

Windows Programming in Assembly Language

1	Readme.txt	9
1.1:	Chapter Overview	9
1.2:	Petzold, Yao, Boling, and Other Acknowledgements	9
1.3:	Ground Zero: The Programmer's Challenge	9
1.4:	The Tools	11
1.5:	Why HLA?	12
1.6:	The Ground Rules	13
1.7:	Using Make/NMake	14
1.8:	The HLA Integrated Development Environment	24
1.9:	Debugging HLA Programs Under Windows	25
1.10:	Other Tools of Interest	25
1.11:	Windows Programming Documentation	25
2	Advanced HLA Programming	26
2.1:	Using Advanced HLA Features	26
2.2:	HLA's High-Level Data Structure Facilities	26
2.2.1:	Basic Data Types	26
2.2.1.1:	Eight Bit Data Types	27
2.2.1.2:	Sixteen-Bit Data Types	29
2.2.1.3:	Thirty-Two-Bit Data Types	29
2.2.1.4:	Sixty-Four-Bit Data Types	30
2.2.1.5:	Eighty-Bit Data Types	31
2.2.1.6:	One Hundred Twenty-Eight Bit Data Types	31
2.2.2:	Composite Data Types	31
2.2.2.1:	HLA Array Types	32
2.2.2.2:	HLA Union Types	36
2.2.2.3:	HLA Record (Structure) Types	38
2.2.3:	Nested and Anonymous Unions and Records	44
2.2.4:	Pointer Types	46
2.2.5:	Thunk Types	47
2.2.6:	Type Coercion	48
2.3:	HLA High-Level Control Structures	49

2.3.1: Boolean Control Expressions	50
2.3.2: The HLA IF..ENDIF Statement	56
2.3.3: The HLA WHILE..ENDWHILE Statement.....	59
2.3.4: The HLA REPEAT..UNTIL Statement	61
2.3.5: The HLA FOR Loops	62
2.3.6: HLA's BREAK, CONTINUE, and EXIT Statements.....	64
2.3.7: Exception Handling Statements in HLA.....	66
2.4: HLA Procedure Declarations and Invocations	73
2.4.1: Disabling HLA's Automatic Code Generation for Procedures	80
2.4.3: Parameter Passing in HLA, Value Parameters.....	86
2.4.4: Parameter Passing in HLA: Reference Parameters	87
2.4.5: Untyped Reference Parameters	89
2.4.6: Hybrid Parameter Passing in HLA.....	90
2.4.7: Parameter Passing in HLA, Register Parameters	91
2.4.8: Passing Pointers and Values as Reference Parameters.....	92
2.5: The HLA Compile-Time Language.....	96
2.5.1: Compile-Time Assignment Statements	98
2.5.2: Compile-Time Functions.....	105
2.5.3: Generating Code With a Compile-Time Statement	110
2.5.4: Conditional Assembly (#if..#elseif..#else..#endif)	110
2.5.5: The #for..#endfor Compile-Time Loop.....	113
2.5.6: The #while..#endwhile Compile-Time Loop.....	115
2.5.7: Compile-Time I/O and Data Facilities	116
2.5.8: Macros (Compile-Time Procedures and Functions)	119
2.5.9: Performance of the HLA Compile-Time Language.....	130
2.5.10: A Complex Macro Example: stdout.put	131
2.6: Even More Advanced HLA Programming... ..	140
3 The C - Assembly Connection	141
3.1: Why are We Reading About C?	141
3.2: Basic C Programming From an Assembly Perspective	141
3.2.1: C Scalar Data Types.....	142
3.2.1.1: C and Assembler Integer Data Types.....	142
3.2.1.2: C and Assembly Character Types	144

3.2.1.3: C and Assembly Floating Point (Real) Data Types	145
3.2.2: C and Assembly Composite Data Types	146
3.2.2.1: C and Assembly Array Types	147
3.2.2.2: C and Assembly Record/Structure Types.....	148
3.2.2.3: C and Assembly Union Types	152
3.2.2.4: C and Assembly Character String Types.....	153
3.2.2.5: C++ and Assembly Class Types	161
3.2.3: C and Assembly Pointer Types	161
3.2.4: C and Assembly Language Constants	169
3.2.5: Arithmetic Expressions in C and Assembly Language	177
3.2.5.1: Converting Simple Expressions Into Assembly Language	180
3.2.5.2: Operator Precedence	189
3.2.5.3: Associativity	189
3.2.5.4: Side Effects and Sequence Points	190
3.2.5.5: Translating C/C++ Expressions to Assembly Language	194
3.2.6: Control Structures in C and Assembly Language.....	199
3.2.6.1: Boolean Expressions in HLA Statements	199
3.2.6.2: Converting C/C++ Boolean Expressions to HLA Boolean Expressions	202
3.2.6.3: The IF Statement	203
3.2.6.4: The SWITCH/CASE Statement	209
3.2.6.5: The WHILE Loop	212
3.2.6.6: The DO..WHILE Loop.....	213
3.2.6.7: The C/C++ FOR Loop	214
3.2.6.8: Break and Continue.....	216
3.2.6.9: The GOTO Statement	216
3.3: Function Calls, Parameters, and the Win32 Interface.....	217
3.3.1: C Versus C++ Functions	217
3.3.2: The Intel 80x86 ABI (Application Binary Interface).....	218
3.3.2.1: Register Preservation and Scratch Registers in Win32 Calls.....	218
3.3.2.2: The Stack Pointer (ESP).....	218
3.3.2.3: The Direction Flag	219
3.3.2.4: Function Return Results	219
3.3.2.5: Data Alignment and Padding.....	219
3.3.3: The C, Pascal, and Stdcall Calling Conventions	220
3.3.4: Win32 Parameter Types	222
3.3.5: Pass by Value Versus Pass by Reference	225
3.4: Calling Win32 API Functions.....	231
3.5: Win32 API Functions and Unicode Data.....	233
3.6: Win32 API Functions and the Parameter Byte Count.....	236
3.7: Creating HLA Procedure Prototypes for Win32 API Functions	236

3.7.1: C/C++ Naming Conventions Versus HLA Naming Conventions	236
3.7.1.1: Reserved Word Conflicts	237
3.7.1.2: Alphabetic Case Conflicts	238
3.7.1.3: Common C/C++ Naming Conventions	238
3.7.1.4: Hungarian Notation	241
3.8: The w.hhf Header File	244
3.9: And Now, on to Assembly Language!	245
4 The RadASM IDE for HLA	246
4.1: Integrated Development Environments	246
4.2: Traditional (Command Line) Development in Assembly	246
4.3: HLA Project Organization	247
4.4: Setting Up RadASM to Work With HLA	248
4.4.1: The RADASM.INI Initialization File	248
4.4.2: The HLA.INI Initialization File	251
4.4.3: Additional Support for RadASM on the CD-ROM	258
4.5: Running RadASM	258
4.5.1: The RadASM Project Management Window	259
4.5.2: Creating a New Project in RadASM	267
4.5.3: Working With RadASM Projects	270
4.5.4: Editing HLA Source Files Within RadASM	272
4.6: Creating and Compiling HLA Projects With RadASM	277
4.7: Developing Small Projects with RadASM	284
4.8: Plus More!	285
5 The Event-Oriented Programming Paradigm	286
5.1: Who Dreamed Up This Nonsense?	286
5.2: Message Passing	287
5.3: Handles	289
5.4: The Main Program	290
5.4.1: Filling in the w.WNDCLASSEX Structure and Registering the Window	290
5.4.2: "What is a 'Window Class' Anyway?"	295
5.4.3: Creating and Displaying a Window	297

5.4.4: The Message Processing Loop	302
5.4.5: The Complete Main Program.....	303
5.5: The Window Procedure.....	304
5.6: Hello World.....	309
5.7: Compiling and Running HelloWorld From the Command Line	316
5.8: Compiling and Running HelloWorld from RadASM.....	318
5.9: Goodbye World!	318
6 Text in a GUI World	320
6.1: Text Display Under Windows	320
6.2: Painting	321
6.2.1: Device Contexts.....	322
6.2.2: Device Context Attributes	324
6.2.3: Painting Text in the Client Area	325
6.2.4: BeginPaint, EndPaint, GetDC, GetWindowDC, and ReleaseDC Macros.....	347
6.3: Device Capabilities.....	352
6.4: Typefaces and Fonts.....	368
6.5: Scroll Bars	404
6.6: The DebugWindow Application	436
6.6.1: Message Passing Under Windows	436
6.6.2: Memory Protection and Messages	441
6.6.3: Coding the DebugWindow Application	443
6.6.4: Using DebugWindow	469
6.7: The Windows Console API	475
6.7.1: Windows' Dirty Secret - GUI Apps Can Do Console I/O!.....	476
6.7.2: Win32 Console Functions.....	482
6.7.2.1: w.AllocConsole	482
6.7.2.2: w.CreateConsoleScreenBuffer	483
6.7.2.3: w.FillConsoleOutputAttribute	483
6.7.2.4: w.FillConsoleOutputCharacter	485
6.7.2.5: w.FlushConsoleInputBuffer	486
6.7.2.6: w.FreeConsole.....	486
6.7.2.7: w.GetConsoleCursorInfo	486
6.7.2.8: w.GetConsoleScreenBufferInfo	487

6.7.2.9: w.GetConsoleTitle	488
6.7.2.10: w.GetConsoleWindow	489
6.7.2.11: w.GetNumberOfConsoleInputEvents	489
6.7.2.12: w.GetStdHandle	490
6.7.2.13: w.PeekConsoleInput.....	490
6.7.2.14: w.ReadConsole	493
6.7.2.15: w.ReadConsoleInput	494
6.7.2.16: w.ReadConsoleOutput	494
6.7.2.17: w.ReadConsoleOutputAttribute	495
6.7.2.18: w.ReadConsoleOutputCharacter.....	496
6.7.2.19: w.ScrollConsoleScreenBuffer	496
6.7.2.20: w.SetConsoleActiveScreenBuffer.....	498
6.7.2.21: w.SetConsoleCursorInfo.....	498
6.7.2.22: w.SetConsoleCursorPosition	498
6.7.2.23: w.SetConsoleScreenBufferSize	499
6.7.2.24: w.SetConsoleTextAttribute	499
6.7.2.25: w.SetConsoleTitle.....	500
6.7.2.26: w.SetConsoleWindowInfo.....	500
6.7.2.27: w.SetStdHandle	501
6.7.2.28: w.WriteConsole.....	501
6.7.2.29: w.WriteConsoleInput.....	502
6.7.2.30: w.WriteConsoleOutput.....	502
6.7.2.31: w.WriteConsoleAttribute	503
6.7.2.32: w.WriteConsoleOutputCharacter	504
6.7.2.33: Plus More!.....	505
6.7.3: HLA Standard Library Console Support	505
6.8: This Isn't the Final Word!	505
7 Graphics.....	506
7.1: Graphics in a GUI World	506
7.2: Types of Graphic Objects You Can Draw in Windows	506
7.3: Facilities That the Windows GDI Provides	506
7.4: The Device Context.....	507
7.5: Obtaining and Using Device Context Information	508
7.6: Line Drawing Under Windows	517
7.6.1: Background Mode and Color for Lines	521
7.6.2: Using the LineTo and MoveToEx API Functions	522
7.6.3: Using the PolyLineTo API Function	533
7.6.4: A Software Engineering Break	540
7.6.5: The FofX.hla Application	550

7.6.6: Pen Drawing Modes	559
7.6.7: Paths	568
7.6.8: Extended Pen Styles	570
7.7: Bounding Boxes and Rectangles in Windows.....	580
7.8: Drawing Objects in Windows	586
7.8.1: Filling Objects and Windows' Brushes.....	586
7.8.2: Filling and Framing Paths.....	589
7.8.3: Drawing Rectangles.....	593
7.8.4: Drawing Circles and Ellipses	599
7.8.5: Drawing Roundangles, Arcs, Chords, and Pies.....	605
7.8.6: Bezier Curves	610
7.8.7: Drawing with the Polygon Function	616
7.9: Regions	622
7.10: Clipping	629
7.11: Bitmaps	636
7.12: Metafiles.....	665
7.13: Additional GDI Functions of Interest	671
7.14: For More Information.....	674
8 Event-Driven Input	675
8.1: Event-Driven Input Versus "Hurry Up and Wait"	675
8.2: Focus on Focus.....	675
8.3: Stick and Caret.....	676
8.4: Keyboard Messages	679
8.5: Mouse Messages	735
8.11: Timer Messages.....	776
8.13: But Wait! There's More!	785
9 Debugging Windows Applications	786
9.1:	786
10 Controls, Dialogs, Menus, and Windows	787

10.1:	787
11 The Resource Compiler	788
11.1:	788
12 Icons, Cursors, Bitmaps, and Strings	789
12.1:	789
13 File I/O and Other Traditional Kernel Services	790
13.1:	790
1 Windows Programming in Assembly Language	791