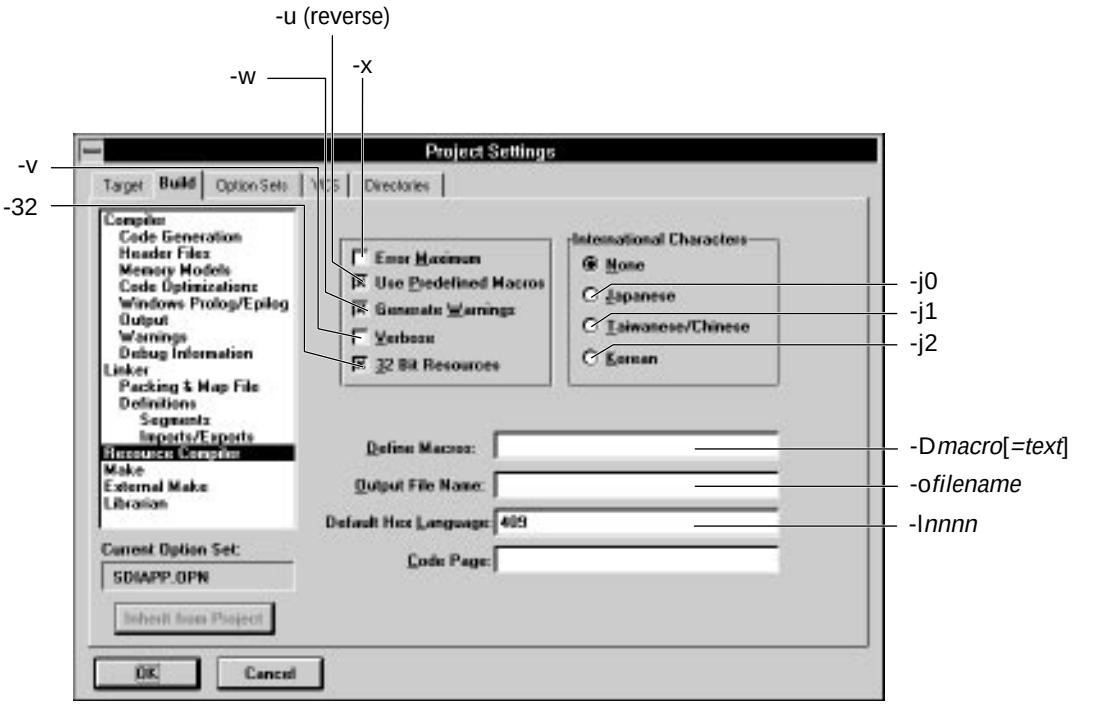

/CO
/NOD
/NOI
/FAR
/REO
/BYO
/NONA
XNOIG
XUPPER
IMPLIB
/A:nnnn
/DO
/NON
/WARND
/DE
/IMPD
/FI
/WARND
/XN
/W
PAGE:nn
/BASE:nnnn
/ENT:nnnn
Project Settings
Target
Build
Option Sets
Directories
Compiler
Code Generation
Header Files
Memory Models
Code Optimizations
Windows Prolog/Epilog
Output
Warnings
Debug Information
Linker
Packing & Map File
Definitions
Segments
Imports/Exports
Resource Compiler
Make
External Make
Librarian
Debug Information
No Default Library
Case Sensitive
Far Call Translation
Reorder Segments
Export By Ordinal
Don't Export Names
Exports
Export, Case Sensitive
Export, Uppercase
DOSSEG Ordering
No Null DOSSEG
Warn if Dupe
Delete EXE/DLL on Error
Create ImpDef
Fix DS
Resource Options
Keep Segments in .DEF Order
Requires Windows 3.0
Requires Windows 3.1
Generate Import Library
Import Lib Page Size:
16
Alignment:
16
Base:
1048576
Entry Point:
Current Option Set:
SDIAPP.OPN
Inherit from Project
OK
Cancel

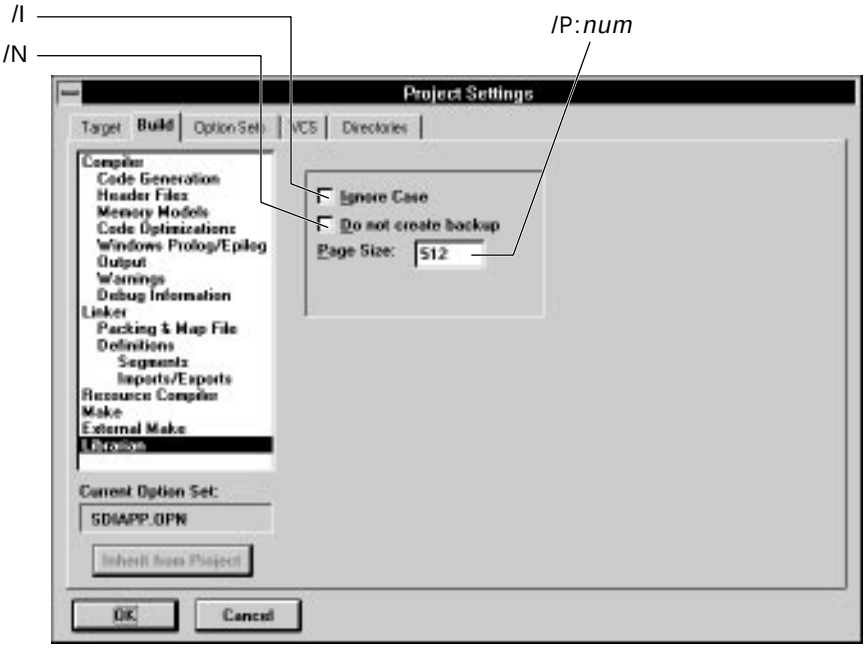

/WI
/E
/PACKF
/NOMAP
/PACKC:nnnn
/PACKD:nnnn
/DET
/XREF
/MAP
/LINE
Project Settings
Target
Build
Option Sets
VCS
Directories
Compiler
Code Generation
Header Files
Memory Models
Code Optimizations
Windows Prolog/Epilog
Output
Warnings
Debug Information
Linker
Packing & Map File
Definitions
Segments
Imports/Exports
Resource Compiler
Make
External Make
Librarian
Packing
Win Pack
Exe Pack
Smart Linking
Pack Code:
65530
Pack Data:
65530
Map File
No Map
Segment Map
Detailed Segment Map
Map File Options
Cross Reference
Line Numbers
Symbol Information
Sorted by Address
Sorted by Address and Name
Current Option Set:
SDIAPP.OPN
Inherit from Project
OK
Cancel


PROCESSINIT
HEAPSIZE
DESCRIPTION
Project Settings
Target
Build
Option Sets
VCS
Directories
Compiler
Code Generation
Header Files
Memory Models
Code Optimizations
Windows Prolog/Epilog
Output
Warnings
Debug Information
Linker
Packing & Map File
Definitions
Segments
Imports/Exports
Resource Compiler
Make
External Make
Librarian
Application
Name:
SDIAPP
Description:
SDIAPP
Heap Size:
4096
Stub:
WINSTUB.EXE
Stack Size:
10240
Version:
1.0
Initialize Once
Initialize Process
Initialize Thread
Private Lib
Terminate Process
Terminate Thread
Current Option Set:
SDIAPP.OPN
Inherit from Project
OK
Cancel
NAME
STUB
VERSION
THREADINIT
THREADTERM
INITGLOBAL
PRIVATELIB
STACKSIZE
PROCESSTERM


ONFORMING
DISCARDABLE
SHARED
PRELOAD
IOPL
MULTIPLE
SINGLE
EXECUTEONLY
MOVABLE
READONLY
PROTMODE
REALTMODE
Project Settings
Target
Build
Option Sets
VCS
Directories
Compiler
Code Generation
Header Files
Memory Models
Code Optimizations
Windows Prolog/Epilog
Output
Warnings
Debug Information
Linker
Packing & Map File
Definitions
Segments
Imports/Exports
Resource Compiler
Make
External Make
Librarian
Type
Code
Data
Name
Segment
Class
Attributes
Conforming
Discardable
Shared
PreLoad
IO Privilege
Execute Only
Movable
Read Only
Write Only
Multiple Data Segment
Single Data Segment
Protected Mode
Real Mode
Current Option Set:
SDIAPP.OPN
Inherit from Project
OK
Cancel


Project Settings
Target
Build
Option Sets
VCS
Directories
Compiler
Code Generation
Header Files
Memory Models
Code Optimizations
Windows Prolog/Epilog
Output
Warnings
Debug Information
Linker
Packing & Map File
Definitions
Segments
Imports/Exports
Resource Compiler
Make
External Make
Librarian
Imports
Internal Name:
External Name:
External File:
Ordinal:
Add
Replace
Remove
Exports
External Name:
Ordinal:
Internal Name:
Parameters:
No Data
No Name
Missing Resident Name
Add
Replace
Remove
Current Option Set:
SDIAPP.OPN
Inherit from Project
OK
Cancel
IMPORTS
EXPORTS
NODATA
NONAME
RESIDENTNAME


-u (reverse)
-w
-x
-v
-32
Project Settings
Target
Build
Option Sets
VMS
Directories
Compiler
Code Generation
Header Files
Memory Models
Code Optimizations
Windows Prolog/Epilog
Output
Warnings
Debug Information
Linker
Packing & Map File
Definitions
Segments
Imports/Exports
Resource Compiler
Make
External Make
Librarian
Error Maximum
Use Predefined Macros
Generate Warnings
Verbose
32 Bit Resources
International Characters
None
Japanese
Taiwanese/Chinese
Korean
-j0
-j1
-j2
Define Macros:
Output File Name:
Default Hex Language:
409
Code Page:
-Dmacro[=text]
-ofilename
-Innnn
Current Option Set:
SDIAPP.OPN
Inherit from Project
OK
Cancel


/I
/N
/P:num
Project Settings
Target
Build
Option Sets
VCS
Directories
Compiler
Code Generation
Header Files
Memory Models
Code Optimizations
Windows Prolog/Epilog
Output
Warnings
Debug Information
Linker
Packing & Map File
Definitions
Segments
Imports/Exports
Resource Compiler
Make
External Make
Browser
Ignore Case
Do not create backup
Page Size:
512
Current Option Set:
SDIAPP.OPN
Inherit from Project
OK
Cancel

Using NetBuild

NetBuild is a feature of Symantec C++ that lets you distribute the task of building a project across the network. Using NetBuild, you can employ any idle PCs on the network to compile your files.

Before you can use NetBuild you must install the NetBuild Administrator software on your network server and the NetBuild Server software on any PCs that will participate in the distributed build. For information on how to install the NetBuild feature of Symantec C++, see the *Getting Started Guide*.

The following sections detail how to use the Symantec C++ NetBuild feature.

Using the Build Client

Your PC (the one controlling the build) is called the build client. Functioning as a client, it must request a server to perform a task. The PCs that compile your project are called build servers.

Configuring the build client

The options for controlling the NetBuild on the build client are located in the **Project Settings** dialog box. To access these options, select **Settings** from the IDDE's **Project** menu, then click on the Build tab to bring up the Build page of the **Project Options** dialog box. In the list of subpages on the left, choose Make.

The Make subpage is shown in Figure C-1.

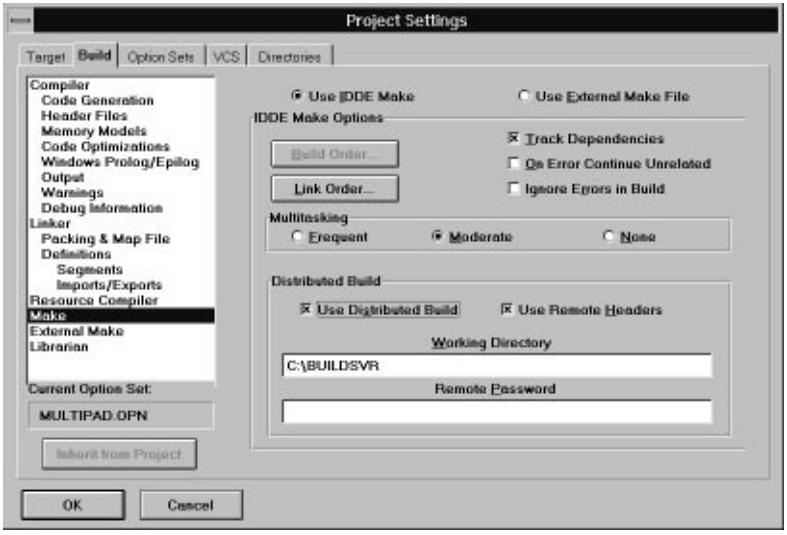


Figure C-1 The Make subpage

To turn on the NetBuild feature, make sure that the Use NetBuild check box has a check mark.

If you enable the Use Remote Headers option, the client PC instructs the build servers to use the header files on their local drives (as opposed to getting the header files from the build client). This option can be overridden on the build server side by disabling the Use Local Headers option in the Build Server Configuration window. See “Configuring a build server,” later in this chapter.

When you enable the Use NetBuild option, you must specify the pathname of the build server control directory in the Working Directory textbox. This is the directory in which the NetBuild Administrator is installed on the network server. You specify this pathname in the Working Directory textbox. All the PCs participating in the distributed build must have access to this directory.

If your network administrator has set up a network password, you must enter it in the Remote Password textbox. If sources are shared through Microsoft Network on a local hard disk, and a sharing password is used for access protection, it must be specified in the Remote Password textbox.

Starting a distributed build

Once you have configured the build client, every time you rebuild or update your project, the NetBuild system identifies the build server(s) that can participate in the distributed build. If there is at least one build server available, the distributed build can occur; otherwise the build client builds the project locally. For information on starting and configuring build servers, see the following section.

When the distributed build process begins, the Build Client window opens on the client PC's screen (see Figure C-2). This window allows you to monitor the progress of the distributed build.

When the build is done, a message specifying successful completion or an error condition is displayed in the output window on the build client.

Monitoring a distributed build

You can monitor a distributed build from the build client PC by viewing the messages displayed in the Build Client window.

Figure C-2 The Build Client window

This window displays an icon each for the build client and each of the build servers participating in the distributed build. Each icon shows a list of the programs and header files being handled by that particular PC. The name of the build server is shown at the top of its icon.

Stopping a distributed build

You can cancel a build from the build client PC by:

- Choosing the Stop! command in the output window
- Choosing **Stop Build** from the **Project** menu
- From the client PC, clicking the Cancel button in the build client window

Depending on the options that have been specified for the project being compiled, there may be a short delay before the distributed build stops.

Using a Build Server

A build server is a PC that is helping the build client PC compile a project. A build server normally is idle until a client requests it to perform a task. There must be at least one build server in the distributed build environment.

Starting a build server

You start the NetBuild Server application by double-clicking on its icon, or as you start any other Windows program. Once started, the server application is minimized into an icon. It then runs in the background, waiting to receive a task from a build client.

Configuring a build server

When the build server compiles files, the build server application name changes to the name of the file it is compiling.

Double-click on the NetBuild Server application icon to display a Build Server Configuration window (see Figure C-3).

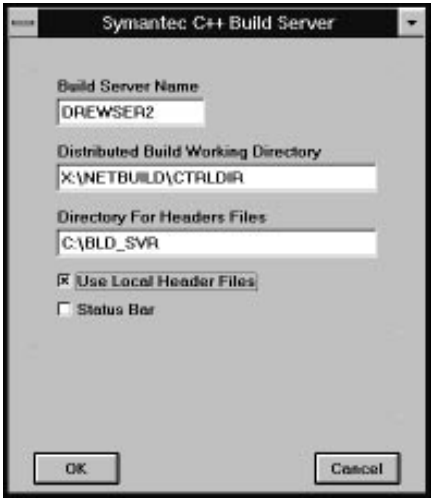


Figure C-3 The Build Server Configuration window

This window displays the build server name, the pathname of the working directory, and the pathname of the directory containing the header files used in the compilation. (If the pathname of the working directory is changed, then the pathname must be identical in the client working directory.) This window also contains a message area; this is where messages, such as the name of the file being compiled, the file being read, and error messages, are displayed.

If Use Local Header Files option is enabled, the build server looks for the header files in the “Directory for Header Files” specified in the build server window.If the Use Local Header Files option is not enabled, the build server gets the header files from the build client PC. This option is enabled if the Use Remote Headers option is enabled on the client PC; however, you can still force the build client to retrieve the headers from the build client by disabling the Use Local Header Files option in the Build Server Configuration window.

Note

If Use Local Header Files is selected and the build server needs a header file and cannot find it, the build server sends the job back to the build client for processing.

If the Status Bar option is selected, the status bar is displayed at the bottom of the Build Server Configuration window.

Stopping a build server

You can close the NetBuild Server application by:

- Pressing Alt+F4
- Double-clicking on the system menu button
- Choosing Close on the system menu

If the server is participating in a distributed build when you try to stop it, the dialog box shown in Figure C-4 is displayed.

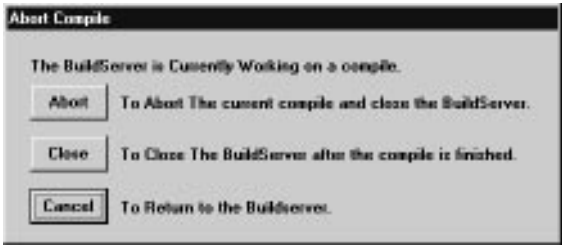


Figure C-4 Abort Compile dialog box

If you click on Cancel in the dialog box, the build server resumes processing. If you click on Close, the build server finishes processing any files that are in the queue (that is, files currently being compiled and files already read), then stops. If you click on Abort, the build server halts the current compilation and returns the uncompleted job to the build client. If you selected Close or Abort, the NetBuild Server application closes and you must restart it if you want the PC to be able to participate in a distributed build again. (See “Starting a build server,” earlier in this chapter.)

Troubleshooting

If the build client or build server seems to hang (because no messages are being displayed), wait for a few minutes. NetBuild tries to establish a connection several times before giving up.

If both the build client and build server start, but the build server does not compile your program, then:

1. Make sure you selected the NetBuild option on the Make Subpage.
2. Make sure the NetBuild Working Directory has the correct pathname for the build server control directory.
3. Make sure the build client and build server have the same drive mapping.
4. Make sure the build servers have access tot he include directories. The include files can be installed on the build server, or in a fully-sharable directory on the build client.

If the build client does not receive messages from the build server(s):

1. Make sure the network is not down.
2. Make sure the build server is still running. If the build server is still displaying messages, the network connection may be broken. If not, Windows may be hung. If the build server is in iconized form, it may have finished the compile, but was not able to communicate with the build client.

If files do not compile or the build server can't find header files:

1. Make sure all relevant files have been moved to the network drive.
2. Make sure the environment variables that indicate where to find the header files are set properly.

NetBuild Messages

This section describes the NetBuild error and information messages.

Build client messages

Build being done on *build_server*
This informational message is repeated for each command line. *build_server* is the name of the build server compiling the module.

Compile has been aborted, Rescheduling to another server
The build server was shut down abruptly. The job has been sent to another server to be compiled. This message is displayed when someone cancels the build and selects the abort option.

NetBuild has lost contact with *build_server*
The netbuild has lost contact with the specified server. The build server PC is hung, the network is down, or the netbios protocol is not set correctly.

There is not enough memory to do distributed builds
The system does not have enough memory to run.

Unable to access *build_server*. *client_server* will compile this build locally
The build server was not able to access the specified file. The more likely problems are that the drive on the client side is not shared or the file is not in the shared directory.

Unable to Load network connection Dlls (WNET32, WNET16)
The system was not able to load the following DLLs: `wnet32.dll`, `wnet16.dll`. They may be missing or corrupted.

Unable to Load network share Dlls (WSHR32, WSHR16)
The system was not able to load the following DLLs: `wshr32.dll`, `wshr16.dll`. They may be missing or corrupted.

Build server messages

Another Station has the control file locked
The control file has been locked by another station. It is probable that station is also hung.

Error checking for Control file, Error *xx*
An unexpected error has occurred. Record the error number (*xx*) and contact technical support, providing as much information as possible.

Error checking for Control Path Error *xx*
An unexpected error has occurred. Record the number (*xx*) and contact technical support, providing as much information as possible.

Error in packet
The build server and build client software are not the same version.

Problems reading ini file.
There is a problem reading the `.ini` file. This message should not occur if the `.ini` file is missing.

Problem with initializing build server name
There is a problem reading the `.ini` file. This message should not occur if the `.ini` file is missing.

Problem with initializing control file
There is a problem reading the `.ini` file. This message should not occur if the `.ini` file is missing.

Problem with initializing header path
There is a problem reading the `.ini` file. This message should not occur if the `.ini` file is missing.

The control file path is invalid
The specified build server control file pathname specified in the **NetBuild Working Directory** dialog box is not valid.

Network errors

Network Dlls Failed to initialize Correctly, Cannot initialize Netbios on lana *xx*
Unable to initialize netbios. It may not be installed correctly. The lana *xx* refers to the netbios protocol that has failed (there can be several on a machine).

Network Dlls Failed to initialize Correctly, Listens Failed To Establish
System was able to initialize netbios but unable to set up communication.

Network Dlls Failed to initialize Correctly, Netbios command *xx* error code *yy* lana *zz*
A call to netbios has failed. This message lists the command, the error code, and the lana number (all in hexadecimal).

Network Dlls Failed to initialize Correctly, Netbios is not installed.
This message refers to the netbios name table. The system probably has crashed. The best solution is to restart your computer.

Network Dlls Failed to initialize Correctly, Netbios name already exists.
This message refers to the netbios name table. The system probably has crashed. The best solution is to restart your computer.

Network Dlls Failed to initialize Correctly, Netbios name table full.
This message refers to the netbios name table. The system probably has crashed. The best solution is to restart your computer.

Network Dlls Failed to initialize Correctly, No Error Information Available
Unknown error. No error information is available. This is a catch-all error that the system uses.

Network Dlls Failed to initialize Correctly, Out Of Memory
System is low on memory.

Network Dlls Failed to initialize Correctly, Reset Timed out on lana *xx*
Unable to reset the netbios settings. It probably is not installed correctly. The lana *xx* refers to the netbios protocol that has failed. (There can be several on a machine.)

Network Dlls Failed to initialize Correctly, Unable To Get Network Address
Unable to get the address of the network card. The card or netbios probably is not installed correctly.

Network Dlls Failed to initialize Correctly, Unknown Error
Unknown error. No error information is available. This is a catch-all error that the system uses.

Unable to Load network Dll because could not find entry point in network DLL
The DLL, which was able to load, is not a valid DLL.

Unable to Load network Dll because could not load NB32.DLL
The system could not load the `nb32.dll` file. This is a Windows NT or Windows 95 message.

Unable to Load network Dll because could not load NBND.DLL
The system could not load the `nbnd.dll` or `nbd.dll` files. This is a Windows NT or Windows 95 message.