BugSlay ™
User's manual

Special Demo Version

By Rex K. Perkins, July 1994

Table of Contents

Introduction

It has happened to us all. Your application suddenly comes to a grinding halt with a protection fault. All you have is an obscure address, possibly not even in your code. Worse still, a customer found the fault and did not log the address.

So you start the hunt. If the fault occurred in your code, usually you can find it using the compiler. Often however, the address is in a core library that could have been called from a dozen places in the code. This address is of little use if you do not know how execution got there.

If the address is in another module, and you do not have the source code, it can be difficult or impossible to find the fault. Further you may not know how your program came to be in that module.

Now BugSlay can help solve these problems. BugSlay traps exceptions, and run-time errors, generated by your application and produces a detailed stack trace showing the events leading to the error. Armed with this information you can easily see exactly where and why the program died.

BugSlay differs in several respects from conventional postmortem debugging aids, such as Dr. Watson from Microsoft. Specifically:

i) BugSlay can trap run-time errors. Dr. Watson does not know run-time errors exist.
ii) BugSlay was designed for Turbo Debugger symbols. These symbols allow far more detailed symbolic dumps than SYM files, including local and global variables.
iii) Designed for Pascal, BugSlay's output is in a format natural to a Pascal programmer. It uses Pascal conventions wherever possible.

Features

- Designed for Borland Pascal and Turbo Pascal For Windows.
- Supports both Borlands' Turbo Debugger symbols (in the EXE) and Microsoft's SYM symbol files.
- Traps both exceptions (Protection Faults etc.) and run-time errors.
- Produces a symbolic stack trace with local variables and parameters.
- Supports multiple modules, each with it's own symbols. Symbols are linked automatically. No pre-loading of symbols is required.
- Optionally check for allocations on the Pascal heap upon normal application termination. Useful for locating potential memory leaks.
- Optionally sends an overview of the trace to a debugging terminal

- Configurable. The level of trace detail can be selected.

Requirements

- Windows 3.1
- Borland Pascal 7.0 or 7.01, or Turbo Pascal For Windows 1.5

Limitations

BugSlay does not:

i) Support runtime error 202, Stack overflow error; or
ii) Expressly support Objects; or
iii) Support DLLs not called from a Pascal application; or
iv) Correctly interpret Turbo Debugger symbols generated by compilers other than Borland Pascal or Turbo Pascal for Windows.

About the Special Demo version of BugSlay

This unregistered, special demo version of BugSlay provides the basic functions for you to evaluate BugSlay's functionality, convenience and compatibility. You are licensed to use it for 30 days to evaluate BugSlay's suitability to your needs. After this period you are required to either register BugSlay or cease using it.

The unregistered, special demo version of BugSlay may not be redistributed with your application.

By registering BugSlay you will be licensed to use the full version of BugSlay and you also will receive:

- Full record and pointer support. Unroll records and deference pointers in variable dumps. See the sample error log for example.
- Log files may contain more than one error 'event' allowing users to log several errors.
- No Unregistered 'nag' screens appearing on errors.
- Notice of any future releases.

To obtain the registered version of BugSlay, see Registering below.

File List

The Special Demo version of BugSlay is distributed in a ZIP archive containing the following files:

BugSlay.DLL	The core of BugSlay. Must be in the search path during application execution for BugSlay to be active.
BugSlay.PAS	Include this unit in your application. Should be the first unit in the main programs' Uses clause.
BugSlayI.PAS	Imports unit for BugSlay.DLL. Used by BugSlay.PAS.
Execptio.PAS	Exception handler. Used by BugSlay.PAS.
CrashMe.PAS	Sample application that causes a run-time error, invoking BugSlay. This application was used to generate the sample error.log.
BugSlay.WRI	This document, in Windows Write format.
File_ID.DIZ	Archive description.

BugSlay's anatomy

BugSlay consists of a DLL and several small Pascal units. The main unit is linked into your application to trap the exit procedures and provide the interface to the DLL.

To use BugSlay, all you need to do is put the BugSlay unit at the start of the application's Uses clause, and BugSlay will do the rest.

The included import unit will *explicitly* link to BugSlay.DLL at run time rather than the more conventional implicit links. This means that if BugSlay.DLL is not found, the application can still run normally without any error messages.

Installing BugSlay

To use BugSlay, copy the *.PAS files to either your work directory, your Units directory or another directory in the IDE's or compiler's unit search path. The DLL file can be in anywhere in the PATH or in the directory your EXEs are generated in. For example:

Put the *.PAS files in c:\bp\units and BugSlay.DLL in c:\bp\bin.

Add BugSlay to the start of the uses clause of the main program:

```
    Uses BugSlay, WinTypes, WinProcs.....
```

Using BugSlay

When BugSlay detects an error it alerts the user to the error and asks them if they want to log the error. If they say no, the error is passed along as normal; otherwise an error log is generated.

BugSlay writes the details to a log file, with the default name of c:\error.log. If BugSlay is unable to open this file it will attempt to open a file called c:\error.alt. If this attempt also fails, no log will be generated.

You may wish to use BugSlay in conjunction with a conventional postmortem debugger such as Dr. Watson. Doing so will gather yet more information about the state of the system as a whole and give the user an opportunity to annotate the crash. Logs can be synchronized by the time stamp.

Symbols

In order for BugSlay to produce a symbolic dump, it needs symbols. You can provide these in two forms:

i) Turbo Debugger info in the EXE (or DLL) file. This is the preferred method of supplying the symbols as the Turbo Debugger information contains very detailed symbolic information. Also, there is no risk of the symbol information getting out of sync with the executable.

   To generate Turbo Debugger symbol information you need to select 'Debugging Information' {$D+}, 'Local Symbols' {$L+} and 'Symbol Information' from the Compiler options dialog as well as 'Debug Info in EXE' in the Linker options dialog.

   Users of Borland Pascal will want to recompile the RTL with symbol information enabled to fully exploit BugSlay's functionality. See the "Creating a debug version of the run-time library" section of the ReadMe file in the RTL base directory.

   Only modules with Turbo Debugger information in them will show the local and global variables.

   Normally, you will want to disable the 'Debug Info in EXE' option for shipping code as the debugging information can add a significant overhead to the size of the executable.

ii) SYM files. If Turbo Debugger information for a module is not found, BugSlay attempts to open a SYM file of the same base name in the same location as the module.

Some authors supply SYM files with their DLLs. Microsoft supplies SYM files for the core Windows components with it's SDK and on the Microsoft Development Platform CDs.

If you do not have a SYM file for a module you are using, you can build one based on the publics in that module by using Borlands' BuildSym. Note that Borland omitted BuildSym.EXE from the 7.0 release of Borland Pascal, but it was included in the 7.01 maintenance release. At the time of writing BuildSym.EXE was also available free of charge on the BPascal forum of CompuServe.

In addition to any DLLs you explicitly use, you will want to generate SYM files for: Display drivers, Printer drivers, User, KRNL?86 and GDI. Note that the system DLLs will not always have a DLL extension, it could be anything including EXE and DRV.

Example: To generate a SYM file for VGA.DRV, type at a DOS prompt from the Windows System directory:

```
buildsym vga.drv
```

Sample Error.Log

The following is an example of a typical log, with commentary. Some detail has been removed and replaced with "[............]" to improve readability. This sample log was generated by CrashMe.PAS, compiled using BPW 7.01. Addresses and symbol capitalization in TPW 1.5 differs slightly.

The following example shows the maximum level of detail generated with the registered version of BugSlay. The unregistered, special demo version will not generate all of the details shown here.

```
Logged at 12:54:22.79 on 12JUL94
```

```
D:\BP\WINWORK\CRASHME.EXE had a Run Time Error, code 201 (Range
check error) at 0001:00F5
```

Shows what the error was (Run Time Error or Unexpected Exception), the task that caused it and the logical address of the instruction that caused the fault. If the fault was an exception, the module containing the logical address is also shown.

```
0001:00F5 is in module CrashMe on line 100 of file d:\bp\
winwork\crashme.pas  11:13:42 12JUL94
```

Identifies the source file containing the fault address. This line will only be generated if the address can be found, that is if there is Turbo Debugger information for that address.

This line shows the module name (Program or Unit name in this context), the line of the error, the source file name, the time and the date of the named file when the binary was built. Check this time and date as the file may have been changed since rendering the line number questionable.

```
Stack trace -- Brief
--------------------

429E 681F:00F5   CRASHME!(0001)  CrashMe.DrawBox
42AA 681F:01DF n CRASHME!(0001)  CrashMe.DrawLine
42BC 681F:0224 n CRASHME!(0001)  CrashMe.DrawBoxes
42C6 681F:0259 n CRASHME!(0001)  CrashMe.Paint
42F6 6837:175C F CRASHME!(0002)  OWindows.WMPaint
4318 6837:016F F CRASHME!(0002)  OWindows.StdWndProc
4332 04C7:2B34 F USER!(0001)  IGROUP.GLOBALGETATOMNAME+$551
4350 6837:1AC3 F CRASHME!(0002)  OWindows.MessageLoop
435A 6837:1A44 F CRASHME!(0002)  OWindows.Run
4364 681F:02AB F CRASHME!(0001)  CrashMe.
4366 681F:0000 n CRASHME!(0001)
```

An overview of the stack. The first column is the address of the stack frame (i.e., the BP register) in the stack segment. Next is the return address for the frame, that is the address of the instruction after the one that called the next frame. (NOTE: Stacks have the first entry at the bottom, much like the Pascal hierarchy).

The 'F' or 'n' indicates whether the return address is a near or far, not normally of interest to Pascal programmers, but can help find problems when this is important.

The next column is the module name (as returned by Windows), followed by the logical segment number for this address.

If the symbols for the module are available, BugSlay attempts to find the name of the segment and the function/procedure the address lies in. The output is in the format Module.Function by default.

Due to the structure of SYM files, the name of a function can not be determined exactly for those modules with SYM files only. In these cases, the nearest symbol before the address is shown, followed by the offset from this function to the given address. While the symbol is often accurate, in some cases where the address lies in a non-exported area of the module the symbol may be incorrect. Check the offset and use your judgment (or a dissassembler!)

```
Stack trace -- Detailed
-----------------------

429E 681F:00F5   CRASHME!(0001)  CrashMe.DrawBox    on line 100
of file d:\bp\winwork\crashme.pas  11:13:42 12JUL94
  LocPar Col:Integer=0 ($0000)
  LocPar Line:Integer=0 ($0000)
  LocVar BoxToDraw:TRect=
            left:Integer=17 ($0011)
            top:Integer=1 ($0001)
            right:Integer=220 ($00DC)
            bottom:Integer=26655 ($681F)
  LocVar ColoredBrush:Word=4962 ($1362)
```

The first frame of the stack trace is where the error occurred. Here we see the start of the detailed stack trace and the scene of the crime. In addition to the details shown in the overview, we now also have file and line number for the address and the function's ('DrawBox') local variables and parameters. A detailed discussion of the variables and parameter list is in a following section, but here we can see the cause of the problem.

Line 100 of crashme.pas is:

```
      ColoredBrush:=CreateSolidBrush(ColorTable^[Col+Line]);
```

Looking up the declaration of ColorTable we see that it is:

```
     Array[1..NumColors] Of TColorRef;
```

Since Col is 0 and Line is 0, Col+Line is also 0. Since the array has a base index of 1, and range checking is on, the attempt to access index 0 caused the run-time error.

```
42AA 681F:01DF n CRASHME!(0001)  CrashMe.DrawLine    on line 120
of file d:\bp\winwork\crashme.pas  11:13:42 12JUL94
  LocPar Line:Integer=0 ($0000)
  LocVar Col:Integer=0 ($0000)
42BC 681F:0224 n CRASHME!(0001)  CrashMe.DrawBoxes    on line
130 of file d:\bp\winwork\crashme.pas  11:13:42 12JUL94
  LocVar WindowSize:TRect=
            left:Integer=0 ($0000)
            top:Integer=0 ($0000)
            right:Integer=731 ($02DB)
            bottom:Integer=447 ($01BF)
  LocVar Line:Integer=0 ($0000)
42C6 681F:0259 n CRASHME!(0001)  CrashMe.Paint    on line 141 of
file d:\bp\winwork\crashme.pas  11:13:42 12JUL94
```

```
  LocPar PaintDC:Word=3182 ($0C6E)
  PasVar PaintInfo:TPaintStruct=
            hdc:Word=3182 ($0C6E)
            fErase:WordBool=FALSE ($0000)
            rcPaint:TRect=
              left:Integer=0 ($0000)
              top:Integer=0 ($0000)
              right:Integer=731 ($02DB)
              bottom:Integer=447 ($01BF)
            fRestore:WordBool=FALSE ($0000)
            fIncUpdate:WordBool=FALSE ($0000)
            rgbReserved:Unsupported type
1C=5D,3F,C7,04,3D,00,58,00,00,00,F7,07,4F,2E,C7,04 ']?Ç.='
  PasVar Self:Object:TCrashMeWindow
  LocVar ColorTable:Pointer=$481F:0000 ^Unsupported type
1C=00,08,DC,00,33,45,AB,00,51,29,5F,00,6C,14,79,00,12,D7,0F,00,4
B,EA,5E,00,C6,53,B2,00,D8,B7,4E,00... ''
42F6 6837:175C F CRASHME!(0002)  OWindows.WMPaint    on line
2176 of file OWINDOWS.PAS  18:43:36 7JAN94
  PasVar Msg:TMessage=
            Receiver:Word=31116 ($798C)
            Message:Word=15 ($000F)
            WParam:Word=0 ($0000)
            LParam:Longint=0 ($00000000)
            Result:Longint=1 ($00000001)
            WParamLo:Byte=0 ($00)
            WParamHi:Byte=0 ($00)
            LParamLo:Word=0 ($0000)
            LParamHi:Word=0 ($0000)
            ResultLo:Word=1 ($0001)
            ResultHi:Word=0 ($0000)
  PasVar Self:Object:TWindow
[............]
435A 6837:1A44 F CRASHME!(0002)  OWindows.Run    on line 3059 of
file OWINDOWS.PAS  18:43:36 7JAN94
  PasVar Self:Object:TApplication
4364 681F:02AB F CRASHME!(0001)  CrashMe.
4366 681F:0000 n CRASHME!(0001)


Global Vars
-----------


CRASHME.CrashMe:
  TConst InitMainWindow:Special Function=InitMainWindow
$681F:0002
  TConst Paint:Special Function=Paint $681F:022D
  TConst CrashMeApp:Object:TCrashMeApplication
CRASHME.OWindows:
  TConst Init:Special Function=Init $6837:032B
  TConst Done:Special Function=Done $6837:03DF
  TConst CreateChildren:Special Function=CreateChildren
$6837:0522
  TConst Transfer:Special Function=Transfer $6837:059B
[............]
CRASHME.OMemory:
  TConst SafetyPoolSize:Word=65527 ($FFF7)
CRASHME.BugSlayImports:
  TConst AppStatusDump:Pointer=$47FF:01AD ^Function
  TConst HandleException:Pointer=$47FF:05BE ^Function
  TConst SetBugSlayOptions:Pointer=$47FF:0A9E ^Function
  TConst BugSlayLoaded:Boolean=TRUE ($01)
```

This section shows the global variables for all modules with Turbo Debugger information. Note that global variables are actually stored as typed constants. Global [untyped] constants can also shown, but their are often hundreds of these and these can be obtained easily from the source or from the object browser (BP7).

```
Code and Data selectors
-----------------------

Selector  Size      Details
--------  ----      -------
4816      00000100  Data, OWL TInstanceBlock
3FBE      00001A00  Code, segment 0004
402E      00002000  Data, Pascal Heap Block
481E      00004000  Data, Unknown Type
480E      00010000  Data, OWL Safety Pool ?
5986      00004420  Code, segment 0003
4696      00001E60  Code, segment 0002
47FE      00000BA0  Code, segment 0001
```

Shows the code and data selectors owned by the application. BugSlay attempts to identify Data segments in the following categories:

OWL TInstanceBlock	A structure internal to OWL
OWL Safety Pool	Segment may be the safety pool allocated by OMemory in OWL. Could also be created by the application as identification can only use the size to determine the match.

Pascal Heap Block	Segment is owned by the Pascal heap
Unknown	Segment was most likely created by the application directly with GlobalAlloc, indirectly with New or GetMem or by a library.

Note: Borland's Pascals use a heap sub-allocation method in Windows. Because of this, heap allocations (using New or GetMem) can appear in two ways.

Small allocations (i.e. <HeapLimit) will be allocated from larger blocks of HeapBlock bytes and will be shown as a 'Pascal Heap Block'.

Larger allocations (i.e. >HeapLimit) are allocated on the Windows global heap directly and are shown as 'Unknown', or possibly 'OWL Safety Pool'. This type of allocation is not shown in the following section.

```
Pascal Heap blocks allocated
----------------------------

Heap selector 402F has 0050 bytes (of 1FF4) allocated
 0050 bytes at 000C 'X...Œy.......'   58 00 00 00 8C 79 00 00 00
00 00 00 00
```

Shows any allocations on the Pascal Heap on termination. This information can be useful in finding the error, or in identifying memory leaks. See the options section for further information on identifying memory leaks. Note that only the first few bytes of each block is shown. A block can contain one or more contiguous allocations.

BugSlay Options

The SetOptions procedure in BugSlay.PAS gives you the opportunity to customize the behavior of BugSlay. These options are: (Defaults shown in brackets)

td_LogFileTrace	Options for the main stack trace, td_xxxx constants. (td_Normal)
td_LogFileOverview	Options for the overview stack trace (td_DoStackTrace OR td_ModuleName)
td_AuxTrace	Options for a stack trace sent to the debugging terminal (td_DoStackTrace OR td_HeapDump OR (td_ModuleName)
MaxFrames	Maximum number of stack frames to dump (10000)
HeapBytesToDump	Maximum Number of heap bytes to dump for the 'Pascal Heap Blocks Allocated' section (13)
OWLSafetyPoolSize	Used to identify OWL's SafetyPool on the global heap. Set to 0 to disable this identification. (OMemory.SafetyPoolSize)

MaxDumpSize	Maximum number of bytes to dump for an unsupported variable type. (32)
MaxUnroll	Unroll records and dereference pointers this many levels. Use caution when changing as it will unravel linked lists, possibly leading to huge, possibly useful, log files.(3)

DoHeapAllocationCheck	If true, check for remaining heap allocations on application exit, even if no error occurred. Used to help identify memory leaks. Users who routinely leave memory allocated, relying on Windows to tidy up afterwards will want to either re-evaluate this practice or disable this function. (True)
AuxName	If you have a secondary (monochrome) screen attached to your development machine you can set this to the name of any character driver you may use for the secondary display. For example, if you use MDRV.SYS to drive the screen, use 'MDRV' here. Or 'AUX' for OX.SYS. If you do not have any auxiliary output, set to 'Nul'. ('Nul')
ErrorLogFilename	Base filename of the error log. Must not include a period or extension. BugSlay makes a copy of this string, so if dynamically allocated it can be destroyed after the call to SetOptions. ('c:\error')

See BugSlayImports for details of the td_xxxx flags.

Runtime errors

It seems common practice among Pascal programmers to disable Range, Stack, I/O and Overflow checking. While such practices have their place, indeed in the case of I/O checking it often essential, it can lead to bugs going unnoticed.

BugSlay will be triggered by Range, I/O and Overflow errors **only** if those options are enabled. Here at Apsley-Bolton Computers, we have all available checking enabled right up until a product ships. I/O checking is the exception in that most I/O related functions must be allowed to fail with the application taking over the error recovery.

I do not want to preach hear though. If you feel the advantages of testing without checking (faster, smaller code, relaxed use of data types) out weigh the disadvantages (slightly larger, slower code, fewer dormant bugs), then go right ahead.

At this time BugSlay does not trap Stack overflow run-time errors. Stack related exceptions are supported however.

Compiler Options

The following compiler options should be selected for optimal functionality:

Directive	Name	Purpose
{$W-}	Windows stack frame	Disables creation of the special stack format for real mode Windows. This stack format is not supported.
{$D+}	Debug Information	Generates symbols
{$L+}	Local symbols	Includes local symbols in the debug information
{$Q+}	Overflow checking	Checks for Overflow run-time errors
{$R+}	Range checking	Checks for Range run-time errors
none	Debug Info in EXE-On	Appends debug info to the EXE or DLL file.

Note: For BugSlay to function correctly, the 'Windows Stack Frame' compiler option must be off {$W-}.

Variable Information

Local variable and parameter information as well as global variable information can be logged for modules having Turbo Debugger information. Many data types are supported by the compiler, but BugSlay can only process the following types:

Type name	Size (Bytes)	Shown as
String	256	String
ShortInt	1	Decimal ($Hex)
Integer	2	Decimal ($Hex)
Longint	4	Decimal ($Hex)
Comp	8	Decimal
Byte	1	Decimal ($Hex)
Word	2	Decimal ($Hex)
Char	1	Char ($Hex)
Single	4	Decimal
Real	6	Decimal
Double	8	Decimal
Extended	10	Decimal
Untyped Pointer	4	$Hex
Typed Pointer	4	Pointer is dereferenced and processed

		if possible. Note 1
Boolean	1	Boolean ($Hex)
WordBool	2	Boolean ($Hex)
LongBool	4	Boolean ($Hex)
Object		Object name
Function	4	Name, if available
Special Function	4	Name, if available
Enumerated type	1 or 2	Name, if available. ($Hex)
Text		File name, mode, handle and buffer size
File		File name, mode, handle and record size
Record		Evaluated where possible. Note 1

Note 1: Record evaluation and pointer dereferencing is available only in the full, registered version of BugSlay. See 'Registering BugSlay'.

Types other than those above are shown as a sequence of hex bytes and as a string, limited to MaxDumpSize bytes (see BugSlay Options), along with the type code. A listing of the type codes can be found in the Borland Open Architecture Handbook for Pascal.

Objects are not supported in this version. However, in stack traces, object methods contain a Self variable that provides the name of the object the method resides in.

Variables, Constants and parameters have a prefix indicating what they are:

Prefix	**Shown for**
TConst	Typed Constants and global variables
ABS	Absolute variable
LocPar	Local Parameter, passed to function/procedure
LocVar	Local Variable
PasVar	Variable passed to function/procedure as a VAR
Const	Untyped constant

Contacting Us

Where ever possible, contact us via email at:

CompuServe:	70651,1611
Internet:	70651.1611@compuserve.com
X.400:	c=US; admd=CompuServe; prmd=CSMaill; dda, ID=70651.1611

By mail at:

Apsley-Bolton Computers, Inc.
P.O. Box 277
Bolton
MA 01740
USA

Or by fax at:

+1 (508) 779 0454

For sales ONLY, call:

1 800 625 3316 in the USA, or
+1 (508) 779 5043

Please note that our sales department will not answer non-sales questions.

License Agreement

BugSlay is not and has never been public domain software, nor is it free software.

Users of the unregistered, special demo version of BugSlay are granted a limited license to use demo version BugSlay on a 30-day trial basis for the purpose of determining whether BugSlay is suitable for their needs. The use of BugSlay, except for the initial 30-day trial, requires registration. The use of unregistered, copies of BugSlay, outside of the initial 30-day trial, by any person, business, corporation, government agency or any other entity is strictly prohibited.

A limited license is granted to copy and distribute the unregistered, Special Demo of BugSlay only for the trial use of others, subject to the above limitations, and also the following:

i) The demo version of BugSlay must be copied in unmodified form, complete with the file containing this license information.
ii) The full machine-readable BugSlay documentation must be included with each copy.
iii) The demo version of BugSlay may not be distributed in conjunction with any

other product without a special license to do so from Apsley-Bolton Computers.

iv) No fee, charge, or other compensation may be requested or accepted, except as authorized below:

a) Operators of electronic bulletin board systems (sysops) may make the demo version of BugSlay available for downloading only as long as the above conditions are met. An overall or time-dependent charge for the use of the bulletin board system is permitted as long as there is not a specific charge for the download of the demo version of BugSlay.

b) Vendors of user-supported or shareware software may distribute the demo version of BugSlay, subject to the above conditions, without specific permission.

Vendors may charge a disk duplication and handling fee, which, when pro-rated to the demo version of the BugSlay product, may not exceed eight United States dollars.

c) Non-profit user groups may distribute the demo version of BugSlay, subject to the above conditions, without specific permission. Non-profit user groups may charge a disk duplication and handling fee, which, when pro-rated to the demo version of the BugSlay product, may not exceed eight United States dollars.

v) Nothing may be added to, deleted from or changed in the archive file which contains BugSlay. This includes adding ZIP file comments, BBS or any other advertisement.

Distribution of the BugSlay Unit files (*.PAS) supplied in the BugSlay demo archive in a compiled state, including but not limited to *.TPU, *.TPW and *.EXE is expressly prohibited.

Limited Warranty

Apsley-Bolton Computers Inc. does not warrant or make any representations regarding the use, or the results of use, of this product in terms of correctness, accuracy, reliability, or otherwise. You assume the responsibility for the use of this product to achieve your intended results, and for the results actually obtained.

Apsley-Bolton Computer Inc. disclaims all other warranties, either expressed or implied, including but not limited to implied warranties of Merchantability and fitness for a particular purpose, with respect to the BugSlay software and the accompanying documentation.

In no event shall Apsley-Bolton Computers, Inc. be liable for any damages whatsoever, including, without limitation, damages for loss of business profits, business interruption, consequential, special, incidental, punitive or indirect damages, loss of business information or other pecuniary loss, arising out of the use of or inability to use the BugSlay product. Apsley-Bolton Computers Inc. entire liability to you and your sale and exclusive remedy for damages under this Agreement, regardless of the form of action, shall not exceed the purchase price paid by you for the license.

This limited warranty gives you specific legal rights and you may have other rights which vary from state to state. The exclusion and limitations or implied warranties, and the exclusion and limitations of liability for special, incidental, or consequential damages, are not allowed by some states and may not apply to you.

This agreement shall be governed by the laws of the Commonwealth of Massachusetts.

Registering BugSlay

This unregistered, special demo version of BugSlay provides the basic functions for you to evaluate BugSlay's functionality, convenience and compatibility. You are licensed to use it for 30 days to evaluate BugSlay's suitability to your needs. After this period you are required to either register BugSlay or cease using it.

The unregistered, special demo version of BugSlay may not be redistributed with your application.

By registering BugSlay you will be licensed to use the full version of BugSlay and you also will receive:

- Full record and pointer support. Unroll records and deference pointers in variable dumps. See the sample error log for example.
- Log files may contain more than one error 'event' allowing users to log several errors.
- No Unregistered 'nag' screens appearing on errors.
- Notice of any future releases.

On CompuServe

Members of the CompuServe Information Service can register BugSlay on-line for a special CompuServe price of $40.00.

From any ! prompt, type:

```
GO SWREG
```

and follow the prompts. The registration ID for BugSlay is 3032.

Your registered version of BugSlay will be sent via CompuServe E-Mail.

By phone

Call the Apsley-Bolton Computers' sales line at:

1 800 625 3316 or, outside the United States,
+1 (508) 779 5043

Pricing is as shown on the order form below.

Please have your credit card ready. Visa and Mastercard accepted.

Your registered version of BugSlay will be sent by first class mail, or air mail for deliveries outside the United States.

Note: The sales personnel will not answer non-sales questions.

By mail

To receive your registered BugSlay via mail complete the order form below and mail to:

Apsley-Bolton Computers, Inc.
P.O. Box 277
Bolton
MA 01740
USA

Or fax to:

+1 (508) 779 0454

Your registered version of BugSlay will be sent by first class mail, or air mail for deliveries outside the United States.

BugSlay 1.0 order form

Name: ______________________________

Company: ______________________________

Address: ______________________________

____

City: ______________________________

State: ______________________________

Zip/Postal code: ____________________ Country: ______________

Phone: ______________________________

Email address (Optional): ______________________________

________ BugSlays @$45.00 each $______
Disk size (check one): [ ] 5.25" [ ] 3.5"

Massachusetts residents add Sales Tax @5% $______

Foreign air shipping $4.50 $______
(for addresses outside the US and Canada)

Total enclosed: $______

Please enclose a check payable to Apsley-Bolton Computers Inc.; or you may use Visa or MasterCard. For credit cards, please enter the information below:

Card #:_____ - _____ - _____ - _____ Expiration date: ____/____

Signature: ______________________________

Where did you get your special demo copy of BugSlay?

____

Comments:

____

____

Trademarks

BugSlay is a trademark of Apsley-Bolton Computers, Inc.

Borland, Turbo Pascal and Turbo Debugger are registered trademarks of Borland International Inc.

Windows is a trademark of Microsoft Corporation.
Microsoft is a registered trademark of Microsoft Corporation.