

WolfNet 6502 WorkBench Computer Emulator

beta

Generated by WolfNet Computing using Doxygen 1.9.5

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	2
2.1 Class Hierarchy	2
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	10
5.1 Emulator Namespace Reference	10
5.2 Emulator.Model Namespace Reference	11
5.3 Emulator.ViewModel Namespace Reference	11
5.4 Hardware Namespace Reference	11
5.4.1 Enumeration Type Documentation	12
5.5 XamlGeneratedNamespace Namespace Reference	13
6 Class Documentation	13
6.1 Hardware.MemoryMap.Devices.ACIA Class Reference	13
6.1.1 Detailed Description	13
6.1.2 Member Data Documentation	13
6.2 Emulator.App Class Reference	14
6.2.1 Detailed Description	14
6.3 Hardware.AT28CXX Class Reference	14
6.3.1 Detailed Description	15
6.3.2 Constructor & Destructor Documentation	15
6.3.3 Member Function Documentation	15
6.3.4 Property Documentation	19
6.4 Hardware.MemoryMap.BankedRam Class Reference	21
6.4.1 Detailed Description	21
6.4.2 Member Data Documentation	21
6.4.3 Property Documentation	22
6.5 Hardware.MemoryMap.BankedRom Class Reference	22
6.5.1 Detailed Description	23
6.5.2 Member Data Documentation	23
6.5.3 Property Documentation	23
6.6 Emulator.Model.Breakpoint Class Reference	23
6.6.1 Detailed Description	24
6.6.2 Property Documentation	24
6.7 Emulator.Model.BreakpointType Class Reference	25
6.7.1 Detailed Description	25

6.7.2 Member Data Documentation	25
6.8 Hardware.MemoryMap.DeviceArea Class Reference	26
6.8.1 Detailed Description	26
6.8.2 Member Data Documentation	26
6.8.3 Property Documentation	27
6.9 Hardware.MemoryMap.Devices Class Reference	27
6.9.1 Detailed Description	27
6.10 Hardware.Disassembly Class Reference	28
6.10.1 Detailed Description	28
6.10.2 Property Documentation	28
6.11 Emulator.ExitCodes Class Reference	29
6.11.1 Detailed Description	29
6.11.2 Member Data Documentation	29
6.12 XamlGeneratedNamespace.GeneratedApplication Class Reference	31
6.12.1 Detailed Description	31
6.12.2 Member Function Documentation	32
6.12.3 Member Data Documentation	36
6.13 XamlGeneratedNamespace.GeneratedInternalTypeHelper Class Reference	36
6.13.1 Detailed Description	37
6.13.2 Member Function Documentation	37
6.14 Hardware.MemoryMap.Devices.GPIO Class Reference	42
6.14.1 Detailed Description	42
6.14.2 Member Data Documentation	42
6.15 Hardware.HM62256 Class Reference	42
6.15.1 Detailed Description	43
6.15.2 Constructor & Destructor Documentation	43
6.15.3 Member Function Documentation	43
6.15.4 Property Documentation	45
6.16 Emulator.IClosable Interface Reference	46
6.16.1 Detailed Description	46
6.16.2 Member Function Documentation	46
6.17 Emulator.ViewModel.MainViewModel Class Reference	47
6.17.1 Detailed Description	49
6.17.2 Constructor & Destructor Documentation	49
6.17.3 Member Function Documentation	50
6.17.4 Member Data Documentation	58
6.17.5 Property Documentation	59
6.18 Emulator.MainWindow Class Reference	64
6.18.1 Detailed Description	65
6.18.2 Constructor & Destructor Documentation	65
6.18.3 Member Function Documentation	66
6.18.4 Member Data Documentation	81

6.19 Hardware.MemoryMap Class Reference	81
6.19.1 Detailed Description	82
6.19.2 Member Function Documentation	82
6.19.3 Member Data Documentation	86
6.19.4 Property Documentation	86
6.20 Emulator.Model.MemoryRowModel Class Reference	87
6.20.1 Detailed Description	88
6.20.2 Property Documentation	88
6.21 Emulator.MemoryVisual Class Reference	91
6.21.1 Detailed Description	92
6.21.2 Constructor & Destructor Documentation	92
6.21.3 Member Function Documentation	92
6.21.4 Member Data Documentation	94
6.22 Emulator.ViewModel.MemoryVisualViewModel Class Reference	95
6.22.1 Detailed Description	96
6.22.2 Constructor & Destructor Documentation	96
6.22.3 Member Function Documentation	96
6.22.4 Member Data Documentation	97
6.22.5 Property Documentation	97
6.23 Hardware.MemoryMap.Devices.MM65SIB Class Reference	98
6.23.1 Detailed Description	98
6.23.2 Member Data Documentation	98
6.24 Emulator.MultiThreadedObservableCollection< T > Class Template Reference	99
6.24.1 Detailed Description	99
6.24.2 Constructor & Destructor Documentation	100
6.24.3 Member Function Documentation	101
6.24.4 Event Documentation	101
6.25 Emulator.Model.OutputLog Class Reference	101
6.25.1 Detailed Description	102
6.25.2 Constructor & Destructor Documentation	102
6.25.3 Property Documentation	102
6.26 Emulator.Versioning.Product Class Reference	104
6.26.1 Detailed Description	104
6.26.2 Member Data Documentation	104
6.27 Hardware.Versioning.Product Class Reference	106
6.27.1 Detailed Description	106
6.27.2 Member Data Documentation	106
6.28 Emulator.Model.RomFileModel Class Reference	107
6.28.1 Detailed Description	107
6.28.2 Property Documentation	107
6.29 Emulator.SaveFile Class Reference	109
6.29.1 Detailed Description	110

6.29.2 Constructor & Destructor Documentation	110
6.29.3 Member Function Documentation	110
6.29.4 Member Data Documentation	115
6.30 Emulator.ViewModel.SaveFileViewModel Class Reference	115
6.30.1 Detailed Description	116
6.30.2 Constructor & Destructor Documentation	116
6.30.3 Member Function Documentation	117
6.30.4 Member Data Documentation	117
6.30.5 Property Documentation	117
6.31 Emulator.Settings Class Reference	119
6.31.1 Detailed Description	120
6.31.2 Constructor & Destructor Documentation	120
6.31.3 Member Function Documentation	120
6.31.4 Member Data Documentation	126
6.32 Emulator.SettingsFile Class Reference	126
6.32.1 Detailed Description	126
6.32.2 Member Function Documentation	126
6.33 Emulator.Versioning.SettingsFile Class Reference	127
6.33.1 Detailed Description	127
6.33.2 Member Data Documentation	127
6.34 Emulator.Model.SettingsModel Class Reference	128
6.34.1 Detailed Description	128
6.34.2 Property Documentation	128
6.35 Emulator.ViewModel.SettingsViewModel Class Reference	129
6.35.1 Detailed Description	130
6.35.2 Constructor & Destructor Documentation	130
6.35.3 Member Function Documentation	132
6.35.4 Member Data Documentation	132
6.35.5 Property Documentation	133
6.36 Hardware.MemoryMap.SharedRom Class Reference	134
6.36.1 Detailed Description	134
6.36.2 Member Data Documentation	134
6.36.3 Property Documentation	135
6.37 Emulator.Model.StateFileModel Class Reference	135
6.37.1 Detailed Description	136
6.37.2 Property Documentation	136
6.38 Hardware.Utility Class Reference	137
6.38.1 Detailed Description	137
6.38.2 Member Function Documentation	137
6.39 Emulator.Versioning Class Reference	142
6.39.1 Detailed Description	142
6.40 Emulator.ViewModel.ViewModelLocator Class Reference	142

6.40.1 Detailed Description	142
6.40.2 Constructor & Destructor Documentation	143
6.40.3 Member Function Documentation	143
6.40.4 Property Documentation	143
6.41 Hardware.W65C02 Class Reference	144
6.41.1 Detailed Description	147
6.41.2 Constructor & Destructor Documentation	147
6.41.3 Member Function Documentation	147
6.41.4 Member Data Documentation	181
6.41.5 Property Documentation	182
6.42 Hardware.W65C22 Class Reference	185
6.42.1 Detailed Description	187
6.42.2 Constructor & Destructor Documentation	187
6.42.3 Member Function Documentation	187
6.42.4 Member Data Documentation	191
6.42.5 Property Documentation	193
6.43 Hardware.W65C51 Class Reference	195
6.43.1 Detailed Description	197
6.43.2 Constructor & Destructor Documentation	197
6.43.3 Member Function Documentation	197
6.43.4 Member Data Documentation	207
6.43.5 Property Documentation	208
6.44 Emulator.Window1 Class Reference	210
6.44.1 Detailed Description	211
6.44.2 Member Function Documentation	211
6.44.3 Member Data Documentation	212
7 File Documentation	212
7.1 Emulator/App.xaml.cs File Reference	212
7.2 App.xaml.cs	212
7.3 Emulator/Classes/ExitCodes.cs File Reference	213
7.4 ExitCodes.cs	213
7.5 Emulator/Classes/FileLocations.cs File Reference	213
7.6 FileLocations.cs	213
7.7 Hardware/Classes/FileLocations.cs File Reference	213
7.8 FileLocations.cs	214
7.9 Emulator/Classes/SettingsFile.cs File Reference	214
7.10 SettingsFile.cs	214
7.11 Emulator/Classes/Versioning.cs File Reference	214
7.12 Versioning.cs	215
7.13 Hardware/Classes/Versioning.cs File Reference	215
7.14 Versioning.cs	215

7.15 Emulator/Interfaces/IClosable.cs File Reference	216
7.16 IClosable.cs	216
7.17 Emulator/MainWindow.xaml.cs File Reference	216
7.18 MainWindow.xaml.cs	216
7.19 Emulator/MemoryVisual.xaml.cs File Reference	217
7.20 MemoryVisual.xaml.cs	217
7.21 Emulator/Model/Breakpoint.cs File Reference	218
7.22 Breakpoint.cs	218
7.23 Emulator/Model/BreakpointType.cs File Reference	218
7.24 BreakpointType.cs	219
7.25 Emulator/Model/MemoryRowModel.cs File Reference	219
7.26 MemoryRowModel.cs	219
7.27 Emulator/Model/OutputLog.cs File Reference	220
7.28 OutputLog.cs	221
7.29 Emulator/Model/RomFileModel.cs File Reference	221
7.30 RomFileModel.cs	222
7.31 Emulator/Model/SettingsModel.cs File Reference	222
7.32 SettingsModel.cs	222
7.33 Emulator/Model/StateFileModel.cs File Reference	223
7.34 StateFileModel.cs	223
7.35 Emulator/MultiThreadedCollection.cs File Reference	224
7.36 MultiThreadedCollection.cs	224
7.37 Emulator/obj/x86/Debug/.NETFramework,Version=v4.8.AssemblyAttributes.cs File Reference	225
7.38 .NETFramework,Version=v4.8.AssemblyAttributes.cs	225
7.39 Emulator/obj/x86/Publish/.NETFramework,Version=v4.8.AssemblyAttributes.cs File Reference	225
7.40 .NETFramework,Version=v4.8.AssemblyAttributes.cs	225
7.41 Emulator/obj/x86/Release/.NETFramework,Version=v4.8.AssemblyAttributes.cs File Reference	226
7.42 .NETFramework,Version=v4.8.AssemblyAttributes.cs	226
7.43 Hardware/obj/Debug/.NETFramework,Version=v4.8.AssemblyAttributes.cs File Reference	226
7.44 .NETFramework,Version=v4.8.AssemblyAttributes.cs	226
7.45 Hardware/obj/Publish/.NETFramework,Version=v4.8.AssemblyAttributes.cs File Reference	226
7.46 .NETFramework,Version=v4.8.AssemblyAttributes.cs	226
7.47 Hardware/obj/Release/.NETFramework,Version=v4.8.AssemblyAttributes.cs File Reference	226
7.48 .NETFramework,Version=v4.8.AssemblyAttributes.cs	226
7.49 Emulator/obj/x86/Debug/App.g.cs File Reference	226
7.50 App.g.cs	227
7.51 Emulator/obj/x86/Publish/App.g.cs File Reference	228
7.52 App.g.cs	228
7.53 Emulator/obj/x86/Release/App.g.cs File Reference	229
7.54 App.g.cs	229
7.55 Emulator/obj/x86/Debug/App.g.i.cs File Reference	230
7.56 App.g.i.cs	231

7.57 Emulator/obj/x86/Publish/App.g.i.cs File Reference	232
7.58 App.g.i.cs	232
7.59 Emulator/obj/x86/Release/App.g.i.cs File Reference	233
7.60 App.g.i.cs	233
7.61 Emulator/obj/x86/Debug/Emulator_Content.g.cs File Reference	234
7.62 Emulator_Content.g.cs	234
7.63 Emulator/obj/x86/Publish/Emulator_Content.g.cs File Reference	235
7.64 Emulator_Content.g.cs	235
7.65 Emulator/obj/x86/Release/Emulator_Content.g.cs File Reference	235
7.66 Emulator_Content.g.cs	235
7.67 Emulator/obj/x86/Debug/Emulator_Content.g.i.cs File Reference	235
7.68 Emulator_Content.g.i.cs	235
7.69 Emulator/obj/x86/Publish/Emulator_Content.g.i.cs File Reference	236
7.70 Emulator_Content.g.i.cs	236
7.71 Emulator/obj/x86/Release/Emulator_Content.g.i.cs File Reference	236
7.72 Emulator_Content.g.i.cs	236
7.73 Emulator/obj/x86/Debug/GeneratedInternalTypeHelper.g.cs File Reference	236
7.74 GeneratedInternalTypeHelper.g.cs	236
7.75 Emulator/obj/x86/Publish/GeneratedInternalTypeHelper.g.cs File Reference	236
7.76 GeneratedInternalTypeHelper.g.cs	236
7.77 Emulator/obj/x86/Release/GeneratedInternalTypeHelper.g.cs File Reference	236
7.78 GeneratedInternalTypeHelper.g.cs	236
7.79 Emulator/obj/x86/Debug/GeneratedInternalTypeHelper.g.i.cs File Reference	237
7.80 GeneratedInternalTypeHelper.g.i.cs	237
7.81 Emulator/obj/x86/Publish/GeneratedInternalTypeHelper.g.i.cs File Reference	238
7.82 GeneratedInternalTypeHelper.g.i.cs	238
7.83 Emulator/obj/x86/Release/GeneratedInternalTypeHelper.g.i.cs File Reference	239
7.84 GeneratedInternalTypeHelper.g.i.cs	239
7.85 Emulator/obj/x86/Debug/MainWindow.g.cs File Reference	240
7.86 MainWindow.g.cs	240
7.87 Emulator/obj/x86/Publish/MainWindow.g.cs File Reference	247
7.88 MainWindow.g.cs	247
7.89 Emulator/obj/x86/Release/MainWindow.g.cs File Reference	254
7.90 MainWindow.g.cs	254
7.91 Emulator/obj/x86/Debug/MainWindow.g.i.cs File Reference	261
7.92 MainWindow.g.i.cs	261
7.93 Emulator/obj/x86/Publish/MainWindow.g.i.cs File Reference	268
7.94 MainWindow.g.i.cs	268
7.95 Emulator/obj/x86/Release/MainWindow.g.i.cs File Reference	275
7.96 MainWindow.g.i.cs	275
7.97 Emulator/obj/x86/Debug/MemoryVisual.g.cs File Reference	282
7.98 MemoryVisual.g.cs	282

7.99 Emulator/obj/x86/Release/MemoryVisual.g.cs File Reference	284
7.100 MemoryVisual.g.cs	284
7.101 Emulator/obj/x86/Debug/MemoryVisual.g.i.cs File Reference	285
7.102 MemoryVisual.g.i.cs	286
7.103 Emulator/obj/x86/Release/MemoryVisual.g.i.cs File Reference	287
7.104 MemoryVisual.g.i.cs	287
7.105 Emulator/obj/x86/Debug/SaveFile.g.cs File Reference	289
7.106 SaveFile.g.cs	289
7.107 Emulator/obj/x86/Publish/SaveFile.g.cs File Reference	291
7.108 SaveFile.g.cs	291
7.109 Emulator/obj/x86/Release/SaveFile.g.cs File Reference	293
7.110 SaveFile.g.cs	293
7.111 Emulator/obj/x86/Debug/SaveFile.g.i.cs File Reference	295
7.112 SaveFile.g.i.cs	295
7.113 Emulator/obj/x86/Publish/SaveFile.g.i.cs File Reference	297
7.114 SaveFile.g.i.cs	297
7.115 Emulator/obj/x86/Release/SaveFile.g.i.cs File Reference	299
7.116 SaveFile.g.i.cs	299
7.117 Emulator/obj/x86/Debug/Settings.g.cs File Reference	301
7.118 Settings.g.cs	301
7.119 Emulator/obj/x86/Publish/Settings.g.cs File Reference	303
7.120 Settings.g.cs	303
7.121 Emulator/obj/x86/Release/Settings.g.cs File Reference	305
7.122 Settings.g.cs	305
7.123 Emulator/obj/x86/Debug/Settings.g.i.cs File Reference	307
7.124 Settings.g.i.cs	307
7.125 Emulator/obj/x86/Publish/Settings.g.i.cs File Reference	308
7.126 Settings.g.i.cs	309
7.127 Emulator/obj/x86/Release/Settings.g.i.cs File Reference	310
7.128 Settings.g.i.cs	311
7.129 Emulator/obj/x86/Release/MemoryMap.g.i.cs File Reference	312
7.130 MemoryMap.g.i.cs	313
7.131 Emulator/obj/x86/Release/Window1.g.i.cs File Reference	314
7.132 Window1.g.i.cs	314
7.133 Emulator/Properties/AssemblyInfo.cs File Reference	315
7.134 AssemblyInfo.cs	315
7.135 Hardware/Properties/AssemblyInfo.cs File Reference	316
7.136 AssemblyInfo.cs	316
7.137 Emulator/SaveFile.xaml.cs File Reference	316
7.138 SaveFile.xaml.cs	317
7.139 Emulator/Settings.xaml.cs File Reference	317
7.140 Settings.xaml.cs	317

7.141 Emulator/ViewModel/MainViewModel.cs File Reference	318
7.141.1 Typedef Documentation	318
7.142 MainViewModel.cs	319
7.143 Emulator/ViewModel/MemoryVisualViewModel.cs File Reference	328
7.144 MemoryVisualViewModel.cs	328
7.145 Emulator/ViewModel/SaveFileViewModel.cs File Reference	329
7.146 SaveFileViewModel.cs	330
7.147 Emulator/ViewModel/SettingsViewModel.cs File Reference	331
7.148 SettingsViewModel.cs	331
7.149 Emulator/ViewModel/ViewModelLocator.cs File Reference	332
7.150 ViewModelLocator.cs	333
7.151 Hardware/Classes/AddressingMode.cs File Reference	334
7.152 AddressingMode.cs	334
7.153 Hardware/Classes/Disassembly.cs File Reference	335
7.154 Disassembly.cs	335
7.155 Hardware/Classes/MemoryMap.cs File Reference	336
7.156 MemoryMap.cs	336
7.157 Hardware/Classes/Utility.cs File Reference	339
7.158 Utility.cs	339
7.159 Hardware/Hardware/AT28CXX.cs File Reference	343
7.160 AT28CXX.cs	343
7.161 Hardware/Hardware/HM62256.cs File Reference	345
7.162 HM62256.cs	345
7.163 Hardware/Hardware/W65C02.cs File Reference	347
7.164 W65C02.cs	347
7.165 Hardware/Hardware/W65C22.cs File Reference	376
7.166 W65C22.cs	376
7.167 Hardware/Hardware/W65C51.cs File Reference	380
7.168 W65C51.cs	380
Index	389

1 Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Emulator	10
Emulator.Model	11
Emulator.ViewModel	11

Hardware	11
XamlGeneratedNamespace	13

2 Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Hardware.MemoryMap.Devices.ACIA	13
Emulator.App	14
System.Windows.Application	
XamlGeneratedNamespace.GeneratedApplication	31
XamlGeneratedNamespace.GeneratedApplication	31
XamlGeneratedNamespace.GeneratedApplication	31
XamlGeneratedNamespace.GeneratedApplication	31
XamlGeneratedNamespace.GeneratedApplication	31
XamlGeneratedNamespace.GeneratedApplication	31
Hardware.AT28CXX	14
Hardware.MemoryMap.BankedRam	21
Hardware.MemoryMap.BankedRom	22
Emulator.Model.Breakpoint	23
Emulator.Model.BreakpointType	25
Hardware.MemoryMap.DeviceArea	26
Hardware.MemoryMap.Devices	27
Hardware.Disassembly	28
Emulator.Model.OutputLog	101
Emulator.ExitCodes	29
Hardware.MemoryMap.Devices.GPIO	42
Hardware.HM62256	42
Emulator.IClosable	46
Emulator.MainWindow	64
System.Windows.Markup.IComponentConnector	
Emulator.MainWindow	64
Emulator.MainWindow	64

Emulator.MainWindow	64
Emulator.MainWindow	64
Emulator.MainWindow	64
Emulator.MainWindow	64
Emulator.MemoryVisual	91
Emulator.MemoryVisual	91
Emulator.MemoryVisual	91
Emulator.MemoryVisual	91
Emulator.SaveFile	109
Emulator.SaveFile	109
Emulator.SaveFile	109
Emulator.SaveFile	109
Emulator.SaveFile	109
Emulator.SaveFile	109
Emulator.Settings	119
Emulator.Settings	119
Emulator.Settings	119
Emulator.Settings	119
Emulator.Settings	119
Emulator.Settings	119
Emulator.Window1	210
Emulator.Window1	210
System.Windows.Markup.InternalTypeHelper	
XamlGeneratedNamespace.GeneratedInternalTypeHelper	36
XamlGeneratedNamespace.GeneratedInternalTypeHelper	36
XamlGeneratedNamespace.GeneratedInternalTypeHelper	36
Hardware.MemoryMap	81
Emulator.Model.MemoryRowModel	87
Hardware.MemoryMap.Devices.MM65SIB	98
ObservableCollection	
Emulator.MultiThreadedObservableCollection< T >	99
Emulator.Versioning.Product	104
Hardware.Versioning.Product	106

Emulator.Model.RomFileModel	107
Emulator.SettingsFile	126
Emulator.Versioning.SettingsFile	127
Emulator.Model.SettingsModel	128
Hardware.MemoryMap.SharedRom	134
Emulator.Model.StateFileModel	135
Hardware.Utility	137
Emulator.Versioning	142
ViewModelBase	
Emulator.ViewModel.MainViewModel	47
Emulator.ViewModel.MemoryVisualViewModel	95
Emulator.ViewModel.SaveFileViewModel	115
Emulator.ViewModel.SettingsViewModel	129
Emulator.ViewModel.ViewModelLocator	142
Hardware.W65C02	144
Hardware.W65C22	185
Hardware.W65C51	195
System.Windows.Window	
Emulator.MainWindow	64
Emulator.MainWindow	64
Emulator.MainWindow	64
Emulator.MainWindow	64
Emulator.MainWindow	64
Emulator.MainWindow	64
Emulator.MemoryVisual	91
Emulator.MemoryVisual	91
Emulator.MemoryVisual	91
Emulator.MemoryVisual	91
Emulator.MemoryVisual	91
Emulator.SaveFile	109
Emulator.SaveFile	109
Emulator.SaveFile	109
Emulator.SaveFile	109

Emulator.SaveFile	109
Emulator.SaveFile	109
Emulator.Settings	119
Emulator.Settings	119
Emulator.Settings	119
Emulator.Settings	119
Emulator.Settings	119
Emulator.Settings	119
Emulator.Window1	210
Emulator.Window1	210
Window	
Emulator.MainWindow	64

3 Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Hardware.MemoryMap.Devices.ACIA	13
Emulator.App	
Interaction logic for App.xaml	14
Hardware.AT28CXX	
An implementation of a W65C02 Processor.	14
Hardware.MemoryMap.BankedRam	21
Hardware.MemoryMap.BankedRom	22
Emulator.Model.Breakpoint	
A Representation of a Breakpoint	23
Emulator.Model.BreakpointType	
The Type of Breakpoint	25
Hardware.MemoryMap.DeviceArea	26
Hardware.MemoryMap.Devices	27
Hardware.Disassembly	
Used to help simulating. This class contains the disassembly properties.	28
Emulator.ExitCodes	29
XamlGeneratedNamespace.GeneratedApplication	
GeneratedApplication	31

XamlGeneratedNamespace.GeneratedInternalTypeHelper GeneratedInternalTypeHelper	36
Hardware.MemoryMap.Devices.GPIO	42
Hardware.HM62256	42
Emulator.IClosable	46
Emulator.ViewModel.MainViewModel The Main ViewModel	47
Emulator.MainWindow Interaction logic for MainWindow.xaml	64
Hardware.MemoryMap	81
Emulator.Model.MemoryRowModel A Model of a Single Page of memory	87
Emulator.MemoryVisual Interaction logic for Window1.xaml	91
Emulator.ViewModel.MemoryVisualViewModel The Main ViewModel	95
Hardware.MemoryMap.Devices.MM65SIB	98
Emulator.MultiThreadedObservableCollection< T > A MultiThreadedObservableCollection. This allows multiple threads to access the same observable collection in a safe manner.	99
Emulator.Model.OutputLog The OutputLog Model. Used by the outputlog grid to show a history of operations performed by the CPU	101
Emulator.Versioning.Product	104
Hardware.Versioning.Product	106
Emulator.Model.RomFileModel The Model used when Loading a Program.	107
Emulator.SaveFile SaveFile	109
Emulator.ViewModel.SaveFileViewModel The ViewModel Used by the SaveFileView	115
Emulator.Settings Settings	119
Emulator.SettingsFile	126
Emulator.Versioning.SettingsFile	127
Emulator.Model.SettingsModel Model that contains the required information needed to save the current settings to disk	128
Emulator.ViewModel.SettingsViewModel The ViewModel Used by the SaveFileView	129

Hardware.MemoryMap.SharedRom	134
Emulator.Model.StateFileModel	
Model that contains the required information needed to save the current state of the processor to disk	135
Hardware.Utility	137
Emulator.Versioning	142
Emulator.ViewModel.ViewModelLocator	
This class contains static references to all the view models in the application and provides an entry point for the bindings.	142
Hardware.W65C02	
An implementation of a W65C02 Processor.	144
Hardware.W65C22	
An implementation of a W65C22 VIA.	185
Hardware.W65C51	
An implementation of a W65C51 ACIA.	195
Emulator.Window1	
Window1	210

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

Emulator/App.xaml.cs	212
Emulator/MainWindow.xaml.cs	216
Emulator/MemoryVisual.xaml.cs	217
Emulator/MultiThreadedCollection.cs	224
Emulator/SaveFile.xaml.cs	316
Emulator/Settings.xaml.cs	317
Emulator/Classes/ExitCodes.cs	213
Emulator/Classes/FileLocations.cs	213
Emulator/Classes/SettingsFile.cs	214
Emulator/Classes/Versioning.cs	214
Emulator/Interfaces/IClosable.cs	216
Emulator/Model/Breakpoint.cs	218
Emulator/Model/BreakpointType.cs	218

Emulator/Model/ MemoryRowModel.cs	219
Emulator/Model/ OutputLog.cs	220
Emulator/Model/ RomFileModel.cs	221
Emulator/Model/ SettingsModel.cs	222
Emulator/Model/ StateFileModel.cs	223
Emulator/obj/x86/Debug/ .NETFramework,Version=v4.8.AssemblyAttributes.cs	225
Emulator/obj/x86/Debug/ App.g.cs	226
Emulator/obj/x86/Debug/ App.g.i.cs	230
Emulator/obj/x86/Debug/ Emulator_Content.g.cs	234
Emulator/obj/x86/Debug/ Emulator_Content.g.i.cs	235
Emulator/obj/x86/Debug/ GeneratedInternalTypeHelper.g.cs	236
Emulator/obj/x86/Debug/ GeneratedInternalTypeHelper.g.i.cs	237
Emulator/obj/x86/Debug/ MainWindow.g.cs	240
Emulator/obj/x86/Debug/ MainWindow.g.i.cs	261
Emulator/obj/x86/Debug/ MemoryVisual.g.cs	282
Emulator/obj/x86/Debug/ MemoryVisual.g.i.cs	285
Emulator/obj/x86/Debug/ SaveFile.g.cs	289
Emulator/obj/x86/Debug/ SaveFile.g.i.cs	295
Emulator/obj/x86/Debug/ Settings.g.cs	301
Emulator/obj/x86/Debug/ Settings.g.i.cs	307
Emulator/obj/x86/Publish/ .NETFramework,Version=v4.8.AssemblyAttributes.cs	225
Emulator/obj/x86/Publish/ App.g.cs	228
Emulator/obj/x86/Publish/ App.g.i.cs	232
Emulator/obj/x86/Publish/ Emulator_Content.g.cs	235
Emulator/obj/x86/Publish/ Emulator_Content.g.i.cs	236
Emulator/obj/x86/Publish/ GeneratedInternalTypeHelper.g.cs	236
Emulator/obj/x86/Publish/ GeneratedInternalTypeHelper.g.i.cs	238
Emulator/obj/x86/Publish/ MainWindow.g.cs	247
Emulator/obj/x86/Publish/ MainWindow.g.i.cs	268
Emulator/obj/x86/Publish/ SaveFile.g.cs	291
Emulator/obj/x86/Publish/ SaveFile.g.i.cs	297
Emulator/obj/x86/Publish/ Settings.g.cs	303

Emulator/obj/x86/Publish/ Settings.g.i.cs	308
Emulator/obj/x86/Release/ .NETFramework,Version=v4.8.AssemblyAttributes.cs	226
Emulator/obj/x86/Release/ App.g.cs	229
Emulator/obj/x86/Release/ App.g.i.cs	233
Emulator/obj/x86/Release/ Emulator_Content.g.cs	235
Emulator/obj/x86/Release/ Emulator_Content.g.i.cs	236
Emulator/obj/x86/Release/ GeneratedInternalTypeHelper.g.cs	236
Emulator/obj/x86/Release/ GeneratedInternalTypeHelper.g.i.cs	239
Emulator/obj/x86/Release/ MainWindow.g.cs	254
Emulator/obj/x86/Release/ MainWindow.g.i.cs	275
Emulator/obj/x86/Release/ MemoryMap.g.i.cs	312
Emulator/obj/x86/Release/ MemoryVisual.g.cs	284
Emulator/obj/x86/Release/ MemoryVisual.g.i.cs	287
Emulator/obj/x86/Release/ SaveFile.g.cs	293
Emulator/obj/x86/Release/ SaveFile.g.i.cs	299
Emulator/obj/x86/Release/ Settings.g.cs	305
Emulator/obj/x86/Release/ Settings.g.i.cs	310
Emulator/obj/x86/Release/ Window1.g.i.cs	314
Emulator/Properties/ AssemblyInfo.cs	315
Emulator/ViewModel/ MainViewModel.cs	318
Emulator/ViewModel/ MemoryVisualViewModel.cs	328
Emulator/ViewModel/ SaveFileViewModel.cs	329
Emulator/ViewModel/ SettingsViewModel.cs	331
Emulator/ViewModel/ ViewModelLocator.cs	332
Hardware/Classes/ AddressingMode.cs	334
Hardware/Classes/ Disassembly.cs	335
Hardware/Classes/ FileLocations.cs	213
Hardware/Classes/ MemoryMap.cs	336
Hardware/Classes/ Utility.cs	339
Hardware/Classes/ Versioning.cs	215
Hardware/Hardware/ AT28CXX.cs	343
Hardware/Hardware/ HM62256.cs	345

Hardware/Hardware/ W65C02.cs	347
Hardware/Hardware/ W65C22.cs	376
Hardware/Hardware/ W65C51.cs	380
Hardware/obj/Debug/ .NETFramework,Version=v4.8.AssemblyAttributes.cs	226
Hardware/obj/Publish/ .NETFramework,Version=v4.8.AssemblyAttributes.cs	226
Hardware/obj/Release/ .NETFramework,Version=v4.8.AssemblyAttributes.cs	226
Hardware/Properties/ AssemblyInfo.cs	316

5 Namespace Documentation

5.1 Emulator Namespace Reference

Namespaces

- namespace [Model](#)
- namespace [ViewModel](#)

Classes

- class [App](#)
Interaction logic for App.xaml
- class [ExitCodes](#)
- interface [IClosable](#)
- class [MainWindow](#)
Interaction logic for MainWindow.xaml
- class [MemoryVisual](#)
Interaction logic for Window1.xaml
- class [MultiThreadedObservableCollection](#)
A MultiThreaedObservableCollection. This allows multiple threads to access the same observable collection in a safe manner.
- class [SaveFile](#)
SaveFile
- class [Settings](#)
Settings
- class [SettingsFile](#)
- class [Versioning](#)
- class [Window1](#)
Window1

5.2 Emulator.Model Namespace Reference

Classes

- class [Breakpoint](#)
A Representation of a [Breakpoint](#)
- class [BreakpointType](#)
The Type of [Breakpoint](#)
- class [MemoryRowModel](#)
A [Model](#) of a Single Page of memory
- class [OutputLog](#)
The [OutputLog Model](#). Used by the outputlog grid to show a history of operations performed by the CPU
- class [RomFileModel](#)
The [Model](#) used when Loading a Program.
- class [SettingsModel](#)
[Model](#) that contains the required information needed to save the current settings to disk
- class [StateFileModel](#)
[Model](#) that contains the required information needed to save the current state of the processor to disk

5.3 Emulator.ViewModel Namespace Reference

Classes

- class [MainViewModel](#)
The Main [ViewModel](#)
- class [MemoryVisualViewModel](#)
The Main [ViewModel](#)
- class [SaveFileViewModel](#)
The [ViewModel](#) Used by the SaveFileView
- class [SettingsViewModel](#)
The [ViewModel](#) Used by the SaveFileView
- class [ViewModelLocator](#)
This class contains static references to all the view models in the application and provides an entry point for the bindings.

5.4 Hardware Namespace Reference

Classes

- class [AT28CXX](#)
An implementation of a [W65C02](#) Processor.
- class [Disassembly](#)
Used to help simulating. This class contains the disassembly properties.
- class [HM62256](#)
- class [MemoryMap](#)
- class [Utility](#)
- class [W65C02](#)
An implementation of a [W65C02](#) Processor.
- class [W65C22](#)
An implementation of a [W65C22](#) VIA.
- class [W65C51](#)
An implementation of a [W65C51](#) ACIA.

Enumerations

- enum [AddressingMode](#)

The addressing modes used by the 6502 Processor

5.4.1 Enumeration Type Documentation

5.4.1.1 AddressingMode enum [Hardware.AddressingMode](#)

The addressing modes used by the 6502 Processor

Definition at line 6 of file [AddressingMode.cs](#).

```

00007 {
00008 /// <summary>
00009 /// In this mode a full address is given to operation on IE: Memory byte[] { 0x60, 0x00, 0xFF }
00010 /// would perform an ADC operation and Add the value at ADDRESS 0xFF00 to the accumulator.
00011 /// The address is always LSB first
00012 /// </summary>
00013     Absolute = 1,
00014 /// <summary>
00015 /// In this mode a full address is given to operation on IE: Memory byte[] { 0x7D, 0x00, 0xFF } The
00016 /// full value would then be added to the X Register.
00017 /// If the X register was 0x01 then the address would be 0xFF01. and the value stored there would
00018 /// have an ADC operation performed on it and the value would
00019 /// be added to the accumulator.
00020 /// </summary>
00021     AbsoluteX = 2,
00022 /// <summary>
00023 /// In this mode a full address is given to operation on IE: Memory byte[] { 0x79, 0x00, 0xFF } The
00024 /// full value would then be added to the Y Register.
00025 /// If the Y register was 0x01 then the address would be 0xFF01. and the value stored there would
00026 /// have an ADC operation performed on it and the value would
00027 /// be added to the accumulator
00028 /// </summary>
00029     AbsoluteY = 3,
00030 /// <summary>
00031 /// In this mode the instruction operates on the accumulator. No operands are needed.
00032 /// </summary>
00033     Accumulator = 4,
00034 /// <summary>
00035 /// In this mode, the value to operate on immediately follows the instruction. IE: Memory byte[] {
00036 /// 0x69, 0x01 }
00037 /// would perform an ADC operation and Add 0x01 directly to the accumulator
00038 /// </summary>
00039     Immediate = 5,
00040 /// <summary>
00041 /// No address is needed for this mode. EX: BRK (Break), CLC (Clear Carry Flag) etc
00042 /// </summary>
00043     Implied = 6,
00044 /// <summary>
00045 /// In this mode assume the following
00046 /// Memory = { 0x61, 0x02, 0x04, 0x00, 0x03 }
00047 /// RegisterX = 0x01
00048 /// 1. Take the sum of the X Register and the value after the opcode 0x01 + 0x01 = 0x02.
00049 /// 2. Starting at position 0x02 get an address (0x04,0x00) = 0x0004
00050 /// 3. Perform the ADC operation and Add the value at 0x0005 to the accumulator
00051 /// Note: if the Zero Page address is greater than 0xff then roll over the value. IE 0x101 rolls
00052 /// over to 0x01
00053 /// </summary>
00054     IndirectX = 7,
00055 /// <summary>
00056 /// In this mode assume the following
00057 /// Memory = { 0x61, 0x02, 0x04, 0x00, 0x03 }
00058 /// RegisterY = 0x01
00059 /// 1. Starting at position 0x02 get an address (0x04,0x00) = 0x0004
00060 /// 2. Take the sum of the Y Register and the absolute address 0x01+0x0004 = 0x0005
00061 /// 3. Perform the ADC operation and Add the value at 0x0005 to the accumulator
00062 /// Note: if the address is great that 0xffff then roll over IE: 0x10001 rolls over to 0x01
00063 /// </summary>
00064     IndirectY = 8,
00065 /// <summary>
00066 /// JMP is the only operation that uses this mode. In this mode an absolute address is specified that
00067 /// points to the location of the absolute address we want to jump to.
00068 /// </summary>
00069     IndirectY = 8,
00070 /// <summary>
00071 /// JMP is the only operation that uses this mode. In this mode an absolute address is specified that
00072 /// points to the location of the absolute address we want to jump to.
00073 /// </summary>
00074     IndirectY = 8,
00075 }

```

```

00062         Indirect = 9,
00063     /// <summary>
00064     /// This Mode Changes the PC. It allows the program to change the location of the PC by 127 in either
    direction.
00065     /// </summary>
00066         Relative = 10,
00067     /// <summary>
00068     /// In this mode, a zero page address of the value to operate on is specified. This mode can only
    operation on values between 0x0 and 0xFF, or those that sit on the zero page of memory. IE: Memory
    byte[] { 0x69, 0x02, 0x01 }
00069     /// would perform an ADC operation and Add 0x01 directly to the Accumulator
00070     /// </summary>
00071         ZeroPage = 11,
00072     /// <summary>
00073     /// In this mode, a zero page address of the value to operate on is specified, however the value of
    the X register is added to the address IE: Memory byte[] { 0x86, 0x02, 0x01, 0x67, 0x04, 0x01 }
00074     /// In this example we store a value of 0x01 into the X register, then we would perform an ADC
    operation using the address of 0x04+0x01=0x05 and Add the result of 0x01 directly to the Accumulator
00075     /// </summary>
00076         ZeroPageX = 12,
00077     /// <summary>
00078     /// This works the same as ZeroPageX except it uses the Y register instead of the X register.
00079     /// </summary>
00080         ZeroPageY = 13,
00081     }

```

5.5 XamlGeneratedNamespace Namespace Reference

Classes

- class [GeneratedApplication](#)
GeneratedApplication
- class [GeneratedInternalTypeHelper](#)
GeneratedInternalTypeHelper

6 Class Documentation

6.1 Hardware.MemoryMap.Devices.ACIA Class Reference

Static Public Attributes

- static int [Length](#) = 0x03
- static byte [Offset](#) = 0x10

6.1.1 Detailed Description

Definition at line 57 of file [MemoryMap.cs](#).

6.1.2 Member Data Documentation

6.1.2.1 Length

int Hardware.MemoryMap.Devices.ACIA.Length = 0x03 [static]

Definition at line 59 of file [MemoryMap.cs](#).

6.1.2.2 Offset `byte Hardware.MemoryMap.Devices.ACIA.Offset = 0x10 [static]`

Definition at line 60 of file [MemoryMap.cs](#).

The documentation for this class was generated from the following file:

- [Hardware/Classes/MemoryMap.cs](#)

6.2 Emulator.App Class Reference

Interaction logic for App.xaml

6.2.1 Detailed Description

Interaction logic for App.xaml

Definition at line 6 of file [App.xaml.cs](#).

The documentation for this class was generated from the following file:

- [Emulator/App.xaml.cs](#)

6.3 Hardware.AT28CXX Class Reference

An implementation of a [W65C02](#) Processor.

Public Member Functions

- [AT28CXX](#) (int offset, int length, byte banks)
Default Constructor, Instantiates a new instance of the processor.
- void [Load](#) (byte[] data)
Loads a program into ROM.
- void [Load](#) (byte bank, byte[] data)
Loads a program into ROM.
- byte[][] [ReadFile](#) (string filename)
- byte [Read](#) (int address)
Returns the byte at a given address without incrementing the cycle. Useful for test harness.
- void [Write](#) (int address, byte data)
Writes data to the given address without incrementing the cycle.
- byte[][] [DumpMemory](#) ()
Dumps the entire memory object. Used when saving the memory state
- byte[] [DumpMemory](#) (byte bank)
Dumps the selected ROM bank.
- void [Clear](#) ()
Clears the ROM.

Properties

- `byte[][] Memory` [get, private set]
The ROM.
- `byte Banks` [get, private set]
The total number of banks on the ROM.
- `byte CurrentBank` [get, private set]
The bank the ROM is currently using.
- `int Offset` [get, private set]
The memory offset
- `int End` [get]
The end of memory
- `int Length` [get, private set]
The memory length
- `W65C02 Processor` [get, private set]
The processor reference

6.3.1 Detailed Description

An implementation of a [W65C02](#) Processor.

Definition at line 10 of file [AT28CXX.cs](#).

6.3.2 Constructor & Destructor Documentation

6.3.2.1 AT28CXX() `Hardware.AT28CXX.AT28CXX (`
`int offset,`
`int length,`
`byte banks) [inline]`

Default Constructor, Instantiates a new instance of the processor.

Definition at line 54 of file [AT28CXX.cs](#).

```
00055     {
00056         Memory = new byte[banks][];
00057         for (int i = 0; i < banks; i++)
00058         {
00059             Memory[i] = new byte[length + 1];
00060         }
00061         Offset = offset;
00062         Length = length;
00063         Banks = banks;
00064         CurrentBank = 0;
00065     }
```

6.3.3 Member Function Documentation

6.3.3.1 Clear() `void Hardware.AT28CXX.Clear () [inline]`

Clears the ROM.

Definition at line 166 of file [AT28CXX.cs](#).

```
00167     {
00168         for (byte i = 0; i < Banks; i++)
00169         {
00170             for (int j = 0; j < Length; j++)
00171             {
00172                 Memory[i][j] = 0x00;
00173             }
00174         }
00175     }
```

6.3.3.2 DumpMemory() [1/2] `byte[][] Hardware.AT28CXX.DumpMemory () [inline]`

Dumps the entire memory object. Used when saving the memory state

Returns

2 dimensional array of data analogous to the ROM of the computer.

Definition at line 143 of file [AT28CXX.cs](#).

```
00144     {
00145         return Memory;
00146     }
```

6.3.3.3 DumpMemory() [2/2] `byte[] Hardware.AT28CXX.DumpMemory (byte bank) [inline]`

Dumps the selected ROM bank.

Parameters

<i>bank</i>	The bank to dump data from.
-------------	-----------------------------

Returns

Array that represents the selected ROM bank.

Definition at line 153 of file [AT28CXX.cs](#).

```
00154     {
00155         byte[] _tempMemory = new byte[MemoryMap.BankedRom.Length + 1];
00156         for (var i = 0; i < MemoryMap.BankedRom.Length; i++)
00157         {
00158             _tempMemory[i] = Memory[bank][i];
00159         }
00160         return _tempMemory;
00161     }
```

6.3.3.4 Load() [1/2] `void Hardware.AT28CXX.Load (`
 `byte bank,`
 `byte[] data) [inline]`

Loads a program into ROM.

Parameters

<i>bank</i>	The bank to load data to.
<i>data</i>	The data to be loaded to ROM.

Definition at line 84 of file [AT28CXX.cs](#).

```
00085     {
00086         for (int i = 0; i <= Length; i++)
00087         {
00088             Memory[bank][i] = data[i];
00089         }
00090     }
```

6.3.3.5 Load() [2/2] void Hardware.AT28CXX.Load (
byte data[][]) [inline]

Loads a program into ROM.

Parameters

<i>data</i>	The program to be loaded
-------------	--------------------------

Definition at line 71 of file [AT28CXX.cs](#).

```
00072     {
00073         for (byte i = 0; i < Banks; i++)
00074         {
00075             Load(i, data[i]);
00076         }
00077     }
```

6.3.3.6 Read() byte Hardware.AT28CXX.Read (
int address) [inline]

Returns the byte at a given address without incrementing the cycle. Useful for test harness.

Parameters

<i>bank</i>	The bank to read data from.
<i>address</i>	

Returns

the byte being returned

Definition at line 121 of file [AT28CXX.cs](#).

```
00122     {
00123         return Memory[CurrentBank][address - Offset];
00124     }
```

6.3.3.7 ReadFile() `byte[][] Hardware.AT28CXX.ReadFile (string filename) [inline]`

Definition at line 92 of file [AT28CXX.cs](#).

```

00093     {
00094         byte[][] bios = new byte[Banks][];
00095         try
00096         {
00097             FileStream file = new FileStream(filename, FileMode.Open, FileAccess.Read);
00098             for (int i = 0; i < Banks; i++)
00099             {
00100                 bios[i] = new byte[Length + 1];
00101                 for (int j = 0; j <= Length; j++)
00102                 {
00103                     bios[i][j] = new byte();
00104                     bios[i][j] = (byte) file.ReadByte();
00105                 }
00106             }
00107         }
00108         catch (Exception)
00109         {
00110             return null;
00111         }
00112         return bios;
00113     }

```

6.3.3.8 Write() `void Hardware.AT28CXX.Write (int address, byte data) [inline]`

Writes data to the given address without incrementing the cycle.

Parameters

<i>bank</i>	The bank to load data to.
<i>address</i>	The address to write data to
<i>data</i>	The data to write

Definition at line 132 of file [AT28CXX.cs](#).

```

00133     {
00134         _ = address;
00135         _ = data;
00136         return;
00137     }

```

6.3.4 Property Documentation

6.3.4.1 Banks `byte Hardware.AT28CXX.Banks [get], [private set]`

The total number of banks on the ROM.

Definition at line 22 of file [AT28CXX.cs](#).

```

00022 { get; private set; }

```

6.3.4.2 CurrentBank `byte Hardware.AT28CXX.CurrentBank [get], [private set]`

The bank the ROM is currently using.

Definition at line 27 of file [AT28CXX.cs](#).

```
00027 { get; private set; }
```

6.3.4.3 End `int Hardware.AT28CXX.End [get]`

The end of memory

Definition at line 37 of file [AT28CXX.cs](#).

```
00037 { get { return Offset + Length; } }
```

6.3.4.4 Length `int Hardware.AT28CXX.Length [get], [private set]`

The memory length

Definition at line 42 of file [AT28CXX.cs](#).

```
00042 { get; private set; }
```

6.3.4.5 Memory `byte [][] Hardware.AT28CXX.Memory [get], [private set]`

The ROM.

Definition at line 17 of file [AT28CXX.cs](#).

```
00017 { get; private set; }
```

6.3.4.6 Offset `int Hardware.AT28CXX.Offset [get], [private set]`

The memory offset

Definition at line 32 of file [AT28CXX.cs](#).

```
00032 { get; private set; }
```

6.3.4.7 Processor `W65C02 Hardware.AT28CXX.Processor [get], [private set]`

The processor reference

Definition at line 47 of file [AT28CXX.cs](#).

```
00047 { get; private set; }
```

The documentation for this class was generated from the following file:

- Hardware/Hardware/[AT28CXX.cs](#)

6.4 Hardware.MemoryMap.BankedRam Class Reference

Static Public Attributes

- static int [TotalLength](#) = ([BankSize](#) * [TotalBanks](#)) - 1
- static int [BankSize](#) = (int)([Length](#) + 1)
- static byte [TotalBanks](#) = 16

Properties

- static int [Offset](#) [get]
- static int [Length](#) [get]

Static Private Attributes

- static int [_Offset](#) = 0x0000
- static int [_Length](#) = 0x7FFF

6.4.1 Detailed Description

Definition at line 7 of file [MemoryMap.cs](#).

6.4.2 Member Data Documentation

6.4.2.1 [_Length](#) int Hardware.MemoryMap.BankedRam._Length = 0x7FFF [static], [private]

Definition at line 10 of file [MemoryMap.cs](#).

6.4.2.2 [_Offset](#) int Hardware.MemoryMap.BankedRam._Offset = 0x0000 [static], [private]

Definition at line 9 of file [MemoryMap.cs](#).

6.4.2.3 [BankSize](#) int Hardware.MemoryMap.BankedRam.BankSize = (int)([Length](#) + 1) [static]

Definition at line 13 of file [MemoryMap.cs](#).

6.4.2.4 TotalBanks `byte Hardware.MemoryMap.BankedRam.TotalBanks = 16 [static]`

Definition at line 14 of file [MemoryMap.cs](#).

6.4.2.5 TotalLength `int Hardware.MemoryMap.BankedRam.TotalLength = (BankSize * TotalBanks) - 1 [static]`

Definition at line 12 of file [MemoryMap.cs](#).

6.4.3 Property Documentation

6.4.3.1 Length `int Hardware.MemoryMap.BankedRam.Length [static], [get]`

Definition at line 17 of file [MemoryMap.cs](#).

```
00017 { get { return _Length; } }
```

6.4.3.2 Offset `int Hardware.MemoryMap.BankedRam.Offset [static], [get]`

Definition at line 16 of file [MemoryMap.cs](#).

```
00016 { get { return _Offset; } }
```

The documentation for this class was generated from the following file:

- [Hardware/Classes/MemoryMap.cs](#)

6.5 Hardware.MemoryMap.BankedRom Class Reference

Static Public Attributes

- static byte [TotalBanks](#) = 16

Properties

- static int [Offset](#) [get]
- static int [Length](#) [get]

Static Private Attributes

- static int [_Offset](#) = 0x8000
- static int [_Length](#) = 0x3FFF

6.5.1 Detailed Description

Definition at line 33 of file [MemoryMap.cs](#).

6.5.2 Member Data Documentation

6.5.2.1 `_Length` `int Hardware.MemoryMap.BankedRom._Length = 0x3FFF [static], [private]`

Definition at line 36 of file [MemoryMap.cs](#).

6.5.2.2 `_Offset` `int Hardware.MemoryMap.BankedRom._Offset = 0x8000 [static], [private]`

Definition at line 35 of file [MemoryMap.cs](#).

6.5.2.3 `TotalBanks` `byte Hardware.MemoryMap.BankedRom.TotalBanks = 16 [static]`

Definition at line 38 of file [MemoryMap.cs](#).

6.5.3 Property Documentation

6.5.3.1 `Length` `int Hardware.MemoryMap.BankedRom.Length [static], [get]`

Definition at line 41 of file [MemoryMap.cs](#).

```
00041 { get { return _Length; } }
```

6.5.3.2 `Offset` `int Hardware.MemoryMap.BankedRom.Offset [static], [get]`

Definition at line 40 of file [MemoryMap.cs](#).

```
00040 { get { return _Offset; } }
```

The documentation for this class was generated from the following file:

- Hardware/Classes/[MemoryMap.cs](#)

6.6 Emulator.Model.Breakpoint Class Reference

A Representation of a [Breakpoint](#)

Properties

- bool **IsEnabled** [get, set]
*Is the **Breakpoint** enabled or disabled*
- string **Value** [get, set]
*The Value of the **Breakpoint***
- string **Type** [get, set]
The Type of breakpoint being set
- List< string > **AllTypes** [get]

6.6.1 Detailed Description

A Representation of a **Breakpoint**

Definition at line 8 of file **Breakpoint.cs**.

6.6.2 Property Documentation

6.6.2.1 AllTypes List<string> Emulator.Model.Breakpoint.AllTypes [get]

Definition at line 25 of file **Breakpoint.cs**.

```
00026     {  
00027         get { return BreakpointType.AllTypes; }  
00028     }
```

6.6.2.2 IsEnabled bool Emulator.Model.Breakpoint.IsEnabled [get], [set]

Is the **Breakpoint** enabled or disabled

Definition at line 13 of file **Breakpoint.cs**.

```
00013 { get; set; }
```

6.6.2.3 Type string Emulator.Model.Breakpoint.Type [get], [set]

The Type of breakpoint being set

Definition at line 23 of file **Breakpoint.cs**.

```
00023 { get; set; }
```

6.6.2.4 Value `string Emulator.Model.Breakpoint.Value [get], [set]`

The Value of the [Breakpoint](#)

Definition at line 18 of file [Breakpoint.cs](#).

```
00018 { get; set; }
```

The documentation for this class was generated from the following file:

- [Emulator/Model/Breakpoint.cs](#)

6.7 Emulator.Model.BreakpointType Class Reference

The Type of [Breakpoint](#)

Static Public Attributes

- static `List< string > AllTypes`
A Listing of all of the Current Types
- const string `ProgramCounterType` = "Program Counter"
The ProgramCounter Breakpoint Type
- const string `NumberOfCycleType` = "Number of Cycles"
The CycleCount Breakpoint Type

6.7.1 Detailed Description

The Type of [Breakpoint](#)

Definition at line 8 of file [BreakpointType.cs](#).

6.7.2 Member Data Documentation

6.7.2.1 AllTypes `List<string> Emulator.Model.BreakpointType.AllTypes [static]`

Initial value:

```
= new List<string>
{
    ProgramCounterType,
    NumberOfCycleType
}
```

A Listing of all of the Current Types

Definition at line 13 of file [BreakpointType.cs](#).

6.7.2.2 NumberOfCycleType `const string Emulator.Model.BreakpointType.NumberOfCycleType = "Number of Cycles" [static]`

The CycleCount [Breakpoint](#) Type

Definition at line [27](#) of file [BreakpointType.cs](#).

6.7.2.3 ProgramCounterType `const string Emulator.Model.BreakpointType.ProgramCounterType = "Program Counter" [static]`

The ProgramCounter [Breakpoint](#) Type

Definition at line [22](#) of file [BreakpointType.cs](#).

The documentation for this class was generated from the following file:

- Emulator/Model/[BreakpointType.cs](#)

6.8 Hardware.MemoryMap.DeviceArea Class Reference

Properties

- static int [End](#) [get]
The end of memory
- static int [Offset](#) [get]
- static int [Length](#) [get]

Static Private Attributes

- static int [_Offset](#) = 0xD000
- static int [_Length](#) = 0x00FF

6.8.1 Detailed Description

Definition at line [20](#) of file [MemoryMap.cs](#).

6.8.2 Member Data Documentation

6.8.2.1 _Length `int Hardware.MemoryMap.DeviceArea._Length = 0x00FF [static], [private]`

Definition at line [23](#) of file [MemoryMap.cs](#).

6.8.2.2 `_Offset` `int Hardware.MemoryMap.DeviceArea._Offset = 0xD000 [static], [private]`

Definition at line 22 of file [MemoryMap.cs](#).

6.8.3 Property Documentation

6.8.3.1 `End` `int Hardware.MemoryMap.DeviceArea.End [static], [get]`

The end of memory

Definition at line 28 of file [MemoryMap.cs](#).

```
00028 { get { return Offset + Length; } }
```

6.8.3.2 `Length` `int Hardware.MemoryMap.DeviceArea.Length [static], [get]`

Definition at line 30 of file [MemoryMap.cs](#).

```
00030 { get { return _Length; } }
```

6.8.3.3 `Offset` `int Hardware.MemoryMap.DeviceArea.Offset [static], [get]`

Definition at line 29 of file [MemoryMap.cs](#).

```
00029 { get { return _Offset; } }
```

The documentation for this class was generated from the following file:

- [Hardware/Classes/MemoryMap.cs](#)

6.9 Hardware.MemoryMap.Devices Class Reference

Classes

- class [ACIA](#)
- class [GPIO](#)
- class [MM65SIB](#)

6.9.1 Detailed Description

Definition at line 55 of file [MemoryMap.cs](#).

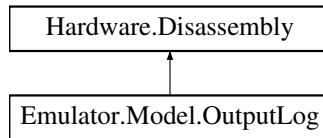
The documentation for this class was generated from the following file:

- [Hardware/Classes/MemoryMap.cs](#)

6.10 Hardware.Disassembly Class Reference

Used to help simulating. This class contains the disassembly properties.

Inheritance diagram for Hardware.Disassembly:



Properties

- string [LowAddress](#) [get, set]
The low Address
- string [HighAddress](#) [get, set]
The High Address
- string [OpCodeString](#) [get, set]
The string representation of the OpCode
- string [DisassemblyOutput](#) [get, set]
The disassembly of the current step

6.10.1 Detailed Description

Used to help simulating. This class contains the disassembly properties.

Definition at line 9 of file [Disassembly.cs](#).

6.10.2 Property Documentation

6.10.2.1 DisassemblyOutput `string Hardware.Disassembly.DisassemblyOutput [get], [set]`

The disassembly of the current step

Definition at line 29 of file [Disassembly.cs](#).

```
00029 { get; set; }
```

6.10.2.2 HighAddress `string Hardware.Disassembly.HighAddress [get], [set]`

The High Address

Definition at line 19 of file [Disassembly.cs](#).

```
00019 { get; set; }
```

6.10.2.3 LowAddress `string Hardware.Disassembly.LowAddress [get], [set]`

The low Address

Definition at line 14 of file [Disassembly.cs](#).

```
00014 { get; set; }
```

6.10.2.4 OpCodeString `string Hardware.Disassembly.OpCodeString [get], [set]`

The string representation of the OpCode

Definition at line 24 of file [Disassembly.cs](#).

```
00024 { get; set; }
```

The documentation for this class was generated from the following file:

- [Hardware/Classes/Disassembly.cs](#)

6.11 Emulator.ExitCodes Class Reference

Static Public Attributes

- static readonly int [NO_ERROR](#) = 0x00
- static readonly int [USER_ERROR](#) = 0x01
- static readonly int [NO_BIOS](#) = 0x02
- static readonly int [LOAD_BIOS_FILE_ERROR](#) = 0x03
- static readonly int [BIOS_LOADPROGRAM_ERROR](#) = 0x04
- static readonly int [LOAD_ROM_FILE_ERROR](#) = 0x05
- static readonly int [ROM_LOADPROGRAM_ERROR](#) = 0x06
- static readonly int [LOAD_STATE_ERROR](#) = 0x07

6.11.1 Detailed Description

Definition at line 3 of file [ExitCodes.cs](#).

6.11.2 Member Data Documentation

6.11.2.1 BIOS_LOADPROGRAM_ERROR `readonly int Emulator.ExitCodes.BIOS_LOADPROGRAM_ERROR = 0x04 [static]`

Definition at line 11 of file [ExitCodes.cs](#).

6.11.2.2 LOAD_BIOS_FILE_ERROR readonly int Emulator.ExitCodes.LOAD_BIOS_FILE_ERROR = 0x03
[static]

Definition at line 10 of file [ExitCodes.cs](#).

6.11.2.3 LOAD_ROM_FILE_ERROR readonly int Emulator.ExitCodes.LOAD_ROM_FILE_ERROR = 0x05
[static]

Definition at line 12 of file [ExitCodes.cs](#).

6.11.2.4 LOAD_STATE_ERROR readonly int Emulator.ExitCodes.LOAD_STATE_ERROR = 0x07 [static]

Definition at line 14 of file [ExitCodes.cs](#).

6.11.2.5 NO_BIOS readonly int Emulator.ExitCodes.NO_BIOS = 0x02 [static]

Definition at line 9 of file [ExitCodes.cs](#).

6.11.2.6 NO_ERROR readonly int Emulator.ExitCodes.NO_ERROR = 0x00 [static]

Definition at line 5 of file [ExitCodes.cs](#).

6.11.2.7 ROM_LOADPROGRAM_ERROR readonly int Emulator.ExitCodes.ROM_LOADPROGRAM_ERROR = 0x06 [static]

Definition at line 13 of file [ExitCodes.cs](#).

6.11.2.8 USER_ERROR readonly int Emulator.ExitCodes.USER_ERROR = 0x01 [static]

Definition at line 7 of file [ExitCodes.cs](#).

The documentation for this class was generated from the following file:

- Emulator/Classes/[ExitCodes.cs](#)

6.12 XamlGeneratedNamespace.GeneratedApplication Class Reference

GeneratedApplication

Inheritance diagram for XamlGeneratedNamespace.GeneratedApplication:



Public Member Functions

- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent

Static Public Member Functions

- static void [Main](#) ()
Application Entry Point.
- static void [Main](#) ()
Application Entry Point.
- static void [Main](#) ()
Application Entry Point.
- static void [Main](#) ()
Application Entry Point.
- static void [Main](#) ()
Application Entry Point.
- static void [Main](#) ()
Application Entry Point.

Private Attributes

- bool [_contentLoaded](#)

6.12.1 Detailed Description

GeneratedApplication

Definition at line 41 of file [App.g.cs](#).

6.12.2 Member Function Documentation

6.12.2.1 InitializeComponent() [1/6] void XamlGeneratedNamespace.GeneratedApplication.InitializeComponent () [inline]

InitializeComponent

Definition at line 50 of file [App.g.cs](#).

```
00050                                     {
00051             if (_contentLoaded) {
00052                 return;
00053             }
00054             _contentLoaded = true;
00055
00056 #line 2 "..\..\..\App.xaml"
00057             this.StartupUri = new System.Uri("MainWindow.xaml", System.UriKind.Relative);
00058
00059 #line default
00060 #line hidden
00061             System.Uri resourceLocator = new System.Uri("/Emulator;component/app.xaml",
00062                 System.UriKind.Relative);
00063 #line 1 "..\..\..\App.xaml"
00064             System.Windows.Application.LoadComponent(this, resourceLocator);
00065
00066 #line default
00067 #line hidden
00068         }
```

6.12.2.2 InitializeComponent() [2/6] void XamlGeneratedNamespace.GeneratedApplication.InitializeComponent () [inline]

InitializeComponent

Definition at line 50 of file [App.g.i.cs](#).

```
00050                                     {
00051             if (_contentLoaded) {
00052                 return;
00053             }
00054             _contentLoaded = true;
00055
00056 #line 2 "..\..\..\App.xaml"
00057             this.StartupUri = new System.Uri("MainWindow.xaml", System.UriKind.Relative);
00058
00059 #line default
00060 #line hidden
00061             System.Uri resourceLocator = new System.Uri("/Emulator;component/app.xaml",
00062                 System.UriKind.Relative);
00063 #line 1 "..\..\..\App.xaml"
00064             System.Windows.Application.LoadComponent(this, resourceLocator);
00065
00066 #line default
00067 #line hidden
00068         }
```

6.12.2.3 InitializeComponent() [3/6] void XamlGeneratedNamespace.GeneratedApplication.InitializeComponent () [inline]

InitializeComponent

Definition at line 50 of file [App.g.cs](#).

```
00050 {
00051     if (_contentLoaded) {
00052         return;
00053     }
00054     _contentLoaded = true;
00055
00056 #line 2 "..\..\..\App.xaml"
00057     this.StartupUri = new System.Uri("MainWindow.xaml", System.UriKind.Relative);
00058
00059 #line default
00060 #line hidden
00061     System.Uri resourceLocator = new System.Uri("/Emulator;component/app.xaml",
00062         System.UriKind.Relative);
00063 #line 1 "..\..\..\App.xaml"
00064     System.Windows.Application.LoadComponent(this, resourceLocator);
00065
00066 #line default
00067 #line hidden
00068 }
```

6.12.2.4 InitializeComponent() [4/6] void XamlGeneratedNamespace.GeneratedApplication.InitializeComponent () [inline]

InitializeComponent

Definition at line 50 of file [App.g.i.cs](#).

```
00050 {
00051     if (_contentLoaded) {
00052         return;
00053     }
00054     _contentLoaded = true;
00055
00056 #line 2 "..\..\..\App.xaml"
00057     this.StartupUri = new System.Uri("MainWindow.xaml", System.UriKind.Relative);
00058
00059 #line default
00060 #line hidden
00061     System.Uri resourceLocator = new System.Uri("/Emulator;component/app.xaml",
00062         System.UriKind.Relative);
00063 #line 1 "..\..\..\App.xaml"
00064     System.Windows.Application.LoadComponent(this, resourceLocator);
00065
00066 #line default
00067 #line hidden
00068 }
```

6.12.2.5 InitializeComponent() [5/6] void XamlGeneratedNamespace.GeneratedApplication.InitializeComponent () [inline]

InitializeComponent

Definition at line 50 of file [App.g.cs](#).

```
00050 {
00051     if (_contentLoaded) {
00052         return;
00053     }
00054     _contentLoaded = true;
00055
00056 #line 2 "..\..\..\App.xaml"
00057     this.StartupUri = new System.Uri("MainWindow.xaml", System.UriKind.Relative);
00058 }
```

```

00059 #line default
00060 #line hidden
00061         System.Uri resourceLocator = new System.Uri("/Emulator;component/app.xaml",
            System.UriKind.Relative);
00062
00063 #line 1 "..\..\..\App.xaml"
00064         System.Windows.Application.LoadComponent(this, resourceLocator);
00065
00066 #line default
00067 #line hidden
00068     }

```

6.12.2.6 InitializeComponent() [6/6] void XamlGeneratedNamespace.GeneratedApplication.InitializeComponent () [inline]

InitializeComponent

Definition at line 50 of file [App.g.i.cs](#).

```

00050                                     {
00051         if (_contentLoaded) {
00052             return;
00053         }
00054         _contentLoaded = true;
00055
00056 #line 2 "..\..\..\App.xaml"
00057         this.StartupUri = new System.Uri("MainWindow.xaml", System.UriKind.Relative);
00058
00059 #line default
00060 #line hidden
00061         System.Uri resourceLocator = new System.Uri("/Emulator;component/app.xaml",
            System.UriKind.Relative);
00062
00063 #line 1 "..\..\..\App.xaml"
00064         System.Windows.Application.LoadComponent(this, resourceLocator);
00065
00066 #line default
00067 #line hidden
00068     }

```

6.12.2.7 Main() [1/6] static void XamlGeneratedNamespace.GeneratedApplication.Main () [inline], [static]

Application Entry Point.

Definition at line 76 of file [App.g.cs](#).

```

00076                                     {
00077         SplashScreen splashScreen = new SplashScreen("splashscreen.png");
00078         splashScreen.Show(true);
00079         XamlGeneratedNamespace.GeneratedApplication app = new
            XamlGeneratedNamespace.GeneratedApplication();
00080         app.InitializeComponent();
00081         app.Run();
00082     }

```

6.12.2.8 Main() [2/6] static void XamlGeneratedNamespace.GeneratedApplication.Main () [inline], [static]

Application Entry Point.

Definition at line 76 of file [App.g.i.cs](#).

```

00076                                     {
00077         SplashScreen splashScreen = new SplashScreen("splashscreen.png");
00078         splashScreen.Show(true);
00079         XamlGeneratedNamespace.GeneratedApplication app = new
            XamlGeneratedNamespace.GeneratedApplication();
00080         app.InitializeComponent();
00081         app.Run();
00082     }

```

6.12.2.9 Main() [3/6] `static void XamlGeneratedNamespace.GeneratedApplication.Main () [inline], [static]`

Application Entry Point.

Definition at line 76 of file [App.g.cs](#).

```
00076          {
00077              SplashScreen splashScreen = new SplashScreen("splashscreen.png");
00078              splashScreen.Show(true);
00079              XamlGeneratedNamespace.GeneratedApplication app = new
XamlGeneratedNamespace.GeneratedApplication();
00080              app.InitializeComponent();
00081              app.Run();
00082          }
```

6.12.2.10 Main() [4/6] `static void XamlGeneratedNamespace.GeneratedApplication.Main () [inline], [static]`

Application Entry Point.

Definition at line 76 of file [App.g.i.cs](#).

```
00076          {
00077              SplashScreen splashScreen = new SplashScreen("splashscreen.png");
00078              splashScreen.Show(true);
00079              XamlGeneratedNamespace.GeneratedApplication app = new
XamlGeneratedNamespace.GeneratedApplication();
00080              app.InitializeComponent();
00081              app.Run();
00082          }
```

6.12.2.11 Main() [5/6] `static void XamlGeneratedNamespace.GeneratedApplication.Main () [inline], [static]`

Application Entry Point.

Definition at line 76 of file [App.g.cs](#).

```
00076          {
00077              SplashScreen splashScreen = new SplashScreen("splashscreen.png");
00078              splashScreen.Show(true);
00079              XamlGeneratedNamespace.GeneratedApplication app = new
XamlGeneratedNamespace.GeneratedApplication();
00080              app.InitializeComponent();
00081              app.Run();
00082          }
```

6.12.2.12 Main() [6/6] `static void XamlGeneratedNamespace.GeneratedApplication.Main () [inline], [static]`

Application Entry Point.

Definition at line 76 of file [App.g.i.cs](#).

```
00076          {
00077              SplashScreen splashScreen = new SplashScreen("splashscreen.png");
00078              splashScreen.Show(true);
00079              XamlGeneratedNamespace.GeneratedApplication app = new
XamlGeneratedNamespace.GeneratedApplication();
00080              app.InitializeComponent();
00081              app.Run();
00082          }
```

6.12.3 Member Data Documentation

6.12.3.1 `_contentLoaded` `bool XamlGeneratedNamespace.GeneratedApplication._contentLoaded [private]`

Definition at line 43 of file [App.g.cs](#).

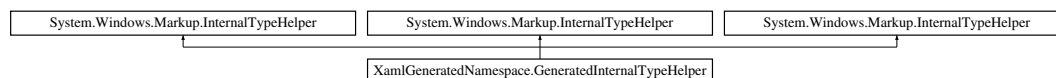
The documentation for this class was generated from the following files:

- Emulator/obj/x86/Debug/[App.g.cs](#)
- Emulator/obj/x86/Debug/[App.g.i.cs](#)
- Emulator/obj/x86/Publish/[App.g.cs](#)
- Emulator/obj/x86/Publish/[App.g.i.cs](#)
- Emulator/obj/x86/Release/[App.g.cs](#)
- Emulator/obj/x86/Release/[App.g.i.cs](#)

6.13 XamlGeneratedNamespace.GeneratedInternalTypeHelper Class Reference

[GeneratedInternalTypeHelper](#)

Inheritance diagram for XamlGeneratedNamespace.GeneratedInternalTypeHelper:



Protected Member Functions

- override object [CreateInstance](#) (System.Type type, System.Globalization.CultureInfo culture)
CreateInstance
- override object [GetPropertyValue](#) (System.Reflection.PropertyInfo propertyInfo, object target, System.Globalization.CultureInfo culture)
GetPropertyValue
- override void [SetPropertyValue](#) (System.Reflection.PropertyInfo propertyInfo, object target, object value, System.Globalization.CultureInfo culture)
SetPropertyValue
- override System.Delegate [CreateDelegate](#) (System.Type delegateType, object target, string handler)
CreateDelegate
- override void [AddEventHandler](#) (System.Reflection.EventInfo eventInfo, object target, System.Delegate handler)
AddEventHandler
- override object [CreateInstance](#) (System.Type type, System.Globalization.CultureInfo culture)
CreateInstance
- override object [GetPropertyValue](#) (System.Reflection.PropertyInfo propertyInfo, object target, System.Globalization.CultureInfo culture)
GetPropertyValue
- override void [SetPropertyValue](#) (System.Reflection.PropertyInfo propertyInfo, object target, object value, System.Globalization.CultureInfo culture)
SetPropertyValue

- override System.Delegate [CreateDelegate](#) (System.Type delegateType, object target, string handler)
CreateDelegate
- override void [AddEventHandler](#) (System.Reflection.EventInfo eventInfo, object target, System.Delegate handler)
AddEventHandler
- override object [CreateInstance](#) (System.Type type, System.Globalization.CultureInfo culture)
CreateInstance
- override object [GetPropertyValue](#) (System.Reflection.PropertyInfo propertyInfo, object target, System.Globalization.CultureInfo culture)
GetPropertyValue
- override void [SetPropertyValue](#) (System.Reflection.PropertyInfo propertyInfo, object target, object value, System.Globalization.CultureInfo culture)
SetPropertyValue
- override System.Delegate [CreateDelegate](#) (System.Type delegateType, object target, string handler)
CreateDelegate
- override void [AddEventHandler](#) (System.Reflection.EventInfo eventInfo, object target, System.Delegate handler)
AddEventHandler

6.13.1 Detailed Description

[GeneratedInternalTypeHelper](#)

Definition at line 20 of file [GeneratedInternalTypeHelper.g.i.cs](#).

6.13.2 Member Function Documentation

6.13.2.1 AddEventHandler() [1/3] override void XamlGeneratedNamespace.GeneratedInternalTypeHelper.AddEventHandler (
 System.Reflection.EventInfo eventInfo,
 object target,
 System.Delegate handler) [inline], [protected]

AddEventHandler

Definition at line 57 of file [GeneratedInternalTypeHelper.g.i.cs](#).

```
00057 {
00058     eventInfo.AddEventHandler(target, handler);
00059 }
```

6.13.2.2 AddEventHandler() [2/3] override void XamlGeneratedNamespace.GeneratedInternalTypeHelper.AddEventHandler (
 System.Reflection.EventInfo eventInfo,
 object target,
 System.Delegate handler) [inline], [protected]

AddEventHandler

Definition at line 57 of file [GeneratedInternalTypeHelper.g.i.cs](#).

```
00057 {
00058     eventInfo.AddEventHandler(target, handler);
00059 }
```

6.13.2.3 AddEventHandler() [3/3] override void XamlGeneratedNamespace.GeneratedInternalTypeHelper.AddEventHandler (
 System.Reflection.EventInfo eventInfo,
 object target,
 System.Delegate handler) [inline], [protected]

AddEventHandler

Definition at line 57 of file [GeneratedInternalTypeHelper.g.i.cs](#).

```
00057 {
00058     eventInfo.AddEventHandler(target, handler);
00059 }
```

6.13.2.4 CreateDelegate() [1/3] override System.Delegate XamlGeneratedNamespace.GeneratedInternalTypeHelper.CreateDelegate (
 System.Type delegateType,
 object target,
 string handler) [inline], [protected]

CreateDelegate

Definition at line 47 of file [GeneratedInternalTypeHelper.g.i.cs](#).

```
00047 {
00048     return ((System.Delegate) (target.GetType().InvokeMember("_CreateDelegate",
00049 (System.Reflection.BindingFlags.InvokeMethod
00049     | (System.Reflection.BindingFlags.NonPublic |
00050 System.Reflection.BindingFlags.Instance)), null, target, new object[] {
00050         delegateType,
00051         handler}, null)));
00052 }
```

6.13.2.5 CreateDelegate() [2/3] override System.Delegate XamlGeneratedNamespace.GeneratedInternalTypeHelper.CreateDelegate (
 System.Type delegateType,
 object target,
 string handler) [inline], [protected]

CreateDelegate

Definition at line 47 of file [GeneratedInternalTypeHelper.g.i.cs](#).

```
00047 {
00048     return ((System.Delegate) (target.GetType().InvokeMember("_CreateDelegate",
00049 (System.Reflection.BindingFlags.InvokeMethod
00049     | (System.Reflection.BindingFlags.NonPublic |
00050 System.Reflection.BindingFlags.Instance)), null, target, new object[] {
00050         delegateType,
00051         handler}, null)));
00052 }
```

6.13.2.6 CreateDelegate() [3/3] override System.Delegate XamlGeneratedNamespace.GeneratedInternalTypeHelper.CreateDelegate (
 System.Type delegateType,
 object target,
 string handler) [inline], [protected]

CreateDelegate

Definition at line 47 of file [GeneratedInternalTypeHelper.g.i.cs](#).

```
00047 {
00048     return ((System.Delegate) (target.GetType().InvokeMember("_CreateDelegate",
00049         (System.Reflection.BindingFlags.InvokeMethod
00049         | (System.Reflection.BindingFlags.NonPublic |
00050         System.Reflection.BindingFlags.Instance)), null, target, new object[] {
00050         delegateType,
00051         handler}, null)));
00052 }
```

6.13.2.7 CreateInstance() [1/3] override object XamlGeneratedNamespace.GeneratedInternalTypeHelper.CreateInstance (
 System.Type type,
 System.Globalization.CultureInfo culture) [inline], [protected]

CreateInstance

Definition at line 25 of file [GeneratedInternalTypeHelper.g.i.cs](#).

```
00025 {
00026     return System.Activator.CreateInstance(type, ((System.Reflection.BindingFlags.Public |
00027         System.Reflection.BindingFlags.NonPublic)
00027         | (System.Reflection.BindingFlags.Instance |
00028         System.Reflection.BindingFlags.CreateInstance)), null, null, culture);
00028 }
```

6.13.2.8 CreateInstance() [2/3] override object XamlGeneratedNamespace.GeneratedInternalTypeHelper.CreateInstance (
 System.Type type,
 System.Globalization.CultureInfo culture) [inline], [protected]

CreateInstance

Definition at line 25 of file [GeneratedInternalTypeHelper.g.i.cs](#).

```
00025 {
00026     return System.Activator.CreateInstance(type, ((System.Reflection.BindingFlags.Public |
00027         System.Reflection.BindingFlags.NonPublic)
00027         | (System.Reflection.BindingFlags.Instance |
00028         System.Reflection.BindingFlags.CreateInstance)), null, null, culture);
00028 }
```


6.13.2.9 CreateInstance() [3/3] override object XamlGeneratedNamespace.GeneratedInternalType↔
 Helper.CreateInstance (
 System.Type type,
 System.Globalization.CultureInfo culture) [inline], [protected]

CreateInstance

Definition at line 25 of file [GeneratedInternalTypeHelper.g.i.cs](#).

```
00025 {
00026     return System.Activator.CreateInstance(type, ((System.Reflection.BindingFlags.Public |
System.Reflection.BindingFlags.NonPublic)
00027         | (System.Reflection.BindingFlags.Instance |
System.Reflection.BindingFlags.CreateInstance)), null, null, culture);
00028 }
```

6.13.2.10 GetPropertyValue() [1/3] override object XamlGeneratedNamespace.GeneratedInternal↔
 TypeHelper.GetPropertyValue (
 System.Reflection.PropertyInfo propertyInfo,
 object target,
 System.Globalization.CultureInfo culture) [inline], [protected]

GetPropertyValue

Definition at line 33 of file [GeneratedInternalTypeHelper.g.i.cs](#).

```
00033 {
00034     return propertyInfo.GetValue(target, System.Reflection.BindingFlags.Default, null, null,
culture);
00035 }
```

6.13.2.11 GetPropertyValue() [2/3] override object XamlGeneratedNamespace.GeneratedInternal↔
 TypeHelper.GetPropertyValue (
 System.Reflection.PropertyInfo propertyInfo,
 object target,
 System.Globalization.CultureInfo culture) [inline], [protected]

GetPropertyValue

Definition at line 33 of file [GeneratedInternalTypeHelper.g.i.cs](#).

```
00033 {
00034     return propertyInfo.GetValue(target, System.Reflection.BindingFlags.Default, null, null,
culture);
00035 }
```

6.13.2.12 GetPropertyValue() [3/3] override object XamlGeneratedNamespace.GeneratedInternal↔
 TypeHelper.GetPropertyValue (
 System.Reflection.PropertyInfo propertyInfo,
 object target,
 System.Globalization.CultureInfo culture) [inline], [protected]

GetPropertyValue

Definition at line 33 of file [GeneratedInternalTypeHelper.g.i.cs](#).

```
00033 {
00034     return propertyInfo.GetValue(target, System.Reflection.BindingFlags.Default, null, null,
culture);
00035 }
```

6.13.2.13 SetPropertyValue() [1/3] override void XamlGeneratedNamespace.GeneratedInternalTypeHelper.SetPropertyValue (
 System.Reflection.PropertyInfo *propertyInfo*,
 object *target*,
 object *value*,
 System.Globalization.CultureInfo *culture*) [inline], [protected]

SetPropertyValue

Definition at line 40 of file [GeneratedInternalTypeHelper.g.i.cs](#).

```
00040 {  
00041     propertyInfo.SetValue(target, value, System.Reflection.BindingFlags.Default, null, null,  
00042     culture);  
}
```

6.13.2.14 SetPropertyValue() [2/3] override void XamlGeneratedNamespace.GeneratedInternalTypeHelper.SetPropertyValue (
 System.Reflection.PropertyInfo *propertyInfo*,
 object *target*,
 object *value*,
 System.Globalization.CultureInfo *culture*) [inline], [protected]

SetPropertyValue

Definition at line 40 of file [GeneratedInternalTypeHelper.g.i.cs](#).

```
00040 {  
00041     propertyInfo.SetValue(target, value, System.Reflection.BindingFlags.Default, null, null,  
00042     culture);  
}
```

6.13.2.15 SetPropertyValue() [3/3] override void XamlGeneratedNamespace.GeneratedInternalTypeHelper.SetPropertyValue (
 System.Reflection.PropertyInfo *propertyInfo*,
 object *target*,
 object *value*,
 System.Globalization.CultureInfo *culture*) [inline], [protected]

SetPropertyValue

Definition at line 40 of file [GeneratedInternalTypeHelper.g.i.cs](#).

```
00040 {  
00041     propertyInfo.SetValue(target, value, System.Reflection.BindingFlags.Default, null, null,  
00042     culture);  
}
```

The documentation for this class was generated from the following files:

- Emulator/obj/x86/Debug/[GeneratedInternalTypeHelper.g.i.cs](#)
- Emulator/obj/x86/Publish/[GeneratedInternalTypeHelper.g.i.cs](#)
- Emulator/obj/x86/Release/[GeneratedInternalTypeHelper.g.i.cs](#)

6.14 Hardware.MemoryMap.Devices.GPIO Class Reference

Static Public Attributes

- static int [Length](#) = 0x0F
- static byte [Offset](#) = 0x20

6.14.1 Detailed Description

Definition at line 63 of file [MemoryMap.cs](#).

6.14.2 Member Data Documentation

6.14.2.1 Length `int Hardware.MemoryMap.Devices.GPIO.Length = 0x0F [static]`

Definition at line 65 of file [MemoryMap.cs](#).

6.14.2.2 Offset `byte Hardware.MemoryMap.Devices.GPIO.Offset = 0x20 [static]`

Definition at line 66 of file [MemoryMap.cs](#).

The documentation for this class was generated from the following file:

- Hardware/Classes/[MemoryMap.cs](#)

6.15 Hardware.HM62256 Class Reference

Public Member Functions

- [HM62256](#) (byte banks, int offset, int length)
Called whenever a new 62256 object is required.
- void [Reset](#) ()
Called whenever the emulated computer is reset.
- void [Clear](#) ()
Clears the memory.
- byte [Read](#) (int address)
Returns the byte at a given address without incrementing the cycle. Useful for test harness.
- void [Write](#) (int address, byte data)
Writes data to the given address without incrementing the cycle.
- byte[][] [DumpMemory](#) ()
Dumps the entire memory object. Used when saving the memory state

Properties

- `byte[][]` [Memory](#) [get, set]
The memory area.
- `int` [Offset](#) [get, set]
The memory offset.
- `int` [Length](#) [get, set]
The memory length.
- `int` [End](#) [get]
The location of the end of memory.
- `byte` [Banks](#) [get, set]
The number of banks the memory has.
- `byte` [CurrentBank](#) [get, set]
The currently selected bank.

6.15.1 Detailed Description

Definition at line 3 of file [HM62256.cs](#).

6.15.2 Constructor & Destructor Documentation

6.15.2.1 HM62256() `Hardware.HM62256.HM62256 (`
`byte banks,`
`int offset,`
`int length) [inline]`

Called whenever a new 62256 object is required.

Parameters

<i>banks</i>	Number of banks the new memory will have.
<i>offset</i>	Offset of the new memory in the address space.
<i>length</i>	Length of each bank of memory.

Definition at line 41 of file [HM62256.cs](#).

```

00042     {
00043         Memory = new byte[banks][];
00044         for (int i = 0; i < banks; i++)
00045         {
00046             Memory[i] = new byte[length + 1];
00047         }
00048         Length = length;
00049         Banks = banks;
00050         Offset = offset;
00051         CurrentBank = 0;
00052     }

```

6.15.3 Member Function Documentation

6.15.3.1 Clear() `void Hardware.HM62256.Clear () [inline]`

Clears the memory.

Definition at line 65 of file [HM62256.cs](#).

```
00066      {
00067          for (var i = 0; i < Banks; i++)
00068          {
00069              for (var j = 0; j < Memory.Length; j++)
00070              {
00071                  Memory[i][j] = 0x00;
00072              }
00073          }
00074      }
```

6.15.3.2 DumpMemory() `byte[][] Hardware.HM62256.DumpMemory () [inline]`

Dumps the entire memory object. Used when saving the memory state

Returns

Jagged array representing the banked memory.

Definition at line 102 of file [HM62256.cs](#).

```
00103      {
00104          return Memory;
00105      }
```

6.15.3.3 Read() `byte Hardware.HM62256.Read (int address) [inline]`

Returns the byte at a given address without incrementing the cycle. Useful for test harness.

Parameters

<i>bank</i>	The bank to read data from.
<i>address</i>	

Returns

The byte being read.

Definition at line 82 of file [HM62256.cs](#).

```
00083      {
00084          return Memory[CurrentBank][address - Offset];
00085      }
```

6.15.3.4 Reset() `void Hardware.HM62256.Reset () [inline]`

Called whenever the emulated computer is reset.

Definition at line 57 of file [HM62256.cs](#).

```
00058     {
00059         Clear();
00060     }
```

6.15.3.5 Write() void Hardware.HM62256.Write (
int address,
byte data) [inline]

Writes data to the given address without incrementing the cycle.

Parameters

<i>bank</i>	The bank to load data to.
<i>address</i>	The address to write data to
<i>data</i>	The data to write

Definition at line 93 of file [HM62256.cs](#).

```
00094     {
00095         Memory[CurrentBank][address - Offset] = data;
00096     }
```

6.15.4 Property Documentation

6.15.4.1 Banks byte Hardware.HM62256.Banks [get], [set]

The number of banks the memory has.

Definition at line 28 of file [HM62256.cs](#).

```
00028 { get; set; }
```

6.15.4.2 CurrentBank byte Hardware.HM62256.CurrentBank [get], [set]

The currently selected bank.

Definition at line 33 of file [HM62256.cs](#).

```
00033 { get; set; }
```

6.15.4.3 End int Hardware.HM62256.End [get]

The location of the end of memory.

Definition at line 23 of file [HM62256.cs](#).

```
00023 { get { return Offset + Length; } }
```

6.15.4.4 Length `int Hardware.HM62256.Length [get], [set]`

The memory length.

Definition at line 18 of file [HM62256.cs](#).

```
00018 { get; set; }
```

6.15.4.5 Memory `byte [][] Hardware.HM62256.Memory [get], [set]`

The memory area.

Definition at line 8 of file [HM62256.cs](#).

```
00008 { get; set; }
```

6.15.4.6 Offset `int Hardware.HM62256.Offset [get], [set]`

The memory offset.

Definition at line 13 of file [HM62256.cs](#).

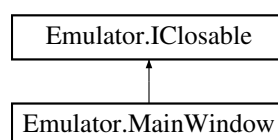
```
00013 { get; set; }
```

The documentation for this class was generated from the following file:

- Hardware/Hardware/[HM62256.cs](#)

6.16 Emulator.IClosable Interface Reference

Inheritance diagram for Emulator.IClosable:



Public Member Functions

- void [Close](#) ()

6.16.1 Detailed Description

Definition at line 3 of file [IClosable.cs](#).

6.16.2 Member Function Documentation

6.16.2.1 Close() `void Emulator.IClosable.Close ()`

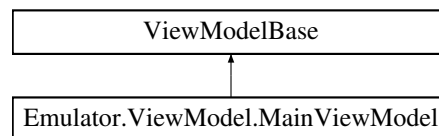
The documentation for this interface was generated from the following file:

- [Emulator/Interfaces/IClosable.cs](#)

6.17 Emulator.ViewModel.MainViewModel Class Reference

The Main [ViewModel](#)

Inheritance diagram for Emulator.ViewModel.MainViewModel:

**Public Member Functions**

- [MainViewModel](#) ()
Creates a new Instance of the [MainViewModel](#).
- void [OnLoad](#) (Object sender, RoutedEventArgs e)
- void [OnClose](#) (Object sender, CancelEventArgs e)

Properties

- [HM62256 HM62256](#) [get, set]
The 62256 RAM.
- [W65C02 W65C02](#) [get, private set]
The 65C02 Processor.
- [W65C22 W65C22](#) [get, private set]
General Purpose I/O, Shift Registers and Timers.
- [W65C22 MM65SIB](#) [get, private set]
Memory management and 65SIB.
- [W65C51 W65C51](#) [get, private set]
The ACIA serial interface.
- [AT28CXX AT28C64](#) [get, private set]
The AT28C010 ROM.
- [AT28CXX AT28C010](#) [get, private set]
The AT28C010 ROM.
- [MultiThreadedObservableCollection< MemoryRowModel > MemoryPage](#) [get, set]
The Current Memory Page
- [MultiThreadedObservableCollection< OutputLog > OutputLog](#) [get, private set]
The output log
- [MultiThreadedObservableCollection< Breakpoint > Breakpoints](#) [get, set]
The Breakpoints
- [Breakpoint SelectedBreakpoint](#) [get, set]
The Currently Selected Breakpoint

- [RomFileModel RomFile](#) [get, set]
The currently loaded binary file. (If it is indeed loaded, that is.)
- string [CurrentDisassembly](#) [get]
The Current Disassembly
- int [NumberOfCycles](#) [get, private set]
The number of cycles.
- bool [IsRunning](#) [get, set]
Is the Program Running
- bool [IsRomLoaded](#) [get, set]
Is the banked ROM Loaded.
- int [CpuSpeed](#) [get, set]
The Slider CPU Speed
- static [SettingsModel SettingsModel](#) [get, set]
The [Model](#) used for saving, loading and using data from [Settings.xml](#)
- RelayCommand [StepCommand](#) [get, set]
RelayCommand for Stepping through the program one instruction at a time.
- RelayCommand [MemoryVisualCommand](#) [get, set]
RelayCommand for opening the Memory View window.
- RelayCommand [ResetCommand](#) [get, set]
Relay Command to Reset the Program back to its initial state.
- RelayCommand [RunPauseCommand](#) [get, set]
Relay Command that Run/Pauses Execution
- RelayCommand [UpdateMemoryMapCommand](#) [get, set]
Relay Command that updates the Memory Map when the Page changes
- RelayCommand [AddBreakPointCommand](#) [get, set]
The Relay Command that adds a new breakpoint
- RelayCommand [AboutCommand](#) [get, set]
The Relay Command that opens the About window.
- RelayCommand [RemoveBreakPointCommand](#) [get, set]
The Relay Command that Removes an existing breakpoint.
- RelayCommand [SettingsCommand](#) [get, set]
The Command that loads or saves the settings.
- RelayCommand< [IClosable](#) > [CloseCommand](#) [get, private set]
The Command that loads or saves the settings.
- string [CurrentSerialPort](#) [get]
The current serial port object name.
- string [WindowTitle](#) [get]
The title for the main window.

Private Member Functions

- void [Close](#) ([IClosable](#) window)
- void [BinaryLoadedNotification](#) (NotificationMessage< [RomFileModel](#) > notificationMessage)
- void [StateLoadedNotification](#) (NotificationMessage< [StateFileModel](#) > notificationMessage)
- void [GenericNotification](#) (NotificationMessage notificationMessage)
- void [SettingsAppliedNotification](#) (NotificationMessage< [SettingsModel](#) > notificationMessage)
- void [Reset](#) ()
- void [Step](#) ()
- void [UpdateUi](#) ()
- void [StepProcessor](#) ()
- [OutputLog](#) [GetOutputLog](#) ()

- void [RunPause](#) ()
- void [BackgroundWorkerDoWork](#) (object sender, DoWorkEventArgs e)
- bool [IsBreakPointTriggered](#) ()
- int [GetLogModValue](#) ()
- int [GetSleepValue](#) ()
- void [UpdateMemoryPage](#) ()
- void [About](#) ()
- void [Settings](#) ()
- void [MemoryView](#) ()
- void [AddBreakPoint](#) ()
- void [RemoveBreakPoint](#) ()

Private Attributes

- readonly BackgroundWorker [_backgroundWorker](#)
- bool [_breakpointTriggered](#)

6.17.1 Detailed Description

The Main [ViewModel](#)

Definition at line 26 of file [MainViewModel.cs](#).

6.17.2 Constructor & Destructor Documentation

6.17.2.1 MainViewModel() `Emulator.ViewModel.MainViewModel.MainViewModel () [inline]`

Creates a new Instance of the [MainViewModel](#).

Definition at line 216 of file [MainViewModel.cs](#).

```

00217     {
00218         var _formatter = new XmlSerializer(typeof(SettingsModel));
00219         Stream _stream = new FileStream(FileLocations.SettingsFile, FileMode.OpenOrCreate);
00220         if (!(_stream == null) || (0 >= _stream.Length))
00221         {
00222             SettingsModel = (SettingsModel)_formatter.Deserialize(_stream);
00223             if ((SettingsModel.SettingsVersionMajor < Versioning.SettingsFile.Major) ||
00224                 (SettingsModel.SettingsVersionMinor < Versioning.SettingsFile.Minor) ||
00225                 (SettingsModel.SettingsVersionBuild < Versioning.SettingsFile.Build) ||
00226                 (SettingsModel.SettingsVersionRevision < Versioning.SettingsFile.Revision))
00227             {
00228                 MessageBox.Show("Settings file contains old information...\nDeleting old settings
file...",
00229                                 "Settings file stale!", MessageBoxButton.OKCancel,
00230                                 MessageBoxImage.Warning,
00231                                 MessageBoxResult.OK);
00232                 // Close the file, then delete it.
00233                 _stream.Close();
00234                 File.Delete(FileLocations.SettingsFile);
00235                 SettingsModel = SettingsFile.CreateNew();
00236             }
00237         }
00238         else
00239         {
00240             MessageBox.Show("Creating new settings file...");
00241             SettingsModel = SettingsFile.CreateNew();
00242         }
00243         _stream.Close();

```

```

00244         HM62256 = new HM62256(MemoryMap.BankedRam.TotalBanks, MemoryMap.BankedRam.Offset,
MemoryMap.BankedRam.Length);
00245         AT28C64 = new AT28CXX(MemoryMap.SharedRom.Offset, MemoryMap.SharedRom.Length, 1);
00246         AT28C010 = new AT28CXX(MemoryMap.BankedRom.Offset, MemoryMap.BankedRom.Length,
MemoryMap.BankedRom.TotalBanks);
00247         W65C02 = new W65C02();
00248         W65C51 = new W65C51(W65C02, MemoryMap.Devices.ACIA.Offset);
00249         W65C51.Init(SettingsModel.ComPortName.ToString());
00250         W65C22 = new W65C22(W65C02, MemoryMap.Devices.GPIO.Offset, MemoryMap.Devices.GPIO.Length);
00251         W65C22.Init(1000);
00252         MM65SIB = new W65C22(W65C02, MemoryMap.Devices.MM65SIB.Offset,
MemoryMap.Devices.MM65SIB.Length);
00253         MM65SIB.Init(1000);
00254
00255         MemoryMap.Init(W65C02, W65C22, MM65SIB, W65C51, HM62256, AT28C010, AT28C64);
00256
00257         // Now we can load the BIOS.
00258         byte[][] _bios = AT28C64.ReadFile(FileLocations.BiosFile);
00259         if (_bios == null)
00260         {
00261             Environment.Exit(ExitCodes.NO_BIOS);
00262         }
00263         AT28C64.Load(_bios);
00264
00265         AboutCommand = new RelayCommand(About);
00266         AddBreakPointCommand = new RelayCommand(AddBreakPoint);
00267         CloseCommand = new RelayCommand<IClosable>(Close);
00268         MemoryVisualCommand = new RelayCommand(MemoryView);
00269         RemoveBreakPointCommand = new RelayCommand(RemoveBreakPoint);
00270         ResetCommand = new RelayCommand(Reset);
00271         RunPauseCommand = new RelayCommand(RunPause);
00272         SettingsCommand = new RelayCommand(Settings);
00273         StepCommand = new RelayCommand(Step);
00274
00275         Messenger.Default.Register<NotificationMessage>(this, GenericNotification);
00276         Messenger.Default.Register<NotificationMessage<RomFileModel>>(this,
BinaryLoadedNotification);
00277         Messenger.Default.Register<NotificationMessage<SettingsModel>>(this,
SettingsAppliedNotification);
00278         Messenger.Default.Register<NotificationMessage<StateFileModel>>(this,
StateLoadedNotification);
00279
00280         MemoryPage = new MultiThreadedObservableCollection<MemoryRowModel>();
00281         OutputLog = new MultiThreadedObservableCollection<OutputLog>();
00282         Breakpoints = new MultiThreadedObservableCollection<Breakpoint>();
00283
00284         UpdateMemoryPage();
00285
00286         _backgroundWorker = new BackgroundWorker { WorkerSupportsCancellation = true,
WorkerReportsProgress = false };
00287         _backgroundWorker.DoWork += BackgroundWorkerDoWork;
00288         Application.Current.MainWindow.Title = Versioning.Product.Title;
00289         Application.Current.MainWindow.Closing += new CancelEventHandler(OnClose);
00290         Application.Current.MainWindow.Loaded += new RoutedEventHandler(OnLoad);
00291
00292         Reset();
00293     }

```

6.17.3 Member Function Documentation

6.17.3.1 About() void Emulator.ViewModel.MainViewModel.About () [inline], [private]

Definition at line 718 of file [MainViewModel.cs](#).

```

00719     {
00720         IsRunning = false;
00721
00722         if (_backgroundWorker.IsBusy)
00723             _backgroundWorker.CancelAsync();
00724
00725         MessageBox.Show(string.Format("{0}\n{1}\nVersion: {2}\nCompany: {3}",
Versioning.Product.Name, Versioning.Product.Description, Versioning.Product.VersionString,
Versioning.Product.Company), Versioning.Product.Title);
00726     }

```

6.17.3.2 AddBreakPoint() void Emulator.ViewModel.MainViewModel.AddBreakPoint () [inline], [private]

Definition at line 743 of file [MainViewModel.cs](#).

```
00744     {
00745         Breakpoints.Add(new Breakpoint());
00746         RaisePropertyChanged("Breakpoints");
00747     }
```

6.17.3.3 BackgroundWorkerDoWork() void Emulator.ViewModel.MainViewModel.BackgroundWorkerDoWork (object sender, DoWorkEventArgs e) [inline], [private]

Definition at line 589 of file [MainViewModel.cs](#).

```
00590     {
00591         var worker = sender as BackgroundWorker;
00592         var outputLogs = new List<OutputLog>();
00593
00594         while (true)
00595         {
00596             if (worker != null && worker.CancellationPending || IsBreakPointTriggered())
00597             {
00598                 e.Cancel = true;
00599
00600                 RaisePropertyChanged("W65C02");
00601
00602                 foreach (var log in outputLogs)
00603                     OutputLog.Insert(0, log);
00604
00605                 UpdateMemoryPage();
00606                 return;
00607             }
00608
00609             StepProcessor();
00610             outputLogs.Add(GetOutputLog());
00611
00612             if (NumberOfCycles % GetLogModValue() == 0)
00613             {
00614                 foreach (var log in outputLogs)
00615                     OutputLog.Insert(0, log);
00616
00617                 outputLogs.Clear();
00618                 UpdateUi();
00619             }
00620             Thread.Sleep(GetSleepValue());
00621         }
00622     }
```

6.17.3.4 BinaryLoadedNotification() void Emulator.ViewModel.MainViewModel.BinaryLoadedNotification (NotificationMessage< RomFileModel > notificationMessage) [inline], [private]

Definition at line 356 of file [MainViewModel.cs](#).

```
00357     {
00358         if (notificationMessage.Notification != "FileLoaded")
00359         {
00360             return;
00361         }
00362
00363         // Load Banked ROM
00364         AT28C010.Load(notificationMessage.Content.Rom);
00365         IsRomLoaded = true;
00366         RaisePropertyChanged("IsRomLoaded");
00367
00368         Reset();
00369     }
```

6.17.3.5 Close() void Emulator.ViewModel.MainViewModel.Close (
 IClosable window) [inline], [private]

Definition at line 348 of file [MainViewModel.cs](#).

```
00349     {
00350         if ((window != null) && (!IsRunning))
00351         {
00352             Environment.Exit(ExitCodes.NO_ERROR);
00353         }
00354     }
```

6.17.3.6 GenericNotification() void Emulator.ViewModel.MainViewModel.GenericNotification (
 NotificationMessage notificationMessage) [inline], [private]

Definition at line 398 of file [MainViewModel.cs](#).

```
00399     {
00400         if (notificationMessage.Notification == "CloseFile")
00401         {
00402             AT28C010.Clear();
00403             if (IsRunning) { RunPause(); }
00404             IsRomLoaded = false;
00405             RaisePropertyChanged("IsRomLoaded");
00406             return;
00407         }
00408         else if (notificationMessage.Notification == "LoadFile")
00409         {
00410             var dialog = new OpenFileDialog
00411             {
00412                 DefaultExt = ".bin",
00413                 Filter =
00414                     "All Files (*.bin, *.65C02)|*.bin;*.65C02|Binary
Assembly (*.bin)|" +
00415                     "*.bin|WolfNet 65C02 Emulator Save State
(*.65C02)|*.65C02"
00416             };
00417             var result = dialog.ShowDialog();
00418             if (result != true)
00419             {
00420                 return;
00421             }
00422
00423             if (Path.GetExtension(dialog.FileName.ToUpper()) == ".BIN")
00424             {
00425                 byte[][] _rom = AT28C010.ReadFile(dialog.FileName);
00426
00427                 Messenger.Default.Send(new NotificationMessage<RomFileModel>(new RomFileModel
00428                 {
00429                     Rom = _rom,
00430                     RomBanks = AT28C010.Banks,
00431                     RomBankSize = AT28C010.Length,
00432                     RomFilePath = dialog.FileName,
00433                     RomFileName = Path.GetFileName(dialog.FileName),
00434                     }, "FileLoaded"));
00435             }
00436             else if (Path.GetExtension(dialog.FileName.ToUpper()) == ".6502")
00437             {
00438                 var formatter = new BinaryFormatter();
00439                 Stream stream = new FileStream(dialog.FileName, FileMode.Open);
00440                 var fileModel = (StateFileModel)formatter.Deserialize(stream);
00441
00442                 stream.Close();
00443
00444                 Messenger.Default.Send(new NotificationMessage<StateFileModel>(fileModel,
"StateLoaded"));
00445             }
00446         }
00447         else if (notificationMessage.Notification == "SaveState")
00448         {
00449             var dialog = new SaveFileDialog
00450             {
00451                 DefaultExt = ".65C02",
00452                 Filter =
00453                     "WolfNet W65C02 Emulator Save State
(*.65C02)|*.65C02"
00454             };
00455             var result = dialog.ShowDialog();
00456
00457             if (result != true)
```

```

00458         {
00459             return;
00460         }
00461
00462         var formatter = new BinaryFormatter();
00463         Stream stream = new FileStream(dialog.FileName, FileMode.Create, FileAccess.Write,
        FileShare.None);
00464
00465         formatter.Serialize(stream, new StateFileModel
00466         {
00467             NumberOfCycles = NumberOfCycles,
00468             OutputLog = OutputLog,
00469             W65C02 = W65C02,
00470             W65C22 = W65C22,
00471             MM65SIB = MM65SIB,
00472             W65C51 = W65C51,
00473             AT28C010 = AT28C010,
00474             AT28C64 = AT28C64,
00475         });
00476         stream.Close();
00477     }
00478     else
00479     {
00480         return;
00481     }
00482 }

```

6.17.3.7 GetLogModValue() `int Emulator.ViewModel.MainViewModel.GetLogModValue () [inline], [private]`

Definition at line 656 of file [MainViewModel.cs](#).

```

00657     {
00658         switch (CpuSpeed)
00659         {
00660             case 0:
00661             case 1:
00662             case 2:
00663             case 3:
00664             case 4:
00665             case 5:
00666                 return 1;
00667             case 6:
00668                 return 5;
00669             case 7:
00670                 return 20;
00671             case 8:
00672                 return 30;
00673             case 9:
00674                 return 40;
00675             case 10:
00676                 return 50;
00677             default:
00678                 return 5;
00679         }
00680     }

```

6.17.3.8 GetOutputLog() `OutputLog Emulator.ViewModel.MainViewModel.GetOutputLog () [inline], [private]`

Definition at line 558 of file [MainViewModel.cs](#).

```

00559     {
00560         if (W65C02.CurrentDisassembly == null)
00561         {
00562             return new OutputLog(new Disassembly());
00563         }
00564
00565         return new OutputLog(W65C02.CurrentDisassembly)
00566         {
00567             XRegister = W65C02.XRegister.ToString("X").PadLeft(2, '0'),
00568             YRegister = W65C02.YRegister.ToString("X").PadLeft(2, '0'),
00569             Accumulator = W65C02.Accumulator.ToString("X").PadLeft(2, '0'),
00570             NumberOfCycles = NumberOfCycles,
00571             StackPointer = W65C02.StackPointer.ToString("X").PadLeft(2, '0'),

```

```

00572         ProgramCounter = W65C02.ProgramCounter.ToString("X").PadLeft(4, '0'),
00573         CurrentOpCode = W65C02.CurrentOpCode.ToString("X").PadLeft(2, '0')
00574     };
00575 }

```

6.17.3.9 GetSleepValue() `int Emulator.ViewModel.MainViewModel.GetSleepValue () [inline], [private]`

Definition at line 682 of file [MainViewModel.cs](#).

```

00683     {
00684         switch (CpuSpeed)
00685         {
00686             case 0:
00687                 return 550;
00688             case 1:
00689                 return 550;
00690             case 2:
00691                 return 440;
00692             case 3:
00693                 return 330;
00694             case 4:
00695                 return 220;
00696             case 5:
00697                 return 160;
00698             case 6:
00699                 return 80;
00700             case 7:
00701                 return 40;
00702             case 8:
00703                 return 20;
00704             case 9:
00705                 return 10;
00706             case 10:
00707                 return 5;
00708             default:
00709                 return 5;
00710         }
00711     }

```

6.17.3.10 IsBreakPointTriggered() `bool Emulator.ViewModel.MainViewModel.IsBreakPointTriggered () [inline], [private]`

Definition at line 624 of file [MainViewModel.cs](#).

```

00625     {
00626         //This prevents the Run Command from getting stuck after reaching a breakpoint
00627         if (_breakpointTriggered)
00628         {
00629             _breakpointTriggered = false;
00630             return false;
00631         }
00632         foreach (var breakpoint in Breakpoints.Where(x => x.IsEnabled))
00633         {
00634             if (!int.TryParse(breakpoint.Value, NumberStyles.AllowHexSpecifier,
00635                 CultureInfo.InvariantCulture, out int value))
00636                 continue;
00637             if (breakpoint.Type == BreakpointType.NumberOfCycleType && value == NumberOfCycles)
00638             {
00639                 _breakpointTriggered = true;
00640                 RunPause();
00641                 return true;
00642             }
00643             if (breakpoint.Type == BreakpointType.ProgramCounterType && value ==
00644                 W65C02.ProgramCounter)
00645             {
00646                 _breakpointTriggered = true;
00647                 RunPause();
00648                 return true;
00649             }
00650         }
00651         return false;
00652     }
00653 }
00654

```

6.17.3.11 MemoryView() void Emulator.ViewModel.MainViewModel.MemoryView () [inline], [private]

Definition at line 738 of file [MainViewModel.cs](#).

```
00739     {
00740         Messenger.Default.Send(new NotificationMessage("MemoryVisualWindow"));
00741     }
```

6.17.3.12 OnClose() void Emulator.ViewModel.MainViewModel.OnClose (
Object sender,
CancelEventArgs e) [inline]

Definition at line 314 of file [MainViewModel.cs](#).

```
00315     {
00316         e.Cancel = false;
00317         if (IsRunning)
00318         {
00319             MessageBox.Show("You can't quit the emulator while it is actively running!",
00320                             "You can't do that!", MessageBoxButton.OK, MessageBoxImage.Stop);
00321             e.Cancel = true;
00322             return;
00323         }
00324         #if !DEBUG
00325         else
00326         {
00327             var result = MessageBox.Show("Are you sure you want to quit the emulator?",
00328                                         "To quit, or not to quit -- that is the question.",
00329                                         MessageBoxButton.YesNo, MessageBoxImage.Question,
00330                                         MessageBoxResult.No);
00331             if (result == MessageBoxResult.No)
00332             {
00333                 e.Cancel = true;
00334                 return;
00335             }
00336         }
00337         #endif
00338         Stream stream = new FileStream(FileLocations.SettingsFile, FileMode.Create,
00339 FileAccess.Write, FileShare.None);
00339         XmlSerializer XmlFormatter = new XmlSerializer(typeof(SettingsModel));
00340         XmlFormatter.Serialize(stream, MainViewModel.SettingsModel);
00341         stream.Flush();
00342         stream.Close();
00343         W65C51.Fini();
00344     }
```

6.17.3.13 OnLoad() void Emulator.ViewModel.MainViewModel.OnLoad (
Object sender,
RoutedEventArgs e) [inline]

Definition at line 295 of file [MainViewModel.cs](#).

```
00296     {
00297         #if !DEBUG
00298         if (Versioning.Product.Major < 1)
00299         {
00300             var result = MessageBox.Show(String.Format("Thank you for using {0}\n" +
00301                                                         "Be warned that this is a beta build.\n" +
00302                                                         "It may break or have bugs.",
00303 Versioning.Product.Name),
00304                                         Versioning.Product.Title,
00305                                         MessageBoxButton.OKCancel,
00306                                         MessageBoxImage.Warning,
00307                                         MessageBoxResult.None);
00308             if (result == MessageBoxResult.Cancel)
00309             {
00310                 // Exit without making any changes.
00311                 Environment.Exit(ExitCodes.NO_ERROR);
00312             }
00313         }
00314         #endif
00315     }
```


6.17.3.14 RemoveBreakPoint() void Emulator.ViewModel.MainViewModel.RemoveBreakPoint () [inline], [private]

Definition at line 749 of file [MainViewModel.cs](#).

```
00750     {
00751         if (SelectedBreakpoint == null)
00752             return;
00753
00754         Breakpoints.Remove(SelectedBreakpoint);
00755         SelectedBreakpoint = null;
00756         RaisePropertyChanged("SelectedBreakpoint");
00757     }
```

6.17.3.15 Reset() void Emulator.ViewModel.MainViewModel.Reset () [inline], [private]

Definition at line 497 of file [MainViewModel.cs](#).

```
00498     {
00499         IsRunning = false;
00500
00501         if (_backgroundWorker.IsBusy)
00502             _backgroundWorker.CancelAsync();
00503
00504         // "Reset" the Hardware...
00505         W65C02.Reset();
00506         RaisePropertyChanged("W65C02");
00507         W65C22.Reset();
00508         RaisePropertyChanged("W65C22");
00509         MM65SIB.Reset();
00510         RaisePropertyChanged("MM65SIB");
00511         W65C51.Reset();
00512         RaisePropertyChanged("W65C51");
00513         HM62256.Reset();
00514         RaisePropertyChanged("HM62256");
00515
00516         IsRunning = false;
00517         NumberOfCycles = 0;
00518         RaisePropertyChanged("NumberOfCycles");
00519
00520         UpdateMemoryPage();
00521         RaisePropertyChanged("MemoryPage");
00522
00523         OutputLog.Clear();
00524         RaisePropertyChanged("CurrentDisassembly");
00525
00526         OutputLog.Insert(0, GetOutputLog());
00527         UpdateUi();
00528     }
```

6.17.3.16 RunPause() void Emulator.ViewModel.MainViewModel.RunPause () [inline], [private]

Definition at line 577 of file [MainViewModel.cs](#).

```
00578     {
00579         var isRunning = !IsRunning;
00580
00581         if (isRunning)
00582             _backgroundWorker.RunWorkerAsync();
00583         else
00584             _backgroundWorker.CancelAsync();
00585
00586         IsRunning = !IsRunning;
00587     }
```

6.17.3.17 Settings() void Emulator.ViewModel.MainViewModel.Settings () [inline], [private]Definition at line 728 of file [MainViewModel.cs](#).

```

00729     {
00730         IsRunning = false;
00731
00732         if (_backgroundWorker.IsBusy)
00733             _backgroundWorker.CancelAsync();
00734
00735         Messenger.Default.Send(new NotificationMessage<SettingsModel>(SettingsModel,
00736             "SettingsWindow"));
00737     }

```

6.17.3.18 SettingsAppliedNotification() void Emulator.ViewModel.MainViewModel.SettingsApplied←Notification (NotificationMessage< SettingsModel > notificationMessage) [inline], [private]Definition at line 484 of file [MainViewModel.cs](#).

```

00485     {
00486         if (notificationMessage.Notification != "SettingsApplied")
00487         {
00488             return;
00489         }
00490
00491         SettingsModel = notificationMessage.Content;
00492         W65C51.Init(notificationMessage.Content.ComPortName);
00493         RaisePropertyChanged("SettingsModel");
00494         UpdateUi();
00495     }

```

6.17.3.19 StateLoadedNotification() void Emulator.ViewModel.MainViewModel.StateLoadedNotification (NotificationMessage< StateFileModel > notificationMessage) [inline], [private]Definition at line 371 of file [MainViewModel.cs](#).

```

00372     {
00373         if (notificationMessage.Notification != "StateLoaded")
00374         {
00375             return;
00376         }
00377
00378         Reset();
00379
00380         OutputLog = new
00381             MultiThreadedObservableCollection<OutputLog>(notificationMessage.Content.OutputLog);
00382         RaisePropertyChanged("OutputLog");
00383
00384         NumberOfCycles = notificationMessage.Content.NumberOfCycles;
00385
00386         W65C02 = notificationMessage.Content.W65C02;
00387         W65C22 = notificationMessage.Content.W65C22;
00388         MM65SIB = notificationMessage.Content.MM65SIB;
00389         W65C51 = notificationMessage.Content.W65C51;
00390         AT28C010 = notificationMessage.Content.AT28C010;
00391         AT28C64 = notificationMessage.Content.AT28C64;
00392         UpdateMemoryPage();
00393         UpdateUi();
00394
00395         IsRomLoaded = true;
00396         RaisePropertyChanged("IsRomLoaded");
00397     }

```

6.17.3.20 Step() void Emulator.ViewModel.MainViewModel.Step () [inline], [private]

Definition at line 530 of file [MainViewModel.cs](#).

```
00531     {
00532         IsRunning = false;
00533
00534         if (_backgroundWorker.IsBusy)
00535             _backgroundWorker.CancelAsync();
00536
00537         StepProcessor();
00538         UpdateMemoryPage();
00539
00540         OutputLog.Insert(0, GetOutputLog());
00541         UpdateUi();
00542     }
```

6.17.3.21 StepProcessor() void Emulator.ViewModel.MainViewModel.StepProcessor () [inline], [private]

Definition at line 552 of file [MainViewModel.cs](#).

```
00553     {
00554         W65C02.NextStep();
00555         NumberOfCycles = W65C02.GetCycleCount();
00556     }
```

6.17.3.22 UpdateMemoryPage() void Emulator.ViewModel.MainViewModel.UpdateMemoryPage () [inline], [private]

Definition at line 713 of file [MainViewModel.cs](#).

```
00714     {
00715         Messenger.Default.Send(new NotificationMessage("UpdateMemoryPage"));
00716     }
```

6.17.3.23 UpdateUi() void Emulator.ViewModel.MainViewModel.UpdateUi () [inline], [private]

Definition at line 544 of file [MainViewModel.cs](#).

```
00545     {
00546         RaisePropertyChanged("W65C02");
00547         RaisePropertyChanged("NumberOfCycles");
00548         RaisePropertyChanged("CurrentDisassembly");
00549         RaisePropertyChanged("MemoryPage");
00550     }
```

6.17.4 Member Data Documentation

6.17.4.1 _backgroundWorker readonly BackgroundWorker Emulator.ViewModel.MainViewModel._↵
backgroundWorker [private]

Definition at line 29 of file [MainViewModel.cs](#).

6.17.4.2 `_breakpointTriggered` `bool` `Emulator.ViewModel.MainViewModel._breakpointTriggered` [private]

Definition at line 30 of file [MainViewModel.cs](#).

6.17.5 Property Documentation

6.17.5.1 `AboutCommand` `RelayCommand` `Emulator.ViewModel.MainViewModel.AboutCommand` [get], [set]

The Relay Command that opens the About window.

Definition at line 178 of file [MainViewModel.cs](#).

```
00178 { get; set; }
```

6.17.5.2 `AddBreakPointCommand` `RelayCommand` `Emulator.ViewModel.MainViewModel.AddBreakPointCommand` [get], [set]

The Relay Command that adds a new breakpoint

Definition at line 173 of file [MainViewModel.cs](#).

```
00173 { get; set; }
```

6.17.5.3 `AT28C010` `AT28CXX` `Emulator.ViewModel.MainViewModel.AT28C010` [get], [private set]

The AT28C010 ROM.

Definition at line 67 of file [MainViewModel.cs](#).

```
00067 { get; private set; }
```

6.17.5.4 `AT28C64` `AT28CXX` `Emulator.ViewModel.MainViewModel.AT28C64` [get], [private set]

The AT28C010 ROM.

Definition at line 62 of file [MainViewModel.cs](#).

```
00062 { get; private set; }
```

6.17.5.5 `Breakpoints` `MultiThreadedObservableCollection<Breakpoint>` `Emulator.ViewModel.MainViewModel.Breakpoints` [get], [set]

The Breakpoints

Definition at line 82 of file [MainViewModel.cs](#).

```
00082 { get; set; }
```

6.17.5.6 CloseCommand `RelayCommand<IClosable> Emulator.ViewModel.MainViewModel.CloseCommand [get], [private set]`

The Command that loads or saves the settings.

Definition at line 193 of file [MainViewModel.cs](#).

```
00193 { get; private set; }
```

6.17.5.7 CpuSpeed `int Emulator.ViewModel.MainViewModel.CpuSpeed [get], [set]`

The Slider CPU Speed

Definition at line 138 of file [MainViewModel.cs](#).

```
00138 { get; set; }
```

6.17.5.8 CurrentDisassembly `string Emulator.ViewModel.MainViewModel.CurrentDisassembly [get]`

The Current Disassembly

Definition at line 97 of file [MainViewModel.cs](#).

```
00098 {
00099     get
00100     {
00101         if (W65C02.CurrentDisassembly != null)
00102         {
00103             return string.Format("{0} {1}", W65C02.CurrentDisassembly.OpCodeString,
W65C02.CurrentDisassembly.DisassemblyOutput);
00104         }
00105         else
00106         {
00107             return string.Empty;
00108         }
00109     }
00110 }
```

6.17.5.9 CurrentSerialPort `string Emulator.ViewModel.MainViewModel.CurrentSerialPort [get]`

The current serial port object name.

Definition at line 198 of file [MainViewModel.cs](#).

```
00199 {
00200     get
00201     {
00202         return W65C51.ObjectName;
00203     }
00204 }
```

6.17.5.10 HM62256 `HM62256 Emulator.ViewModel.MainViewModel.HM62256 [get], [set], [private]`

The 62256 RAM.

Definition at line 37 of file [MainViewModel.cs](#).

```
00037 { get; set; }
```

6.17.5.11 IsRomLoaded `bool Emulator.ViewModel.MainViewModel.IsRomLoaded [get], [set]`

Is the banked ROM Loaded.

Definition at line 133 of file [MainViewModel.cs](#).

```
00133 { get; set; }
```

6.17.5.12 IsRunning `bool Emulator.ViewModel.MainViewModel.IsRunning [get], [set]`

Is the Program Running

Definition at line 120 of file [MainViewModel.cs](#).

```
00121     {  
00122         get { return W65C02.IsRunning; }  
00123         set  
00124     {  
00125         W65C02.IsRunning = value;  
00126         RaisePropertyChanged("IsRunning");  
00127     }  
00128 }
```

6.17.5.13 MemoryPage `MultiThreadedObservableCollection<MemoryRowModel> Emulator.ViewModel.MainViewModel.MemoryPage [get], [set]`

The Current Memory Page

Definition at line 72 of file [MainViewModel.cs](#).

```
00072 { get; set; }
```

6.17.5.14 MemoryVisualCommand `RelayCommand Emulator.ViewModel.MainViewModel.MemoryVisualCommand [get], [set]`

RelayCommand for opening the Memory View window.

Definition at line 153 of file [MainViewModel.cs](#).

```
00153 { get; set; }
```

6.17.5.15 MM65SIB `W65C22 Emulator.ViewModel.MainViewModel.MM65SIB [get], [private set]`

Memory management and 65SIB.

Definition at line 52 of file [MainViewModel.cs](#).

```
00052 { get; private set; }
```

6.17.5.16 NumberOfCycles `int Emulator.ViewModel.MainViewModel.NumberOfCycles [get], [private set]`

The number of cycles.

Definition at line 115 of file [MainViewModel.cs](#).

```
00115 { get; private set; }
```

6.17.5.17 OutputLog `MultiThreadedObservableCollection<OutputLog> Emulator.ViewModel.MainViewModel.OutputLog [get], [private set]`

The output log

Definition at line 77 of file [MainViewModel.cs](#).

```
00077 { get; private set; }
```

6.17.5.18 RemoveBreakPointCommand `RelayCommand Emulator.ViewModel.MainViewModel.RemoveBreakPointCommand [get], [set]`

The Relay Command that Removes an existing breakpoint.

Definition at line 183 of file [MainViewModel.cs](#).

```
00183 { get; set; }
```

6.17.5.19 ResetCommand `RelayCommand Emulator.ViewModel.MainViewModel.ResetCommand [get], [set]`

Relay Command to Reset the Program back to its initial state.

Definition at line 158 of file [MainViewModel.cs](#).

```
00158 { get; set; }
```

6.17.5.20 RomFile `RomFileModel Emulator.ViewModel.MainViewModel.RomFile [get], [set]`

The currently loaded binary file. (If it is indeed loaded, that is.)

Definition at line 92 of file [MainViewModel.cs](#).

```
00092 { get; set; }
```

6.17.5.21 RunPauseCommand `RelayCommand Emulator.ViewModel.MainViewModel.RunPauseCommand [get], [set]`

Relay Command that Run/Pauses Execution

Definition at line 163 of file [MainViewModel.cs](#).

```
00163 { get; set; }
```

6.17.5.22 SelectedBreakpoint [Breakpoint](#) Emulator.ViewModel.MainViewModel.SelectedBreakpoint
[get], [set]

The Currently Selected Breakpoint

Definition at line 87 of file [MainViewModel.cs](#).

```
00087 { get; set; }
```

6.17.5.23 SettingsCommand [RelayCommand](#) Emulator.ViewModel.MainViewModel.SettingsCommand
[get], [set]

The Command that loads or saves the settings.

Definition at line 188 of file [MainViewModel.cs](#).

```
00188 { get; set; }
```

6.17.5.24 SettingsModel [SettingsModel](#) Emulator.ViewModel.MainViewModel.SettingsModel [static],
[get], [set]

The [Model](#) used for saving, loading and using data from Settings.xml

Definition at line 143 of file [MainViewModel.cs](#).

```
00143 { get; set; }
```

6.17.5.25 StepCommand [RelayCommand](#) Emulator.ViewModel.MainViewModel.StepCommand [get],
[set]

RelayCommand for Stepping through the program one instruction at a time.

Definition at line 148 of file [MainViewModel.cs](#).

```
00148 { get; set; }
```

6.17.5.26 UpdateMemoryMapCommand [RelayCommand](#) Emulator.ViewModel.MainViewModel.Update↵
MemoryMapCommand [get], [set]

Relay Command that updates the Memory Map when the Page changes

Definition at line 168 of file [MainViewModel.cs](#).

```
00168 { get; set; }
```

6.17.5.27 W65C02 [W65C02](#) Emulator.ViewModel.MainViewModel.W65C02 [get], [private set]

The 65C02 Processor.

Definition at line 42 of file [MainViewModel.cs](#).

```
00042 { get; private set; }
```


6.17.5.28 W65C22 [W65C22](#) `Emulator.ViewModel.MainViewModel.W65C22` `[get], [private set]`

General Purpose I/O, Shift Registers and Timers.

Definition at line 47 of file [MainViewModel.cs](#).

```
00047 { get; private set; }
```

6.17.5.29 W65C51 [W65C51](#) `Emulator.ViewModel.MainViewModel.W65C51` `[get], [private set]`

The ACIA serial interface.

Definition at line 57 of file [MainViewModel.cs](#).

```
00057 { get; private set; }
```

6.17.5.30 WindowTitle `string Emulator.ViewModel.MainViewModel.WindowTitle` `[get]`

The title for the main window.

Definition at line 209 of file [MainViewModel.cs](#).

```
00209 { get { return Versioning.Product.Title; } }
```

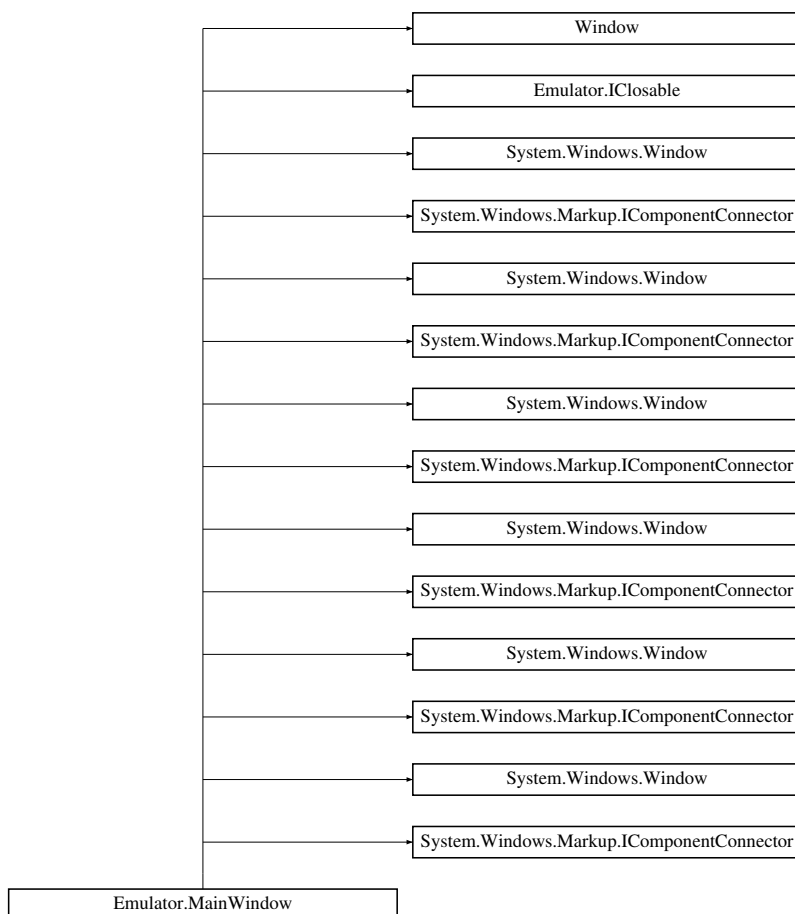
The documentation for this class was generated from the following file:

- [Emulator/ViewModel/MainViewModel.cs](#)

6.18 Emulator.MainWindow Class Reference

Interaction logic for MainWindow.xaml

Inheritance diagram for Emulator.MainWindow:



Public Member Functions

- [MainWindow](#) ()
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent

Private Member Functions

- void [ToClose](#) (Object sender, EventArgs e)
- void [LoadFile](#) (Object sender, EventArgs e)
- void [SaveFile](#) (Object sender, EventArgs e)
- void [CloseFile](#) (Object sender, EventArgs e)
- void [NotificationMessageReceived](#) (NotificationMessage notificationMessage)
- void [NotificationMessageReceived](#) (NotificationMessage< [SettingsModel](#) > notificationMessage)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)

Private Attributes

- bool [_contentLoaded](#)

6.18.1 Detailed Description

Interaction logic for MainWindow.xaml

[MainWindow](#)

Definition at line 12 of file [MainWindow.xaml.cs](#).

6.18.2 Constructor & Destructor Documentation

6.18.2.1 MainWindow() Emulator.MainWindow.MainWindow () [inline]

Definition at line 14 of file [MainWindow.xaml.cs](#).

```
00015     {
00016         InitializeComponent();
00017         Messenger.Default.Register<NotificationMessage>(this, NotificationMessageReceived);
00018         Messenger.Default.Register<NotificationMessage<SettingsModel>>(this,
NotificationMessageReceived);
00019     }
```

6.18.3 Member Function Documentation

6.18.3.1 CloseFile() void Emulator.MainWindow.CloseFile (Object sender, EventArgs e) [inline], [private]

Definition at line 36 of file [MainWindow.xaml.cs](#).

```
00037     {
00038         Messenger.Default.Send(new NotificationMessage("CloseFile"));
00039     }
```

6.18.3.2 Connect() [1/6] void System.Windows.Markup.IComponentConnector. Emulator.MainWindow.↔ Connect (int connectionId, object target) [inline], [private]

Definition at line 373 of file [MainWindow.g.cs](#).

```
00373     {
00374         switch (connectionId)
00375         {
00376             case 1:
00377                 this.EmulatorWindow = ((Emulator.MainWindow)(target));
00378                 return;
00379             case 2:
00380
00381 #line 72 "..\..\..\MainWindows.xaml"
00382                 ((System.Windows.Controls.MenuItem)(target)).Click += new
System.Windows.RoutedEventHandler(this.LoadFile);
00383
00384 #line default
00385 #line hidden
00386                 return;
00387             case 3:
00388
00389 #line 73 "..\..\..\MainWindows.xaml"
00390                 ((System.Windows.Controls.MenuItem)(target)).Click += new
System.Windows.RoutedEventHandler(this.SaveFile);
00391
00392 #line default
00393 #line hidden
00394                 return;
00395             case 4:
00396
00397 #line 74 "..\..\..\MainWindows.xaml"
00398                 ((System.Windows.Controls.MenuItem)(target)).Click += new
System.Windows.RoutedEventHandler(this.CloseFile);
00399
00400 #line default
00401 #line hidden
00402                 return;
00403             case 5:
00404
00405 #line 76 "..\..\..\MainWindows.xaml"
00406                 ((System.Windows.Controls.MenuItem)(target)).Click += new
System.Windows.RoutedEventHandler(this.ToClose);
00407
00408 #line default
```

```
00409 #line hidden
00410         return;
00411     case 6:
00412         this.OutputLog = ((System.Windows.Controls.DataGrid) (target));
00413         return;
00414     case 7:
00415         this.Run = ((System.Windows.Controls.Button) (target));
00416         return;
00417     case 8:
00418         this.Step = ((System.Windows.Controls.Button) (target));
00419         return;
00420     case 9:
00421         this.Reset = ((System.Windows.Controls.Button) (target));
00422         return;
00423     case 10:
00424         this.RomFileNameText = ((System.Windows.Controls.TextBlock) (target));
00425         return;
00426     case 11:
00427         this.ComPortNameText = ((System.Windows.Controls.TextBlock) (target));
00428         return;
00429     case 12:
00430         this.Breakpoints = ((System.Windows.Controls.DataGrid) (target));
00431         return;
00432     case 13:
00433         this.YRegister = ((System.Windows.Controls.TextBox) (target));
00434         return;
00435     case 14:
00436         this.XRegister = ((System.Windows.Controls.TextBox) (target));
00437         return;
00438     case 15:
00439         this.Accumulator = ((System.Windows.Controls.TextBox) (target));
00440         return;
00441     case 16:
00442         this.StackPointer = ((System.Windows.Controls.TextBox) (target));
00443         return;
00444     case 17:
00445         this.ProgramCounter = ((System.Windows.Controls.TextBox) (target));
00446         return;
00447     case 18:
00448         this.Dissassembly = ((System.Windows.Controls.TextBox) (target));
00449         return;
00450     case 19:
00451         this.CycleCount = ((System.Windows.Controls.TextBox) (target));
00452         return;
00453     case 20:
00454         this.XRegisterText = ((System.Windows.Controls.TextBlock) (target));
00455         return;
00456     case 21:
00457         this.YRegisterText = ((System.Windows.Controls.TextBlock) (target));
00458         return;
00459     case 22:
00460         this.StackPointerRegisterText = ((System.Windows.Controls.TextBlock) (target));
00461         return;
00462     case 23:
00463         this.AText = ((System.Windows.Controls.TextBlock) (target));
00464         return;
00465     case 24:
00466         this.CurrentInstructionText = ((System.Windows.Controls.TextBlock) (target));
00467         return;
00468     case 25:
00469         this.ProgramCounterText = ((System.Windows.Controls.TextBlock) (target));
00470         return;
00471     case 26:
00472         this.CycleCountText = ((System.Windows.Controls.TextBlock) (target));
00473         return;
00474     case 27:
00475         this.CarryFlag = ((System.Windows.Controls.CheckBox) (target));
00476         return;
00477     case 28:
00478         this.CarryFlagText = ((System.Windows.Controls.TextBlock) (target));
00479         return;
00480     case 29:
00481         this.ZeroFlag = ((System.Windows.Controls.CheckBox) (target));
00482         return;
00483     case 30:
00484         this.ZeroFlagText = ((System.Windows.Controls.TextBlock) (target));
00485         return;
00486     case 31:
00487         this.InterruptFlag = ((System.Windows.Controls.CheckBox) (target));
00488         return;
00489     case 32:
00490         this.InterruptFlagText = ((System.Windows.Controls.TextBlock) (target));
00491         return;
00492     case 33:
00493         this.BcdFlag = ((System.Windows.Controls.CheckBox) (target));
00494         return;
00495     case 34:
```

```

00496         this.BcdFlagText = ((System.Windows.Controls.TextBlock) (target));
00497         return;
00498         case 35:
00499             this.BreakFlag = ((System.Windows.Controls.CheckBox) (target));
00500             return;
00501             case 36:
00502                 this.BreakFlagText = ((System.Windows.Controls.TextBlock) (target));
00503                 return;
00504                 case 37:
00505                     this.OverflowFlag = ((System.Windows.Controls.CheckBox) (target));
00506                     return;
00507                     case 38:
00508                         this.OverflowFlagText = ((System.Windows.Controls.TextBlock) (target));
00509                         return;
00510                         case 39:
00511                             this.NegativeFlag = ((System.Windows.Controls.CheckBox) (target));
00512                             return;
00513                             case 40:
00514                                 this.NegativeFlagText = ((System.Windows.Controls.TextBlock) (target));
00515                                 return;
00516                                 case 41:
00517                                     this.CpuSpeed = ((System.Windows.Controls.Slider) (target));
00518                                     return;
00519                                     case 42:
00520                                         this.SpeedText = ((System.Windows.Controls.TextBlock) (target));
00521                                         return;
00522                                         }
00523             this._contentLoaded = true;
00524     }

```

6.18.3.3 Connect() [2/6] void System.Windows.Markup.IComponentConnector. Emulator.MainWindow.↔
Connect (

```

    int connectionId,
    object target ) [inline], [private]

```

Definition at line 373 of file [MainWindow.g.i.cs](#).

```

00373                                                                 {
00374         switch (connectionId)
00375         {
00376             case 1:
00377                 this.EmulatorWindow = ((Emulator.MainWindow) (target));
00378                 return;
00379                 case 2:
00380
00381 #line 72 "..\..\..\MainWindows.xml"
00382                 ((System.Windows.Controls.MenuItem) (target)).Click += new
System.Windows.RoutedEventHandler(this.LoadFile);
00383
00384 #line default
00385 #line hidden
00386                 return;
00387                 case 3:
00388
00389 #line 73 "..\..\..\MainWindows.xml"
00390                 ((System.Windows.Controls.MenuItem) (target)).Click += new
System.Windows.RoutedEventHandler(this.SaveFile);
00391
00392 #line default
00393 #line hidden
00394                 return;
00395                 case 4:
00396
00397 #line 74 "..\..\..\MainWindows.xml"
00398                 ((System.Windows.Controls.MenuItem) (target)).Click += new
System.Windows.RoutedEventHandler(this.CloseFile);
00399
00400 #line default
00401 #line hidden
00402                 return;
00403                 case 5:
00404
00405 #line 76 "..\..\..\MainWindows.xml"
00406                 ((System.Windows.Controls.MenuItem) (target)).Click += new
System.Windows.RoutedEventHandler(this.ToClose);
00407
00408 #line default
00409 #line hidden
00410                 return;
00411                 case 6:

```

```
00412         this.OutputLog = ((System.Windows.Controls.DataGrid) (target));
00413         return;
00414         case 7:
00415             this.Run = ((System.Windows.Controls.Button) (target));
00416             return;
00417         case 8:
00418             this.Step = ((System.Windows.Controls.Button) (target));
00419             return;
00420         case 9:
00421             this.Reset = ((System.Windows.Controls.Button) (target));
00422             return;
00423         case 10:
00424             this.RomFileNameText = ((System.Windows.Controls.TextBlock) (target));
00425             return;
00426         case 11:
00427             this.ComPortNameText = ((System.Windows.Controls.TextBlock) (target));
00428             return;
00429         case 12:
00430             this.Breakpoints = ((System.Windows.Controls.DataGrid) (target));
00431             return;
00432         case 13:
00433             this.YRegister = ((System.Windows.Controls.TextBox) (target));
00434             return;
00435         case 14:
00436             this.XRegister = ((System.Windows.Controls.TextBox) (target));
00437             return;
00438         case 15:
00439             this.Accumulator = ((System.Windows.Controls.TextBox) (target));
00440             return;
00441         case 16:
00442             this.StackPointer = ((System.Windows.Controls.TextBox) (target));
00443             return;
00444         case 17:
00445             this.ProgramCounter = ((System.Windows.Controls.TextBox) (target));
00446             return;
00447         case 18:
00448             this.Dissassembly = ((System.Windows.Controls.TextBox) (target));
00449             return;
00450         case 19:
00451             this.CycleCount = ((System.Windows.Controls.TextBox) (target));
00452             return;
00453         case 20:
00454             this.XRegisterText = ((System.Windows.Controls.TextBlock) (target));
00455             return;
00456         case 21:
00457             this.YRegisterText = ((System.Windows.Controls.TextBlock) (target));
00458             return;
00459         case 22:
00460             this.StackPointerRegisterText = ((System.Windows.Controls.TextBlock) (target));
00461             return;
00462         case 23:
00463             this.AText = ((System.Windows.Controls.TextBlock) (target));
00464             return;
00465         case 24:
00466             this.CurrentInstructionText = ((System.Windows.Controls.TextBlock) (target));
00467             return;
00468         case 25:
00469             this.ProgramCounterText = ((System.Windows.Controls.TextBlock) (target));
00470             return;
00471         case 26:
00472             this.CycleCountText = ((System.Windows.Controls.TextBlock) (target));
00473             return;
00474         case 27:
00475             this.CarryFlag = ((System.Windows.Controls.CheckBox) (target));
00476             return;
00477         case 28:
00478             this.CarryFlagText = ((System.Windows.Controls.TextBlock) (target));
00479             return;
00480         case 29:
00481             this.ZeroFlag = ((System.Windows.Controls.CheckBox) (target));
00482             return;
00483         case 30:
00484             this.ZeroFlagText = ((System.Windows.Controls.TextBlock) (target));
00485             return;
00486         case 31:
00487             this.InterruptFlag = ((System.Windows.Controls.CheckBox) (target));
00488             return;
00489         case 32:
00490             this.InterruptFlagText = ((System.Windows.Controls.TextBlock) (target));
00491             return;
00492         case 33:
00493             this.BcdFlag = ((System.Windows.Controls.CheckBox) (target));
00494             return;
00495         case 34:
00496             this.BcdFlagText = ((System.Windows.Controls.TextBlock) (target));
00497             return;
00498         case 35:
```

```

00499         this.BreakFlag = ((System.Windows.Controls.CheckBox) (target));
00500         return;
00501         case 36:
00502             this.BreakFlagText = ((System.Windows.Controls.TextBlock) (target));
00503             return;
00504             case 37:
00505                 this.OverflowFlag = ((System.Windows.Controls.CheckBox) (target));
00506                 return;
00507                 case 38:
00508                     this.OverflowFlagText = ((System.Windows.Controls.TextBlock) (target));
00509                     return;
00510                     case 39:
00511                         this.NegativeFlag = ((System.Windows.Controls.CheckBox) (target));
00512                         return;
00513                         case 40:
00514                             this.NegativeFlagText = ((System.Windows.Controls.TextBlock) (target));
00515                             return;
00516                             case 41:
00517                                 this.CpuSpeed = ((System.Windows.Controls.Slider) (target));
00518                                 return;
00519                                 case 42:
00520                                     this.SpeedText = ((System.Windows.Controls.TextBlock) (target));
00521                                     return;
00522                                     }
00523             this._contentLoaded = true;
00524     }

```

6.18.3.4 Connect() [3/6] void System.Windows.Markup.IComponentConnector. Emulator.MainWindow.↔
Connect (

```

    int connectionId,
    object target ) [inline], [private]

```

Definition at line 373 of file [MainWindow.g.cs](#).

```

00373                                                                 {
00374         switch (connectionId)
00375         {
00376             case 1:
00377                 this.EmulatorWindow = ((Emulator.MainWindow) (target));
00378                 return;
00379                 case 2:
00380
00381 #line 72 "..\..\..\MainWindow.xaml"
00382             ((System.Windows.Controls.MenuItem) (target)).Click += new
                System.Windows.RoutedEventHandler(this.LoadFile);
00383
00384 #line default
00385 #line hidden
00386                 return;
00387                 case 3:
00388
00389 #line 73 "..\..\..\MainWindow.xaml"
00390             ((System.Windows.Controls.MenuItem) (target)).Click += new
                System.Windows.RoutedEventHandler(this.SaveFile);
00391
00392 #line default
00393 #line hidden
00394                 return;
00395                 case 4:
00396
00397 #line 74 "..\..\..\MainWindow.xaml"
00398             ((System.Windows.Controls.MenuItem) (target)).Click += new
                System.Windows.RoutedEventHandler(this.CloseFile);
00399
00400 #line default
00401 #line hidden
00402                 return;
00403                 case 5:
00404
00405 #line 76 "..\..\..\MainWindow.xaml"
00406             ((System.Windows.Controls.MenuItem) (target)).Click += new
                System.Windows.RoutedEventHandler(this.ToClose);
00407
00408 #line default
00409 #line hidden
00410                 return;
00411                 case 6:
00412                     this.OutputLog = ((System.Windows.Controls.DataGrid) (target));
00413                     return;
00414                     case 7:

```

```
00415         this.Run = ((System.Windows.Controls.Button) (target));
00416         return;
00417     case 8:
00418         this.Step = ((System.Windows.Controls.Button) (target));
00419         return;
00420     case 9:
00421         this.Reset = ((System.Windows.Controls.Button) (target));
00422         return;
00423     case 10:
00424         this.RomFileNameText = ((System.Windows.Controls.TextBlock) (target));
00425         return;
00426     case 11:
00427         this.ComPortNameText = ((System.Windows.Controls.TextBlock) (target));
00428         return;
00429     case 12:
00430         this.Breakpoints = ((System.Windows.Controls.DataGrid) (target));
00431         return;
00432     case 13:
00433         this.YRegister = ((System.Windows.Controls.TextBox) (target));
00434         return;
00435     case 14:
00436         this.XRegister = ((System.Windows.Controls.TextBox) (target));
00437         return;
00438     case 15:
00439         this.Accumulator = ((System.Windows.Controls.TextBox) (target));
00440         return;
00441     case 16:
00442         this.StackPointer = ((System.Windows.Controls.TextBox) (target));
00443         return;
00444     case 17:
00445         this.ProgramCounter = ((System.Windows.Controls.TextBox) (target));
00446         return;
00447     case 18:
00448         this.Dissambly = ((System.Windows.Controls.TextBox) (target));
00449         return;
00450     case 19:
00451         this.CycleCount = ((System.Windows.Controls.TextBox) (target));
00452         return;
00453     case 20:
00454         this.XRegisterText = ((System.Windows.Controls.TextBlock) (target));
00455         return;
00456     case 21:
00457         this.YRegisterText = ((System.Windows.Controls.TextBlock) (target));
00458         return;
00459     case 22:
00460         this.StackPointerRegisterText = ((System.Windows.Controls.TextBlock) (target));
00461         return;
00462     case 23:
00463         this.AText = ((System.Windows.Controls.TextBlock) (target));
00464         return;
00465     case 24:
00466         this.CurrentInstructionText = ((System.Windows.Controls.TextBlock) (target));
00467         return;
00468     case 25:
00469         this.ProgramCounterText = ((System.Windows.Controls.TextBlock) (target));
00470         return;
00471     case 26:
00472         this.CycleCountText = ((System.Windows.Controls.TextBlock) (target));
00473         return;
00474     case 27:
00475         this.CarryFlag = ((System.Windows.Controls.CheckBox) (target));
00476         return;
00477     case 28:
00478         this.CarryFlagText = ((System.Windows.Controls.TextBlock) (target));
00479         return;
00480     case 29:
00481         this.ZeroFlag = ((System.Windows.Controls.CheckBox) (target));
00482         return;
00483     case 30:
00484         this.ZeroFlagText = ((System.Windows.Controls.TextBlock) (target));
00485         return;
00486     case 31:
00487         this.InterruptFlag = ((System.Windows.Controls.CheckBox) (target));
00488         return;
00489     case 32:
00490         this.InterruptFlagText = ((System.Windows.Controls.TextBlock) (target));
00491         return;
00492     case 33:
00493         this.BcdFlag = ((System.Windows.Controls.CheckBox) (target));
00494         return;
00495     case 34:
00496         this.BcdFlagText = ((System.Windows.Controls.TextBlock) (target));
00497         return;
00498     case 35:
00499         this.BreakFlag = ((System.Windows.Controls.CheckBox) (target));
00500         return;
00501     case 36:
```



```

00502         this.BreakFlagText = ((System.Windows.Controls.TextBlock) (target));
00503         return;
00504     case 37:
00505         this.OverflowFlag = ((System.Windows.Controls.CheckBox) (target));
00506         return;
00507     case 38:
00508         this.OverflowFlagText = ((System.Windows.Controls.TextBlock) (target));
00509         return;
00510     case 39:
00511         this.NegativeFlag = ((System.Windows.Controls.CheckBox) (target));
00512         return;
00513     case 40:
00514         this.NegativeFlagText = ((System.Windows.Controls.TextBlock) (target));
00515         return;
00516     case 41:
00517         this.CpuSpeed = ((System.Windows.Controls.Slider) (target));
00518         return;
00519     case 42:
00520         this.SpeedText = ((System.Windows.Controls.TextBlock) (target));
00521         return;
00522     }
00523     this._contentLoaded = true;
00524 }

```

6.18.3.5 Connect() [4/6] void System.Windows.Markup.IComponentConnector. Emulator.MainWindow.↔
Connect (

```

    int connectionId,
    object target ) [inline], [private]

```

Definition at line 373 of file [MainWindow.g.i.cs](#).

```

00373                                                                 {
00374         switch (connectionId)
00375         {
00376             case 1:
00377                 this.EmulatorWindow = ((Emulator.MainWindow) (target));
00378                 return;
00379             case 2:
00380
00381 #line 72 "..\..\..\MainWindows.xml"
00382                 ((System.Windows.Controls.MenuItem) (target)).Click += new
00383                     System.Windows.RoutedEventHandler(this.LoadFile);
00384 #line default
00385 #line hidden
00386                 return;
00387             case 3:
00388
00389 #line 73 "..\..\..\MainWindows.xml"
00390                 ((System.Windows.Controls.MenuItem) (target)).Click += new
00391                     System.Windows.RoutedEventHandler(this.SaveFile);
00392 #line default
00393 #line hidden
00394                 return;
00395             case 4:
00396
00397 #line 74 "..\..\..\MainWindows.xml"
00398                 ((System.Windows.Controls.MenuItem) (target)).Click += new
00399                     System.Windows.RoutedEventHandler(this.CloseFile);
00400 #line default
00401 #line hidden
00402                 return;
00403             case 5:
00404
00405 #line 76 "..\..\..\MainWindows.xml"
00406                 ((System.Windows.Controls.MenuItem) (target)).Click += new
00407                     System.Windows.RoutedEventHandler(this.ToClose);
00408 #line default
00409 #line hidden
00410                 return;
00411             case 6:
00412                 this.OutputLog = ((System.Windows.Controls.DataGrid) (target));
00413                 return;
00414             case 7:
00415                 this.Run = ((System.Windows.Controls.Button) (target));
00416                 return;
00417             case 8:

```

```
00418         this.Step = ((System.Windows.Controls.Button) (target));
00419         return;
00420     case 9:
00421         this.Reset = ((System.Windows.Controls.Button) (target));
00422         return;
00423     case 10:
00424         this.RomFileNameText = ((System.Windows.Controls.TextBlock) (target));
00425         return;
00426     case 11:
00427         this.ComPortNameText = ((System.Windows.Controls.TextBlock) (target));
00428         return;
00429     case 12:
00430         this.Breakpoints = ((System.Windows.Controls.DataGrid) (target));
00431         return;
00432     case 13:
00433         this.YRegister = ((System.Windows.Controls.TextBox) (target));
00434         return;
00435     case 14:
00436         this.XRegister = ((System.Windows.Controls.TextBox) (target));
00437         return;
00438     case 15:
00439         this.Accumulator = ((System.Windows.Controls.TextBox) (target));
00440         return;
00441     case 16:
00442         this.StackPointer = ((System.Windows.Controls.TextBox) (target));
00443         return;
00444     case 17:
00445         this.ProgramCounter = ((System.Windows.Controls.TextBox) (target));
00446         return;
00447     case 18:
00448         this.Dissassembly = ((System.Windows.Controls.TextBox) (target));
00449         return;
00450     case 19:
00451         this.CycleCount = ((System.Windows.Controls.TextBox) (target));
00452         return;
00453     case 20:
00454         this.XRegisterText = ((System.Windows.Controls.TextBlock) (target));
00455         return;
00456     case 21:
00457         this.YRegisterText = ((System.Windows.Controls.TextBlock) (target));
00458         return;
00459     case 22:
00460         this.StackPointerRegisterText = ((System.Windows.Controls.TextBlock) (target));
00461         return;
00462     case 23:
00463         this.AText = ((System.Windows.Controls.TextBlock) (target));
00464         return;
00465     case 24:
00466         this.CurrentInstructionText = ((System.Windows.Controls.TextBlock) (target));
00467         return;
00468     case 25:
00469         this.ProgramCounterText = ((System.Windows.Controls.TextBlock) (target));
00470         return;
00471     case 26:
00472         this.CycleCountText = ((System.Windows.Controls.TextBlock) (target));
00473         return;
00474     case 27:
00475         this.CarryFlag = ((System.Windows.Controls.CheckBox) (target));
00476         return;
00477     case 28:
00478         this.CarryFlagText = ((System.Windows.Controls.TextBlock) (target));
00479         return;
00480     case 29:
00481         this.ZeroFlag = ((System.Windows.Controls.CheckBox) (target));
00482         return;
00483     case 30:
00484         this.ZeroFlagText = ((System.Windows.Controls.TextBlock) (target));
00485         return;
00486     case 31:
00487         this.InterruptFlag = ((System.Windows.Controls.CheckBox) (target));
00488         return;
00489     case 32:
00490         this.InterruptFlagText = ((System.Windows.Controls.TextBlock) (target));
00491         return;
00492     case 33:
00493         this.BcdFlag = ((System.Windows.Controls.CheckBox) (target));
00494         return;
00495     case 34:
00496         this.BcdFlagText = ((System.Windows.Controls.TextBlock) (target));
00497         return;
00498     case 35:
00499         this.BreakFlag = ((System.Windows.Controls.CheckBox) (target));
00500         return;
00501     case 36:
00502         this.BreakFlagText = ((System.Windows.Controls.TextBlock) (target));
00503         return;
00504     case 37:
```

```

00505         this.OverflowFlag = ((System.Windows.Controls.CheckBox) (target));
00506         return;
00507         case 38:
00508             this.OverflowFlagText = ((System.Windows.Controls.TextBlock) (target));
00509             return;
00510         case 39:
00511             this.NegativeFlag = ((System.Windows.Controls.CheckBox) (target));
00512             return;
00513         case 40:
00514             this.NegativeFlagText = ((System.Windows.Controls.TextBlock) (target));
00515             return;
00516         case 41:
00517             this.CpuSpeed = ((System.Windows.Controls.Slider) (target));
00518             return;
00519         case 42:
00520             this.SpeedText = ((System.Windows.Controls.TextBlock) (target));
00521             return;
00522         }
00523         this._contentLoaded = true;
00524     }

```

6.18.3.6 Connect() [5/6] void System.Windows.Markup.IComponentConnector. Emulator.MainWindow.↔
Connect (
 int connectionId,
 object target) [inline], [private]

Definition at line 373 of file [MainWindow.g.cs](#).

```

00373                                     {
00374         switch (connectionId)
00375         {
00376             case 1:
00377                 this.EmulatorWindow = ((Emulator.MainWindow) (target));
00378                 return;
00379             case 2:
00380
00381 #line 72 "..\..\..\MainWindows.xml"
00382                 ((System.Windows.Controls.MenuItem) (target)).Click += new
System.Windows.RoutedEventHandler(this.LoadFile);
00383
00384 #line default
00385 #line hidden
00386                 return;
00387             case 3:
00388
00389 #line 73 "..\..\..\MainWindows.xml"
00390                 ((System.Windows.Controls.MenuItem) (target)).Click += new
System.Windows.RoutedEventHandler(this.SaveFile);
00391
00392 #line default
00393 #line hidden
00394                 return;
00395             case 4:
00396
00397 #line 74 "..\..\..\MainWindows.xml"
00398                 ((System.Windows.Controls.MenuItem) (target)).Click += new
System.Windows.RoutedEventHandler(this.CloseFile);
00399
00400 #line default
00401 #line hidden
00402                 return;
00403             case 5:
00404
00405 #line 76 "..\..\..\MainWindows.xml"
00406                 ((System.Windows.Controls.MenuItem) (target)).Click += new
System.Windows.RoutedEventHandler(this.ToClose);
00407
00408 #line default
00409 #line hidden
00410                 return;
00411             case 6:
00412                 this.OutputLog = ((System.Windows.Controls.DataGrid) (target));
00413                 return;
00414             case 7:
00415                 this.Run = ((System.Windows.Controls.Button) (target));
00416                 return;
00417             case 8:
00418                 this.Step = ((System.Windows.Controls.Button) (target));
00419                 return;
00420             case 9:

```

```
00421         this.Reset = ((System.Windows.Controls.Button) (target));
00422         return;
00423         case 10:
00424             this.RomFileNameText = ((System.Windows.Controls.TextBlock) (target));
00425             return;
00426         case 11:
00427             this.ComPortNameText = ((System.Windows.Controls.TextBlock) (target));
00428             return;
00429         case 12:
00430             this.Breakpoints = ((System.Windows.Controls.DataGrid) (target));
00431             return;
00432         case 13:
00433             this.YRegister = ((System.Windows.Controls.TextBox) (target));
00434             return;
00435         case 14:
00436             this.XRegister = ((System.Windows.Controls.TextBox) (target));
00437             return;
00438         case 15:
00439             this.Accumulator = ((System.Windows.Controls.TextBox) (target));
00440             return;
00441         case 16:
00442             this.StackPointer = ((System.Windows.Controls.TextBox) (target));
00443             return;
00444         case 17:
00445             this.ProgramCounter = ((System.Windows.Controls.TextBox) (target));
00446             return;
00447         case 18:
00448             this.Dissassembly = ((System.Windows.Controls.TextBox) (target));
00449             return;
00450         case 19:
00451             this.CycleCount = ((System.Windows.Controls.TextBox) (target));
00452             return;
00453         case 20:
00454             this.XRegisterText = ((System.Windows.Controls.TextBlock) (target));
00455             return;
00456         case 21:
00457             this.YRegisterText = ((System.Windows.Controls.TextBlock) (target));
00458             return;
00459         case 22:
00460             this.StackPointerRegisterText = ((System.Windows.Controls.TextBlock) (target));
00461             return;
00462         case 23:
00463             this.AText = ((System.Windows.Controls.TextBlock) (target));
00464             return;
00465         case 24:
00466             this.CurrentInstructionText = ((System.Windows.Controls.TextBlock) (target));
00467             return;
00468         case 25:
00469             this.ProgramCounterText = ((System.Windows.Controls.TextBlock) (target));
00470             return;
00471         case 26:
00472             this.CycleCountText = ((System.Windows.Controls.TextBlock) (target));
00473             return;
00474         case 27:
00475             this.CarryFlag = ((System.Windows.Controls.CheckBox) (target));
00476             return;
00477         case 28:
00478             this.CarryFlagText = ((System.Windows.Controls.TextBlock) (target));
00479             return;
00480         case 29:
00481             this.ZeroFlag = ((System.Windows.Controls.CheckBox) (target));
00482             return;
00483         case 30:
00484             this.ZeroFlagText = ((System.Windows.Controls.TextBlock) (target));
00485             return;
00486         case 31:
00487             this.InterruptFlag = ((System.Windows.Controls.CheckBox) (target));
00488             return;
00489         case 32:
00490             this.InterruptFlagText = ((System.Windows.Controls.TextBlock) (target));
00491             return;
00492         case 33:
00493             this.BcdFlag = ((System.Windows.Controls.CheckBox) (target));
00494             return;
00495         case 34:
00496             this.BcdFlagText = ((System.Windows.Controls.TextBlock) (target));
00497             return;
00498         case 35:
00499             this.BreakFlag = ((System.Windows.Controls.CheckBox) (target));
00500             return;
00501         case 36:
00502             this.BreakFlagText = ((System.Windows.Controls.TextBlock) (target));
00503             return;
00504         case 37:
00505             this.OverflowFlag = ((System.Windows.Controls.CheckBox) (target));
00506             return;
00507         case 38:
```

```

00508         this.OverflowFlagText = ((System.Windows.Controls.TextBlock) (target));
00509         return;
00510     case 39:
00511         this.NegativeFlag = ((System.Windows.Controls.CheckBox) (target));
00512         return;
00513     case 40:
00514         this.NegativeFlagText = ((System.Windows.Controls.TextBlock) (target));
00515         return;
00516     case 41:
00517         this.CpuSpeed = ((System.Windows.Controls.Slider) (target));
00518         return;
00519     case 42:
00520         this.SpeedText = ((System.Windows.Controls.TextBlock) (target));
00521         return;
00522     }
00523     this._contentLoaded = true;
00524 }

```

6.18.3.7 Connect() [6/6] void System.Windows.Markup.IComponentConnector. Emulator.MainWindow.↔
 Connect (
 int connectionId,
 object target) [inline], [private]

Definition at line 373 of file [MainWindow.g.i.cs](#).

```

00373                                                                 {
00374         switch (connectionId)
00375         {
00376             case 1:
00377                 this.EmulatorWindow = ((Emulator.MainWindow) (target));
00378                 return;
00379             case 2:
00380
00381 #line 72 "..\..\..\MainWindows.xml"
00382                 ((System.Windows.Controls.MenuItem) (target)).Click += new
00383                 System.Windows.RoutedEventHandler(this.LoadFile);
00384 #line default
00385 #line hidden
00386                 return;
00387             case 3:
00388
00389 #line 73 "..\..\..\MainWindows.xml"
00390                 ((System.Windows.Controls.MenuItem) (target)).Click += new
00391                 System.Windows.RoutedEventHandler(this.SaveFile);
00392 #line default
00393 #line hidden
00394                 return;
00395             case 4:
00396
00397 #line 74 "..\..\..\MainWindows.xml"
00398                 ((System.Windows.Controls.MenuItem) (target)).Click += new
00399                 System.Windows.RoutedEventHandler(this.CloseFile);
00400 #line default
00401 #line hidden
00402                 return;
00403             case 5:
00404
00405 #line 76 "..\..\..\MainWindows.xml"
00406                 ((System.Windows.Controls.MenuItem) (target)).Click += new
00407                 System.Windows.RoutedEventHandler(this.ToClose);
00408 #line default
00409 #line hidden
00410                 return;
00411             case 6:
00412                 this.OutputLog = ((System.Windows.Controls.DataGrid) (target));
00413                 return;
00414             case 7:
00415                 this.Run = ((System.Windows.Controls.Button) (target));
00416                 return;
00417             case 8:
00418                 this.Step = ((System.Windows.Controls.Button) (target));
00419                 return;
00420             case 9:
00421                 this.Reset = ((System.Windows.Controls.Button) (target));
00422                 return;
00423             case 10:

```

```
00424         this.RomFileNameText = ((System.Windows.Controls.TextBlock) (target));
00425         return;
00426     case 11:
00427         this.ComPortNameText = ((System.Windows.Controls.TextBlock) (target));
00428         return;
00429     case 12:
00430         this.Breakpoints = ((System.Windows.Controls.DataGrid) (target));
00431         return;
00432     case 13:
00433         this.YRegister = ((System.Windows.Controls.TextBox) (target));
00434         return;
00435     case 14:
00436         this.XRegister = ((System.Windows.Controls.TextBox) (target));
00437         return;
00438     case 15:
00439         this.Accumulator = ((System.Windows.Controls.TextBox) (target));
00440         return;
00441     case 16:
00442         this.StackPointer = ((System.Windows.Controls.TextBox) (target));
00443         return;
00444     case 17:
00445         this.ProgramCounter = ((System.Windows.Controls.TextBox) (target));
00446         return;
00447     case 18:
00448         this.Dissassembly = ((System.Windows.Controls.TextBox) (target));
00449         return;
00450     case 19:
00451         this.CycleCount = ((System.Windows.Controls.TextBox) (target));
00452         return;
00453     case 20:
00454         this.XRegisterText = ((System.Windows.Controls.TextBlock) (target));
00455         return;
00456     case 21:
00457         this.YRegisterText = ((System.Windows.Controls.TextBlock) (target));
00458         return;
00459     case 22:
00460         this.StackPointerRegisterText = ((System.Windows.Controls.TextBlock) (target));
00461         return;
00462     case 23:
00463         this.AText = ((System.Windows.Controls.TextBlock) (target));
00464         return;
00465     case 24:
00466         this.CurrentInstructionText = ((System.Windows.Controls.TextBlock) (target));
00467         return;
00468     case 25:
00469         this.ProgramCounterText = ((System.Windows.Controls.TextBlock) (target));
00470         return;
00471     case 26:
00472         this.CycleCountText = ((System.Windows.Controls.TextBlock) (target));
00473         return;
00474     case 27:
00475         this.CarryFlag = ((System.Windows.Controls.CheckBox) (target));
00476         return;
00477     case 28:
00478         this.CarryFlagText = ((System.Windows.Controls.TextBlock) (target));
00479         return;
00480     case 29:
00481         this.ZeroFlag = ((System.Windows.Controls.CheckBox) (target));
00482         return;
00483     case 30:
00484         this.ZeroFlagText = ((System.Windows.Controls.TextBlock) (target));
00485         return;
00486     case 31:
00487         this.InterruptFlag = ((System.Windows.Controls.CheckBox) (target));
00488         return;
00489     case 32:
00490         this.InterruptFlagText = ((System.Windows.Controls.TextBlock) (target));
00491         return;
00492     case 33:
00493         this.BcdFlag = ((System.Windows.Controls.CheckBox) (target));
00494         return;
00495     case 34:
00496         this.BcdFlagText = ((System.Windows.Controls.TextBlock) (target));
00497         return;
00498     case 35:
00499         this.BreakFlag = ((System.Windows.Controls.CheckBox) (target));
00500         return;
00501     case 36:
00502         this.BreakFlagText = ((System.Windows.Controls.TextBlock) (target));
00503         return;
00504     case 37:
00505         this.OverflowFlag = ((System.Windows.Controls.CheckBox) (target));
00506         return;
00507     case 38:
00508         this.OverflowFlagText = ((System.Windows.Controls.TextBlock) (target));
00509         return;
00510     case 39:
```

```

00511         this.NegativeFlag = ((System.Windows.Controls.CheckBox) (target));
00512         return;
00513     case 40:
00514         this.NegativeFlagText = ((System.Windows.Controls.TextBlock) (target));
00515         return;
00516     case 41:
00517         this.CpuSpeed = ((System.Windows.Controls.Slider) (target));
00518         return;
00519     case 42:
00520         this.SpeedText = ((System.Windows.Controls.TextBlock) (target));
00521         return;
00522     }
00523     this._contentLoaded = true;
00524 }

```

6.18.3.8 InitializeComponent() [1/6] void Emulator.MainWindow.InitializeComponent () [inline]

InitializeComponent

Definition at line 353 of file [MainWindow.g.cs](#).

```

00353         {
00354             if (_contentLoaded) {
00355                 return;
00356             }
00357             _contentLoaded = true;
00358             System.Uri resourceLocater = new System.Uri("/Emulator;component/mainwindow.xaml",
System.UriKind.Relative);
00359
00360 #line 1 "..\..\..\MainWindows\MainWindow.xaml"
00361             System.Windows.Application.LoadComponent(this, resourceLocater);
00362
00363 #line default
00364 #line hidden
00365         }

```

6.18.3.9 InitializeComponent() [2/6] void Emulator.MainWindow.InitializeComponent () [inline]

InitializeComponent

Definition at line 353 of file [MainWindow.g.i.cs](#).

```

00353         {
00354             if (_contentLoaded) {
00355                 return;
00356             }
00357             _contentLoaded = true;
00358             System.Uri resourceLocater = new System.Uri("/Emulator;component/mainwindow.xaml",
System.UriKind.Relative);
00359
00360 #line 1 "..\..\..\MainWindows\MainWindow.xaml"
00361             System.Windows.Application.LoadComponent(this, resourceLocater);
00362
00363 #line default
00364 #line hidden
00365         }

```

6.18.3.10 InitializeComponent() [3/6] void Emulator.MainWindow.InitializeComponent () [inline]

InitializeComponent

Definition at line 353 of file [MainWindow.g.cs](#).

```

00353                                     {
00354             if (_contentLoaded) {
00355                 return;
00356             }
00357             _contentLoaded = true;
00358             System.Uri resourceLocator = new System.Uri("/Emulator;component/mainwindow.xaml",
System.UriKind.Relative);
00359
00360 #line 1 "..\..\..\MainWindow.xaml"
00361             System.Windows.Application.LoadComponent(this, resourceLocator);
00362
00363 #line default
00364 #line hidden
00365         }

```

6.18.3.11 InitializeComponent() [4/6] void Emulator.MainWindow.InitializeComponent () [inline]

InitializeComponent

Definition at line 353 of file [MainWindow.g.i.cs](#).

```

00353                                     {
00354             if (_contentLoaded) {
00355                 return;
00356             }
00357             _contentLoaded = true;
00358             System.Uri resourceLocator = new System.Uri("/Emulator;component/mainwindow.xaml",
System.UriKind.Relative);
00359
00360 #line 1 "..\..\..\MainWindow.xaml"
00361             System.Windows.Application.LoadComponent(this, resourceLocator);
00362
00363 #line default
00364 #line hidden
00365         }

```

6.18.3.12 InitializeComponent() [5/6] void Emulator.MainWindow.InitializeComponent () [inline]

InitializeComponent

Definition at line 353 of file [MainWindow.g.cs](#).

```

00353                                     {
00354             if (_contentLoaded) {
00355                 return;
00356             }
00357             _contentLoaded = true;
00358             System.Uri resourceLocator = new System.Uri("/Emulator;component/mainwindow.xaml",
System.UriKind.Relative);
00359
00360 #line 1 "..\..\..\MainWindow.xaml"
00361             System.Windows.Application.LoadComponent(this, resourceLocator);
00362
00363 #line default
00364 #line hidden
00365         }

```


6.18.3.13 InitializeComponent() [6/6] `void Emulator.MainWindow.InitializeComponent () [inline]`

InitializeComponent

Definition at line 353 of file [MainWindow.g.i.cs](#).

```

00353                                     {
00354             if (_contentLoaded) {
00355                 return;
00356             }
00357             _contentLoaded = true;
00358             System.Uri resourceLocator = new System.Uri("/Emulator;component/mainwindow.xaml",
System.UriKind.Relative);
00359
00360 #line 1 "..\..\..\MainWindows.xaml"
00361             System.Windows.Application.LoadComponent(this, resourceLocator);
00362
00363 #line default
00364 #line hidden
00365         }

```

6.18.3.14 LoadFile() `void Emulator.MainWindow.LoadFile (
Object sender,
EventArgs e) [inline], [private]`Definition at line 26 of file [MainWindow.xaml.cs](#).

```

00027     {
00028         Messenger.Default.Send(new NotificationMessage("LoadFile"));
00029     }

```

6.18.3.15 NotificationMessageReceived() [1/2] `void Emulator.MainWindow.NotificationMessage↵
Received (
NotificationMessage notificationMessage) [inline], [private]`Definition at line 41 of file [MainWindow.xaml.cs](#).

```

00042     {
00043         if (notificationMessage.Notification == "CloseWindow")
00044         {
00045             Close();
00046         }
00047         else if (notificationMessage.Notification == "MemoryVisualWindow")
00048         {
00049             var memoryVisual = new MemoryVisual { DataContext = new MemoryVisualViewModel() };
00050             memoryVisual.Show();
00051         }
00052     }

```

6.18.3.16 NotificationMessageReceived() [2/2] `void Emulator.MainWindow.NotificationMessage↵
Received (
NotificationMessage< SettingsModel > notificationMessage) [inline], [private]`Definition at line 54 of file [MainWindow.xaml.cs](#).

```

00055     {
00056         if (notificationMessage.Notification == "SettingsWindow")
00057         {
00058             var settingsFile = new Settings { DataContext = new
SettingsViewModel(notificationMessage.Content) };
00059             settingsFile.ShowDialog();
00060         }
00061     }

```

6.18.3.17 SaveFile() `void Emulator.MainWindow.SaveFile (`
 `Object sender,`
 `EventArgs e) [inline], [private]`

Definition at line 31 of file [MainWindow.xaml.cs](#).

```
00032     {  
00033         Messenger.Default.Send(new NotificationMessage("SaveState"));  
00034     }
```

6.18.3.18 ToClose() `void Emulator.MainWindow.ToClose (`
 `Object sender,`
 `EventArgs e) [inline], [private]`

Definition at line 21 of file [MainWindow.xaml.cs](#).

```
00022     {  
00023         Close();  
00024     }
```

6.18.4 Member Data Documentation

6.18.4.1 _contentLoaded `bool Emulator.MainWindow._contentLoaded [private]`

Definition at line 346 of file [MainWindow.g.cs](#).

The documentation for this class was generated from the following files:

- Emulator/[MainWindow.xaml.cs](#)
- Emulator/obj/x86/Debug/[MainWindow.g.cs](#)
- Emulator/obj/x86/Debug/[MainWindow.g.i.cs](#)
- Emulator/obj/x86/Publish/[MainWindow.g.cs](#)
- Emulator/obj/x86/Publish/[MainWindow.g.i.cs](#)
- Emulator/obj/x86/Release/[MainWindow.g.cs](#)
- Emulator/obj/x86/Release/[MainWindow.g.i.cs](#)

6.19 Hardware.MemoryMap Class Reference

Classes

- class [BankedRam](#)
- class [BankedRom](#)
- class [DeviceArea](#)
- class [Devices](#)
- class [SharedRom](#)

Static Public Member Functions

- static void [Init](#) ([W65C02](#) processor, [W65C22](#) gpio, [W65C22](#) mm65sib, [W65C51](#) acia, [HM62256](#) bankedRam, [AT28CXX](#) bankedRom, [AT28CXX](#) sharedRom)
- static byte [Read](#) (int address)
Returns the byte at the given address.
- static byte [ReadWithoutCycle](#) (int address)
Returns the byte at the given address without incrementing the cycle count.
- static void [Write](#) (int address, byte data)
Writes data to the given address.
- static void [WriteWithoutCycle](#) (int address, byte data)
Writes data to the given address without incrementing the cycle count.

Static Public Attributes

- static readonly int [Length](#) = 0xFFFF

Properties

- static [W65C02 Processor](#) [get, set]
- static [W65C22 GPIO](#) [get, set]
- static [W65C22 MM65SIB](#) [get, set]
- static [W65C51 ACIA](#) [get, set]
- static [AT28CXX SharedROM](#) [get, set]
- static [AT28CXX BankedROM](#) [get, set]
- static [HM62256 BankedRAM](#) [get, set]

6.19.1 Detailed Description

Definition at line 5 of file [MemoryMap.cs](#).

6.19.2 Member Function Documentation

6.19.2.1 Init() static void Hardware.MemoryMap.Init (
[W65C02](#) processor,
[W65C22](#) gpio,
[W65C22](#) mm65sib,
[W65C51](#) acia,
[HM62256](#) bankedRam,
[AT28CXX](#) bankedRom,
[AT28CXX](#) sharedRom) [inline], [static]

Definition at line 86 of file [MemoryMap.cs](#).

```
00087     {
00088         Processor = processor;
00089         GPIO = gpio;
00090         MM65SIB = mm65sib;
00091         ACIA = acia;
00092         SharedROM = sharedRom;
00093         BankedROM = bankedRom;
00094         BankedRAM = bankedRam;
00095     }
```

6.19.2.2 Read() `static byte Hardware.MemoryMap.Read (`
`int address) [inline], [static]`

Returns the byte at the given address.

Parameters

<i>address</i>	The address to return
----------------	-----------------------

Returns

the byte being returned

Definition at line 102 of file [MemoryMap.cs](#).

```
00103     {
00104         var value = ReadWithoutCycle(address);
00105         Processor.IncrementCycleCount();
00106         return value;
00107     }
```

6.19.2.3 ReadWithoutCycle() static byte Hardware.MemoryMap.ReadWithoutCycle (int address) [inline], [static]

Returns the byte at the given address without incrementing the cycle count.

Parameters

<i>address</i>	The address to return
----------------	-----------------------

Returns

the byte being returned

Definition at line 114 of file [MemoryMap.cs](#).

```
00115     {
00116         if ((ACIA.Offset <= address) && (address <= (ACIA.Offset + ACIA.Length)))
00117         {
00118             return ACIA.Read(address);
00119         }
00120         else if ((GPIO.Offset <= address) && (address <= (GPIO.Offset + GPIO.Length)))
00121         {
00122             return GPIO.Read(address);
00123         }
00124         else if ((MM65SIB.Offset <= address) && (address <= (MM65SIB.Offset + MM65SIB.Length)))
00125         {
00126             return MM65SIB.Read(address);
00127         }
00128         else if ((DeviceArea.Offset <= address) && (address <= DeviceArea.End))
00129         {
00130             return 0x00;
00131         }
00132         else if ((SharedROM.Offset <= address) && (address <= SharedROM.End))
00133         {
00134             return SharedROM.Read(address);
00135         }
00136         else if ((BankedROM.Offset <= address) && (address <= BankedROM.End))
00137         {
00138             return BankedROM.Read(address);
00139         }
00140         else if ((BankedRAM.Offset <= address) && (address <= BankedRAM.End))
00141         {
00142             return BankedRAM.Read(address);
00143         }
00144         else
00145         {
00146             return 0x00;
00147         }
00148     }
```

6.19.2.4 Write() static void Hardware.MemoryMap.Write (
 int address,
 byte data) [inline], [static]

Writes data to the given address.

Parameters

<i>address</i>	The address to write data to.
<i>data</i>	The data to write.

Definition at line 155 of file [MemoryMap.cs](#).

```
00156     {
00157         Processor.IncrementCycleCount();
00158         WriteWithoutCycle(address, data);
00159     }
```

6.19.2.5 WriteWithoutCycle() static void Hardware.MemoryMap.WriteWithoutCycle (
 int address,
 byte data) [inline], [static]

Writes data to the given address without incrementing the cycle count.

Parameters

<i>address</i>	The address to write data to.
<i>data</i>	The data to write.

Definition at line 166 of file [MemoryMap.cs](#).

```
00167     {
00168         if ((ACIA.Offset <= address) && (address <= (ACIA.Offset + ACIA.Length)))
00169         {
00170             ACIA.Write(address, data);
00171         }
00172         else if ((GPIO.Offset <= address) && (address <= (GPIO.Offset + GPIO.Length)))
00173         {
00174             GPIO.Write(address, data);
00175         }
00176         else if ((MM65SIB.Offset <= address) && (address <= (MM65SIB.Offset + MM65SIB.Length)))
00177         {
00178             MM65SIB.Write(address, data);
00179         }
00180         else if ((SharedROM.Offset <= address) && (address <= (SharedROM.Offset +
SharedROM.Length)))
00181         {
00182             SharedROM.Write(address, data);
00183         }
00184         else if ((BankedROM.Offset <= address) && (address <= (BankedROM.Offset +
BankedROM.Length)))
00185         {
00186             BankedROM.Write(address, data);
00187         }
00188         else if ((BankedRAM.Offset <= address) && (address <= (BankedRAM.Offset +
BankedRAM.Length)))
00189         {
00190             BankedRAM.Write(address, data);
00191         }
00192         else
00193         {
00194             throw new ApplicationException(String.Format("Cannot write to address: {0}",
address));
00195         }
00196     }
```

6.19.3 Member Data Documentation

6.19.3.1 Length readonly int Hardware.MemoryMap.Length = 0xFFFF [static]

Definition at line 76 of file [MemoryMap.cs](#).

6.19.4 Property Documentation

6.19.4.1 ACIA [W65C51](#) Hardware.MemoryMap.ACIA [static], [get], [set], [private]

Definition at line 81 of file [MemoryMap.cs](#).

```
00081 { get; set; }
```

6.19.4.2 BankedRAM [HM62256](#) Hardware.MemoryMap.BankedRAM [static], [get], [set], [private]

Definition at line 84 of file [MemoryMap.cs](#).

```
00084 { get; set; }
```

6.19.4.3 BankedROM [AT28CXX](#) Hardware.MemoryMap.BankedROM [static], [get], [set], [private]

Definition at line 83 of file [MemoryMap.cs](#).

```
00083 { get; set; }
```

6.19.4.4 GPIO [W65C22](#) Hardware.MemoryMap.GPIO [static], [get], [set], [private]

Definition at line 79 of file [MemoryMap.cs](#).

```
00079 { get; set; }
```

6.19.4.5 MM65SIB [W65C22](#) Hardware.MemoryMap.MM65SIB [static], [get], [set], [private]

Definition at line 80 of file [MemoryMap.cs](#).

```
00080 { get; set; }
```

6.19.4.6 Processor [W65C02](#) Hardware.MemoryMap.Processor [static], [get], [set], [private]

Definition at line 78 of file [MemoryMap.cs](#).

```
00078 { get; set; }
```

6.19.4.7 SharedROM [AT28CXX](#) Hardware.MemoryMap.SharedROM [static], [get], [set], [private]

Definition at line 82 of file [MemoryMap.cs](#).

```
00082 { get; set; }
```

The documentation for this class was generated from the following file:

- Hardware/Classes/[MemoryMap.cs](#)

6.20 Emulator.Model.MemoryRowModel Class Reference

A [Model](#) of a Single Page of memory

Properties

- string [Offset](#) [get, set]
The offset of this row. Expressed in hex
- string [Location00](#) [get, set]
The memory at the location offset + 00
- string [Location01](#) [get, set]
The memory at the location offset + 01
- string [Location02](#) [get, set]
The memory at the location offset + 02
- string [Location03](#) [get, set]
The memory at the location offset + 03
- string [Location04](#) [get, set]
The memory at the location offset + 04
- string [Location05](#) [get, set]
The memory at the location offset + 05
- string [Location06](#) [get, set]
The memory at the location offset + 06
- string [Location07](#) [get, set]
The memory at the location offset + 07
- string [Location08](#) [get, set]
The memory at the location offset + 08
- string [Location09](#) [get, set]
The memory at the location offset + 09
- string [Location0A](#) [get, set]
The memory at the location offset + 0A
- string [Location0B](#) [get, set]
The memory at the location offset + 0B
- string [Location0C](#) [get, set]
The memory at the location offset + 0C
- string [Location0D](#) [get, set]
The memory at the location offset + 0D
- string [Location0E](#) [get, set]
The memory at the location offset + 0E
- string [Location0F](#) [get, set]
The memory at the location offset + 0F

6.20.1 Detailed Description

A [Model](#) of a Single Page of memory

Definition at line 6 of file [MemoryRowModel.cs](#).

6.20.2 Property Documentation

6.20.2.1 Location00 `string Emulator.Model.MemoryRowModel.Location00 [get], [set]`

The memory at the location offset + 00

Definition at line 15 of file [MemoryRowModel.cs](#).

```
00015 { get; set; }
```

6.20.2.2 Location01 `string Emulator.Model.MemoryRowModel.Location01 [get], [set]`

The memory at the location offset + 01

Definition at line 19 of file [MemoryRowModel.cs](#).

```
00019 { get; set; }
```

6.20.2.3 Location02 `string Emulator.Model.MemoryRowModel.Location02 [get], [set]`

The memory at the location offset + 02

Definition at line 23 of file [MemoryRowModel.cs](#).

```
00023 { get; set; }
```

6.20.2.4 Location03 `string Emulator.Model.MemoryRowModel.Location03 [get], [set]`

The memory at the location offset + 03

Definition at line 27 of file [MemoryRowModel.cs](#).

```
00027 { get; set; }
```

6.20.2.5 Location04 `string Emulator.Model.MemoryRowModel.Location04 [get], [set]`

The memory at the location offset + 04

Definition at line 31 of file [MemoryRowModel.cs](#).

```
00031 { get; set; }
```

6.20.2.6 Location05 `string Emulator.Model.MemoryRowModel.Location05 [get], [set]`

The memory at the location offset + 05

Definition at line 35 of file [MemoryRowModel.cs](#).

```
00035 { get; set; }
```

6.20.2.7 Location06 `string Emulator.Model.MemoryRowModel.Location06 [get], [set]`

The memory at the location offset + 06

Definition at line 39 of file [MemoryRowModel.cs](#).

```
00039 { get; set; }
```

6.20.2.8 Location07 `string Emulator.Model.MemoryRowModel.Location07 [get], [set]`

The memory at the location offset + 07

Definition at line 43 of file [MemoryRowModel.cs](#).

```
00043 { get; set; }
```

6.20.2.9 Location08 `string Emulator.Model.MemoryRowModel.Location08 [get], [set]`

The memory at the location offset + 08

Definition at line 47 of file [MemoryRowModel.cs](#).

```
00047 { get; set; }
```

6.20.2.10 Location09 `string Emulator.Model.MemoryRowModel.Location09 [get], [set]`

The memory at the location offset + 09

Definition at line 51 of file [MemoryRowModel.cs](#).

```
00051 { get; set; }
```

6.20.2.11 Location0A `string Emulator.Model.MemoryRowModel.Location0A [get], [set]`

The memory at the location offset + 0A

Definition at line 55 of file [MemoryRowModel.cs](#).

```
00055 { get; set; }
```

6.20.2.12 Location0B `string Emulator.Model.MemoryRowModel.Location0B [get], [set]`

The memory at the location offset + 0B

Definition at line 59 of file [MemoryRowModel.cs](#).

```
00059 { get; set; }
```

6.20.2.13 Location0C `string Emulator.Model.MemoryRowModel.Location0C [get], [set]`

The memory at the location offset + 0C

Definition at line 63 of file [MemoryRowModel.cs](#).

```
00063 { get; set; }
```

6.20.2.14 Location0D `string Emulator.Model.MemoryRowModel.Location0D [get], [set]`

The memory at the location offset + 0D

Definition at line 67 of file [MemoryRowModel.cs](#).

```
00067 { get; set; }
```

6.20.2.15 Location0E `string Emulator.Model.MemoryRowModel.Location0E [get], [set]`

The memory at the location offset + 0E

Definition at line 71 of file [MemoryRowModel.cs](#).

```
00071 { get; set; }
```

6.20.2.16 Location0F `string Emulator.Model.MemoryRowModel.Location0F [get], [set]`

The memory at the location offset + 0F

Definition at line 75 of file [MemoryRowModel.cs](#).

```
00075 { get; set; }
```

6.20.2.17 Offset `string Emulator.Model.MemoryRowModel.Offset [get], [set]`

The offset of this row. Expressed in hex

Definition at line 11 of file [MemoryRowModel.cs](#).

```
00011 { get; set; }
```

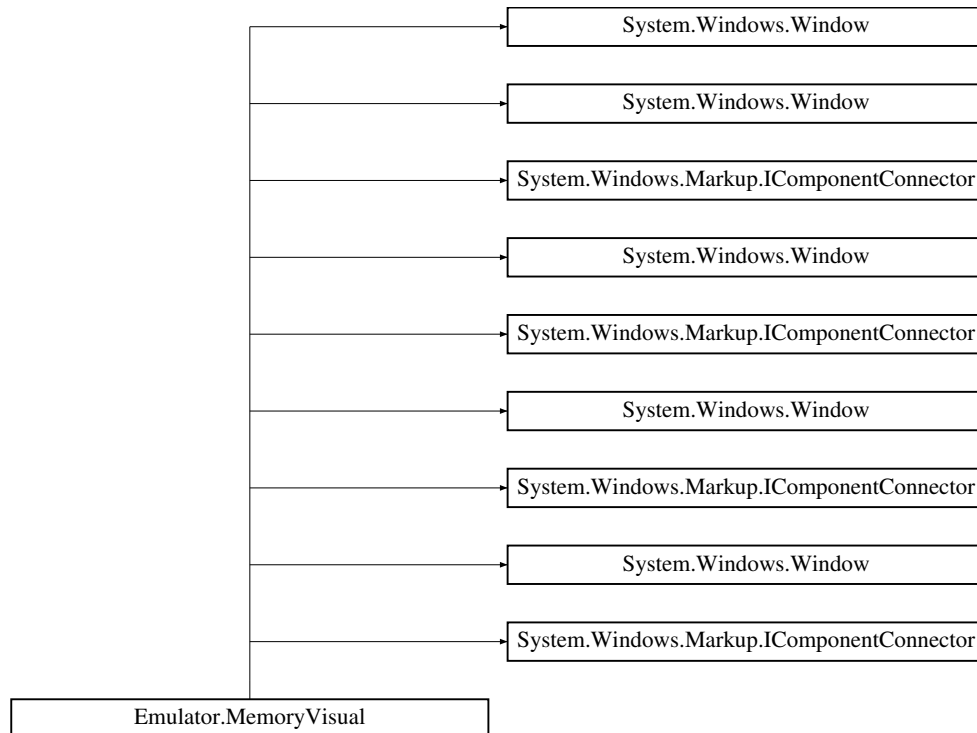
The documentation for this class was generated from the following file:

- [Emulator/Model/MemoryRowModel.cs](#)

6.21 Emulator.MemoryVisual Class Reference

Interaction logic for Window1.xaml

Inheritance diagram for Emulator.MemoryVisual:



Public Member Functions

- [MemoryVisual](#) ()
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent

Private Member Functions

- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)

Private Attributes

- bool [_contentLoaded](#)

6.21.1 Detailed Description

Interaction logic for Window1.xaml

[MemoryVisual](#)

Definition at line 8 of file [MemoryVisual.xaml.cs](#).

6.21.2 Constructor & Destructor Documentation

6.21.2.1 MemoryVisual() `Emulator.MemoryVisual.MemoryVisual () [inline]`

Definition at line 10 of file [MemoryVisual.xaml.cs](#).

```
00011     {
00012         InitializeComponent();
00013     }
```

6.21.3 Member Function Documentation

6.21.3.1 Connect() [1/4] `void System.Windows.Markup.IComponentConnector. Emulator.Memory↔Visual.Connect (int connectionId, object target) [inline], [private]`

Definition at line 93 of file [MemoryVisual.g.cs](#).

```
00093     {
00094         switch (connectionId)
00095         {
00096             case 1:
00097                 this.MemoryMap = ((System.Windows.Controls.DataGrid) (target));
00098                 return;
00099             case 2:
00100                 this.CurrentPage = ((System.Windows.Controls.TextBox) (target));
00101                 return;
00102             case 3:
00103                 this.CurrentPageText = ((System.Windows.Controls.TextBlock) (target));
00104                 return;
00105             }
00106             this._contentLoaded = true;
00107         }
```

6.21.3.2 Connect() [2/4] `void System.Windows.Markup.IComponentConnector. Emulator.Memory↔Visual.Connect (int connectionId, object target) [inline], [private]`

Definition at line 93 of file [MemoryVisual.g.i.cs](#).

```
00093     {
00094         switch (connectionId)
00095         {
00096             case 1:
00097                 this.MemoryMap = ((System.Windows.Controls.DataGrid) (target));
00098                 return;
00099             case 2:
00100                 this.CurrentPage = ((System.Windows.Controls.TextBox) (target));
00101                 return;
00102             case 3:
00103                 this.CurrentPageText = ((System.Windows.Controls.TextBlock) (target));
00104                 return;
00105             }
00106             this._contentLoaded = true;
00107         }
```

6.21.3.3 Connect() [3/4] void System.Windows.Markup.IComponentConnector. Emulator.Memory↔
 Visual.Connect (
 int *connectionId*,
 object *target*) [inline], [private]

Definition at line 93 of file [MemoryVisual.g.cs](#).

```
00093                                     {
00094         switch (connectionId)
00095         {
00096         case 1:
00097             this.MemoryMap = ((System.Windows.Controls.DataGrid) (target));
00098             return;
00099         case 2:
00100             this.CurrentPage = ((System.Windows.Controls.TextBox) (target));
00101             return;
00102         case 3:
00103             this.CurrentPageText = ((System.Windows.Controls.TextBlock) (target));
00104             return;
00105         }
00106         this._contentLoaded = true;
00107     }
```

6.21.3.4 Connect() [4/4] void System.Windows.Markup.IComponentConnector. Emulator.Memory↔
 Visual.Connect (
 int *connectionId*,
 object *target*) [inline], [private]

Definition at line 93 of file [MemoryVisual.g.i.cs](#).

```
00093                                     {
00094         switch (connectionId)
00095         {
00096         case 1:
00097             this.MemoryMap = ((System.Windows.Controls.DataGrid) (target));
00098             return;
00099         case 2:
00100             this.CurrentPage = ((System.Windows.Controls.TextBox) (target));
00101             return;
00102         case 3:
00103             this.CurrentPageText = ((System.Windows.Controls.TextBlock) (target));
00104             return;
00105         }
00106         this._contentLoaded = true;
00107     }
```

6.21.3.5 InitializeComponent() [1/4] void Emulator.MemoryVisual.InitializeComponent () [inline]

InitializeComponent

Definition at line 73 of file [MemoryVisual.g.cs](#).

```
00073                                     {
00074         if (!_contentLoaded) {
00075             return;
00076         }
00077         _contentLoaded = true;
00078         System.Uri resourceLocator = new System.Uri("/Emulator;component/memoryvisual.xaml",
00079             System.UriKind.Relative);
00079
00080 #line 1 "..\..\..\MemoryVisual.xaml"
00081         System.Windows.Application.LoadComponent(this, resourceLocator);
00082
00083 #line default
00084 #line hidden
00085     }
```

6.21.3.6 InitializeComponent() [2/4] void Emulator.MemoryVisual.InitializeComponent () [inline]

InitializeComponent

Definition at line 73 of file [MemoryVisual.g.i.cs](#).

```

00073                                     {
00074             if (_contentLoaded) {
00075                 return;
00076             }
00077             _contentLoaded = true;
00078             System.Uri resourceLocator = new System.Uri("/Emulator;component/memoryvisual.xaml",
System.UriKind.Relative);
00079
00080 #line 1 "..\\..\\..\\MemoryVisual.xaml"
00081             System.Windows.Application.LoadComponent(this, resourceLocator);
00082
00083 #line default
00084 #line hidden
00085         }

```

6.21.3.7 InitializeComponent() [3/4] void Emulator.MemoryVisual.InitializeComponent () [inline]

InitializeComponent

Definition at line 73 of file [MemoryVisual.g.cs](#).

```

00073                                     {
00074             if (_contentLoaded) {
00075                 return;
00076             }
00077             _contentLoaded = true;
00078             System.Uri resourceLocator = new System.Uri("/Emulator;component/memoryvisual.xaml",
System.UriKind.Relative);
00079
00080 #line 1 "..\\..\\..\\MemoryVisual.xaml"
00081             System.Windows.Application.LoadComponent(this, resourceLocator);
00082
00083 #line default
00084 #line hidden
00085         }

```

6.21.3.8 InitializeComponent() [4/4] void Emulator.MemoryVisual.InitializeComponent () [inline]

InitializeComponent

Definition at line 73 of file [MemoryVisual.g.i.cs](#).

```

00073                                     {
00074             if (_contentLoaded) {
00075                 return;
00076             }
00077             _contentLoaded = true;
00078             System.Uri resourceLocator = new System.Uri("/Emulator;component/memoryvisual.xaml",
System.UriKind.Relative);
00079
00080 #line 1 "..\\..\\..\\MemoryVisual.xaml"
00081             System.Windows.Application.LoadComponent(this, resourceLocator);
00082
00083 #line default
00084 #line hidden
00085         }

```

6.21.4 Member Data Documentation

6.21.4.1 `_contentLoaded` `bool` `Emulator.MemoryVisual._contentLoaded` `[private]`

Definition at line 66 of file [MemoryVisual.g.cs](#).

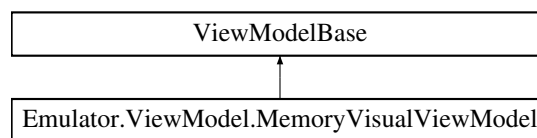
The documentation for this class was generated from the following files:

- [Emulator/MemoryVisual.xaml.cs](#)
- [Emulator/obj/x86/Debug/MemoryVisual.g.cs](#)
- [Emulator/obj/x86/Debug/MemoryVisual.g.i.cs](#)
- [Emulator/obj/x86/Release/MemoryVisual.g.cs](#)
- [Emulator/obj/x86/Release/MemoryVisual.g.i.cs](#)

6.22 Emulator.ViewModel.MemoryVisualViewModel Class Reference

The Main [ViewModel](#)

Inheritance diagram for `Emulator.ViewModel.MemoryVisualViewModel`:



Public Member Functions

- [MemoryVisualViewModel](#) ()
Creates a new Instance of the [MemoryVisualViewModel](#).
- void [UpdateMemoryPage](#) ()

Properties

- [MultiThreadedObservableCollection< MemoryRowModel > MemoryPage](#) `[get, set]`
The Current Memory Page
- string [MemoryPageOffset](#) `[get, set]`
The Memory Page number.
- RelayCommand [UpdateMemoryMapCommand](#) `[get, set]`
Relay Command that updates the Memory Map when the Page changes

Private Member Functions

- void [GenericNotification](#) (NotificationMessage notificationMessage)
- void [UpdateUi](#) ()

Private Attributes

- int [_memoryPageOffset](#)

6.22.1 Detailed Description

The Main [ViewModel](#)

Definition at line 13 of file [MemoryVisualViewModel.cs](#).

6.22.2 Constructor & Destructor Documentation

6.22.2.1 MemoryVisualViewModel() `Emulator.ViewModel.MemoryVisualViewModel.MemoryVisualView↔
Model () [inline]`

Creates a new Instance of the [MemoryVisualViewModel](#).

Definition at line 53 of file [MemoryVisualViewModel.cs](#).

```
00054     {  
00055         UpdateMemoryMapCommand = new RelayCommand(UpdateMemoryPage);  
00056  
00057         Messenger.Default.Register<NotificationMessage>(this, GenericNotification);  
00058  
00059         MemoryPage = new MultiThreadedObservableCollection<MemoryRowModel>();  
00060  
00061         UpdateMemoryPage();  
00062         UpdateUi();  
00063     }
```

6.22.3 Member Function Documentation

6.22.3.1 GenericNotification() `void Emulator.ViewModel.MemoryVisualViewModel.GenericNotification
(
 NotificationMessage notificationMessage) [inline], [private]`

Definition at line 65 of file [MemoryVisualViewModel.cs](#).

```
00066     {  
00067         if (notificationMessage.Notification == "UpdateMemoryPage")  
00068         {  
00069             UpdateMemoryPage();  
00070             UpdateUi();  
00071         }  
00072     }
```

6.22.3.2 UpdateMemoryPage() void Emulator.ViewModel.MemoryVisualViewModel.UpdateMemoryPage () [inline]

Definition at line 74 of file [MemoryVisualViewModel.cs](#).

```

00075     {
00076         MemoryPage.Clear();
00077         var offset = _memoryPageOffset * 256;
00078
00079         var multiplier = 0;
00080         for (ushort i = (ushort)offset; i < 256 * (_memoryPageOffset + 1); i++)
00081         {
00082             MemoryPage.Add(new MemoryRowModel
00083             {
00084                 Offset = ((16 * multiplier) + offset).ToString("X").PadLeft(4, '0'),
00085                 Location00 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00086                 Location01 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00087                 Location02 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00088                 Location03 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00089                 Location04 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00090                 Location05 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00091                 Location06 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00092                 Location07 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00093                 Location08 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00094                 Location09 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00095                 Location0A = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00096                 Location0B = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00097                 Location0C = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00098                 Location0D = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00099                 Location0E = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00100                 Location0F = MemoryMap.ReadWithoutCycle(i).ToString("X").PadLeft(2, '0'),
00101             });
00102             multiplier++;
00103         }
00104     }

```

6.22.3.3 UpdateUi() void Emulator.ViewModel.MemoryVisualViewModel.UpdateUi () [inline], [private]

Definition at line 108 of file [MemoryVisualViewModel.cs](#).

```

00109     {
00110         RaisePropertyChanged("MemoryPage");
00111     }

```

6.22.4 Member Data Documentation

6.22.4.1 _memoryPageOffset int Emulator.ViewModel.MemoryVisualViewModel._memoryPageOffset [private]

Definition at line 16 of file [MemoryVisualViewModel.cs](#).

6.22.5 Property Documentation

6.22.5.1 MemoryPage [MultiThreadedObservableCollection<MemoryRowModel>](#) Emulator.ViewModel.↔
MemoryVisualViewModel.MemoryPage [get], [set]

The Current Memory Page

Definition at line 23 of file [MemoryVisualViewModel.cs](#).

```
00023 { get; set; }
```

6.22.5.2 MemoryPageOffset `string Emulator.ViewModel.MemoryVisualViewModel.MemoryPageOffset`
`[get], [set]`

The Memory Page number.

Definition at line 28 of file [MemoryVisualViewModel.cs](#).

```
00029     {
00030         get { return _memoryPageOffset.ToString("X"); }
00031         set
00032         {
00033             if (string.IsNullOrEmpty(value))
00034                 return;
00035             try
00036             {
00037                 _memoryPageOffset = Convert.ToInt32(value, 16);
00038             }
00039             catch { }
00040         }
00041     }
```

6.22.5.3 UpdateMemoryMapCommand `RelayCommand Emulator.ViewModel.MemoryVisualViewModel.↔`
`UpdateMemoryMapCommand [get], [set]`

Relay Command that updates the Memory Map when the Page changes

Definition at line 46 of file [MemoryVisualViewModel.cs](#).

```
00046 { get; set; }
```

The documentation for this class was generated from the following file:

- [Emulator/ViewModel/MemoryVisualViewModel.cs](#)

6.23 Hardware.MemoryMap.Devices.MM65SIB Class Reference

Static Public Attributes

- static int [Length](#) = 0x0F
- static byte [Offset](#) = 0x30

6.23.1 Detailed Description

Definition at line 69 of file [MemoryMap.cs](#).

6.23.2 Member Data Documentation

6.23.2.1 Length `int Hardware.MemoryMap.Devices.MM65SIB.Length = 0x0F [static]`

Definition at line 71 of file [MemoryMap.cs](#).

6.23.2.2 Offset `byte Hardware.MemoryMap.Devices.MM65SIB.Offset = 0x30 [static]`

Definition at line 72 of file [MemoryMap.cs](#).

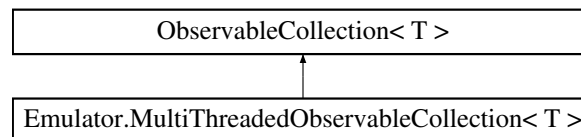
The documentation for this class was generated from the following file:

- [Hardware/Classes/MemoryMap.cs](#)

6.24 Emulator.MultiThreadedObservableCollection< T > Class Template Reference

A MultiThreadedObservableCollection. This allows multiple threads to access the same observable collection in a safe manner.

Inheritance diagram for Emulator.MultiThreadedObservableCollection< T >:



Public Member Functions

- [MultiThreadedObservableCollection](#) ()
Instantiates a new instance of the [MultiThreadedObservableCollection](#)
- [MultiThreadedObservableCollection](#) (IEnumerable< T > collection)
Instantiates a new instance of the [MultiThreadedObservableCollection](#)
- [MultiThreadedObservableCollection](#) (List< T > list)
Instantiates a new instance of the [MultiThreadedObservableCollection](#)

Protected Member Functions

- override void [OnCollectionChanged](#) (NotifyCollectionChangedEventArgs e)
The NotifyCollectionChangedEventHandler, Notifies the listeners in a thread safe manner

Events

- override NotifyCollectionChangedEventHandler [CollectionChanged](#)
The NotifyCollectionChangedEventHandler, Sends a notification anytime the collection has been modified.

6.24.1 Detailed Description

A MultiThreadedObservableCollection. This allows multiple threads to access the same observable collection in a safe manner.

Template Parameters

<i>T</i>	
----------	--

Definition at line 14 of file [MultiThreadedCollection.cs](#).

6.24.2 Constructor & Destructor Documentation

6.24.2.1 MultiThreadedObservableCollection() [1/3] [Emulator.MultiThreadedObservableCollection< T >.MultiThreadedObservableCollection \(\)](#) [inline]

Instantiates a new instance of the [MultiThreadedObservableCollection](#)

Definition at line 19 of file [MultiThreadedCollection.cs](#).

```
00020     {  
00021  
00022     }
```

6.24.2.2 MultiThreadedObservableCollection() [2/3] [Emulator.MultiThreadedObservableCollection< T >.MultiThreadedObservableCollection \(IEnumerable< T > collection \)](#) [inline]

Instantiates a new instance of the [MultiThreadedObservableCollection](#)

Parameters

<i>collection</i>	The initial collection to be loaded
-------------------	-------------------------------------

Definition at line 28 of file [MultiThreadedCollection.cs](#).

```
00029         : base(collection)  
00030     {  
00031  
00032     }
```

6.24.2.3 MultiThreadedObservableCollection() [3/3] [Emulator.MultiThreadedObservableCollection< T >.MultiThreadedObservableCollection \(List< T > list \)](#) [inline]

Instantiates a new instance of the [MultiThreadedObservableCollection](#)

Parameters

<i>list</i>	The initial list to be loaded
-------------	-------------------------------

Definition at line 38 of file [MultiThreadedCollection.cs](#).

```
00039         : base(list)  
00040     {  
00041  
00042     }
```

6.24.3 Member Function Documentation

6.24.3.1 OnCollectionChanged() override void [Emulator.MultiThreadedObservableCollection](#)< T >.OnCollectionChanged (NotifyCollectionChangedEventArgs e) [inline], [protected]

The NotifyCollectionChangedEventHandler, Notifies the listeners in a thread safe manner

Definition at line 53 of file [MultiThreadedCollection.cs](#).

```

00054     {
00055         var collectionChanged = CollectionChanged;
00056         if (collectionChanged != null)
00057             foreach (NotifyCollectionChangedEventHandler nh in
collectionChanged.GetInvocationList())
00058             {
00059                 var dispObj = nh.Target as DispatcherObject;
00060                 if (dispObj != null)
00061                 {
00062                     var dispatcher = dispObj.Dispatcher;
00063                     if (dispatcher != null && !dispatcher.CheckAccess())
00064                     {
00065                         var nh1 = nh;
00066                         dispatcher.BeginInvoke(
00067                             (Action) (() => nh1.Invoke(this,
00068                                 new
NotifyCollectionChangedEventArgs(NotifyCollectionChangedAction.Reset))),
00069                             DispatcherPriority.DataBind);
00070                         continue;
00071                     }
00072                 }
00073                 nh.Invoke(this, e);
00074             }
00075     }

```

6.24.4 Event Documentation

6.24.4.1 CollectionChanged override NotifyCollectionChangedEventHandler [Emulator.MultiThreadedObservableCollection](#)< T >.CollectionChanged

The NotifyCollectionChangedEventHandler, Sends a notification anytime the collection has been modified.

Definition at line 47 of file [MultiThreadedCollection.cs](#).

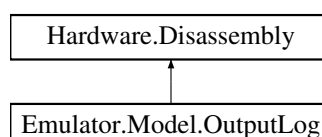
The documentation for this class was generated from the following file:

- [Emulator/MultiThreadedCollection.cs](#)

6.25 Emulator.Model.OutputLog Class Reference

The [OutputLog Model](#). Used by the outputlog grid to show a history of operations performed by the CPU

Inheritance diagram for Emulator.Model.OutputLog:



Public Member Functions

- [OutputLog](#) ([Disassembly](#) disassembly)

Properties

- string [ProgramCounter](#) [get, set]
The Program Counter Value
- string [CurrentOpCode](#) [get, set]
The Current Ope Code
- string [XRegister](#) [get, set]
The X Register
- string [YRegister](#) [get, set]
The Y Register
- string [Accumulator](#) [get, set]
The Accumulator
- string [StackPointer](#) [get, set]
The Stack Pointer
- int [NumberOfCycles](#) [get, set]
The number of cycles executed since the last load or reset

6.25.1 Detailed Description

The [OutputLog Model](#). Used by the outputlog grid to show a history of operations performed by the CPU

Definition at line 10 of file [OutputLog.cs](#).

6.25.2 Constructor & Destructor Documentation

6.25.2.1 [OutputLog\(\)](#) `Emulator.Model.OutputLog.OutputLog (
 Disassembly disassembly) [inline]`

Definition at line 12 of file [OutputLog.cs](#).

```
00013     {  
00014         DisassemblyOutput = disassembly.DisassemblyOutput;  
00015         HighAddress = disassembly.HighAddress;  
00016         LowAddress = disassembly.LowAddress;  
00017         OpCodeString = disassembly.OpCodeString;  
00018     }
```

6.25.3 Property Documentation

6.25.3.1 Accumulator `string Emulator.Model.OutputLog.Accumulator [get], [set]`

The Accumulator

Definition at line 39 of file [OutputLog.cs](#).

```
00039 { get; set; }
```

6.25.3.2 CurrentOpCode `string Emulator.Model.OutputLog.CurrentOpCode [get], [set]`

The Current Ope Code

Definition at line 27 of file [OutputLog.cs](#).

```
00027 { get; set; }
```

6.25.3.3 NumberOfCycles `int Emulator.Model.OutputLog.NumberOfCycles [get], [set]`

The number of cycles executed since the last load or reset

Definition at line 47 of file [OutputLog.cs](#).

```
00047 { get; set; }
```

6.25.3.4 ProgramCounter `string Emulator.Model.OutputLog.ProgramCounter [get], [set]`

The Program Counter Value

Definition at line 23 of file [OutputLog.cs](#).

```
00023 { get; set; }
```

6.25.3.5 StackPointer `string Emulator.Model.OutputLog.StackPointer [get], [set]`

The Stack Pointer

Definition at line 43 of file [OutputLog.cs](#).

```
00043 { get; set; }
```

6.25.3.6 XRegister `string Emulator.Model.OutputLog.XRegister [get], [set]`

The X Register

Definition at line 31 of file [OutputLog.cs](#).

```
00031 { get; set; }
```


6.25.3.7 YRegister `string Emulator.Model.OutputLog.YRegister [get], [set]`

The Y Register

Definition at line 35 of file [OutputLog.cs](#).

```
00035 { get; set; }
```

The documentation for this class was generated from the following file:

- [Emulator/Model/OutputLog.cs](#)

6.26 Emulator.Versioning.Product Class Reference

Static Public Attributes

- const int [Major](#) = 0
- const int [Minor](#) = 1
- const int [Build](#) = 3
- const int [Revision](#) = 1
- const string [Title](#) = [Name](#)
- const string [Name](#) = "WolfNet 65C02 WorkBench Computer Emulator"
- const string [Company](#) = "WolfNet Computing"
- const string [Copyright](#) = "Copyright f WolfNet Computing 2022"
- const string [VersionString](#) = "0.2.4.1"
- const string [Description](#) = "Emulator for the WolfNet 65C02 WorkBench Computer coded in C# using the .NET Framework"

6.26.1 Detailed Description

Definition at line 5 of file [Versioning.cs](#).

6.26.2 Member Data Documentation

6.26.2.1 Build `const int Emulator.Versioning.Product.Build = 3 [static]`

Definition at line 9 of file [Versioning.cs](#).

6.26.2.2 Company `const string Emulator.Versioning.Product.Company = "WolfNet Computing" [static]`

Definition at line 13 of file [Versioning.cs](#).

6.26.2.3 Copyright `const string Emulator.Versioning.Product.Copyright = "Copyright 1' WolfNet Computing 2022" [static]`

Definition at line 14 of file [Versioning.cs](#).

6.26.2.4 Description `const string Emulator.Versioning.Product.Description = "Emulator for the WolfNet 65C02 WorkBench Computer coded in C# using the .NET Framework" [static]`

Definition at line 16 of file [Versioning.cs](#).

6.26.2.5 Major `const int Emulator.Versioning.Product.Major = 0 [static]`

Definition at line 7 of file [Versioning.cs](#).

6.26.2.6 Minor `const int Emulator.Versioning.Product.Minor = 1 [static]`

Definition at line 8 of file [Versioning.cs](#).

6.26.2.7 Name `const string Emulator.Versioning.Product.Name = "WolfNet 65C02 WorkBench Computer Emulator" [static]`

Definition at line 12 of file [Versioning.cs](#).

6.26.2.8 Revision `const int Emulator.Versioning.Product.Revision = 1 [static]`

Definition at line 10 of file [Versioning.cs](#).

6.26.2.9 Title `const string Emulator.Versioning.Product.Title = Name [static]`

Definition at line 11 of file [Versioning.cs](#).

6.26.2.10 VersionString `const string Emulator.Versioning.Product.VersionString = "0.2.4.1" [static]`

Definition at line 15 of file [Versioning.cs](#).

The documentation for this class was generated from the following file:

- [Emulator/Classes/Versioning.cs](#)

6.27 Hardware.Versioning.Product Class Reference

Static Public Attributes

- const string [Title](#) = [Name](#)
- const string [Name](#) = "WolfNet 65C02 Hardware Library"
- const string [Company](#) = "WolfNet Computing"
- const string [Copyright](#) = "Copyright I WolfNet Computing 2022"
- const string [Version](#) = "1.3.0.0"
- const string [Description](#) = "65C02 Hardware Library, coded in C# using the .NET Framework"

6.27.1 Detailed Description

Definition at line 5 of file [Versioning.cs](#).

6.27.2 Member Data Documentation

6.27.2.1 Company `const string Hardware.Versioning.Product.Company = "WolfNet Computing"`
[static]

Definition at line 9 of file [Versioning.cs](#).

6.27.2.2 Copyright `const string Hardware.Versioning.Product.Copyright = "Copyright I' WolfNet Computing 2022"` [static]

Definition at line 10 of file [Versioning.cs](#).

6.27.2.3 Description `const string Hardware.Versioning.Product.Description = "65C02 Hardware Library, coded in C# using the .NET Framework"` [static]

Definition at line 12 of file [Versioning.cs](#).

6.27.2.4 Name `const string Hardware.Versioning.Product.Name = "WolfNet 65C02 Hardware Library"`
[static]

Definition at line 8 of file [Versioning.cs](#).

6.27.2.5 Title `const string Hardware.Versioning.Product.Title = Name [static]`

Definition at line 7 of file [Versioning.cs](#).

6.27.2.6 Version `const string Hardware.Versioning.Product.Version = "1.3.0.0" [static]`

Definition at line 11 of file [Versioning.cs](#).

The documentation for this class was generated from the following file:

- Hardware/Classes/[Versioning.cs](#)

6.28 Emulator.Model.RomFileModel Class Reference

The [Model](#) used when Loading a Program.

Properties

- `byte[][] Rom [get, set]`
The Program Converted into Hex.
- `byte RomBanks [get, set]`
The path of the Program that was loaded.
- `int RomBankSize [get, set]`
The name of the Program that was loaded.
- `string RomFileName [get, set]`
The name of the Program that was loaded.
- `string RomFilePath [get, set]`
The path of the Program that was loaded.

6.28.1 Detailed Description

The [Model](#) used when Loading a Program.

Definition at line 6 of file [RomFileModel.cs](#).

6.28.2 Property Documentation

6.28.2.1 Rom `byte [][] Emulator.Model.RomFileModel.Rom [get], [set]`

The Program Converted into Hex.

Definition at line 11 of file [RomFileModel.cs](#).

```
00011 { get; set; }
```

6.28.2.2 RomBanks `byte Emulator.Model.RomFileModel.RomBanks [get], [set]`

The path of the Program that was loaded.

Definition at line 16 of file [RomFileModel.cs](#).

```
00016 { get; set; }
```

6.28.2.3 RomBankSize `int Emulator.Model.RomFileModel.RomBankSize [get], [set]`

The name of the Program that was loaded.

Definition at line 21 of file [RomFileModel.cs](#).

```
00021 { get; set; }
```

6.28.2.4 RomFileName `string Emulator.Model.RomFileModel.RomFileName [get], [set]`

The name of the Program that was loaded.

Definition at line 26 of file [RomFileModel.cs](#).

```
00026 { get; set; }
```

6.28.2.5 RomFilePath `string Emulator.Model.RomFileModel.RomFilePath [get], [set]`

The path of the Program that was loaded.

Definition at line 31 of file [RomFileModel.cs](#).

```
00031 { get; set; }
```

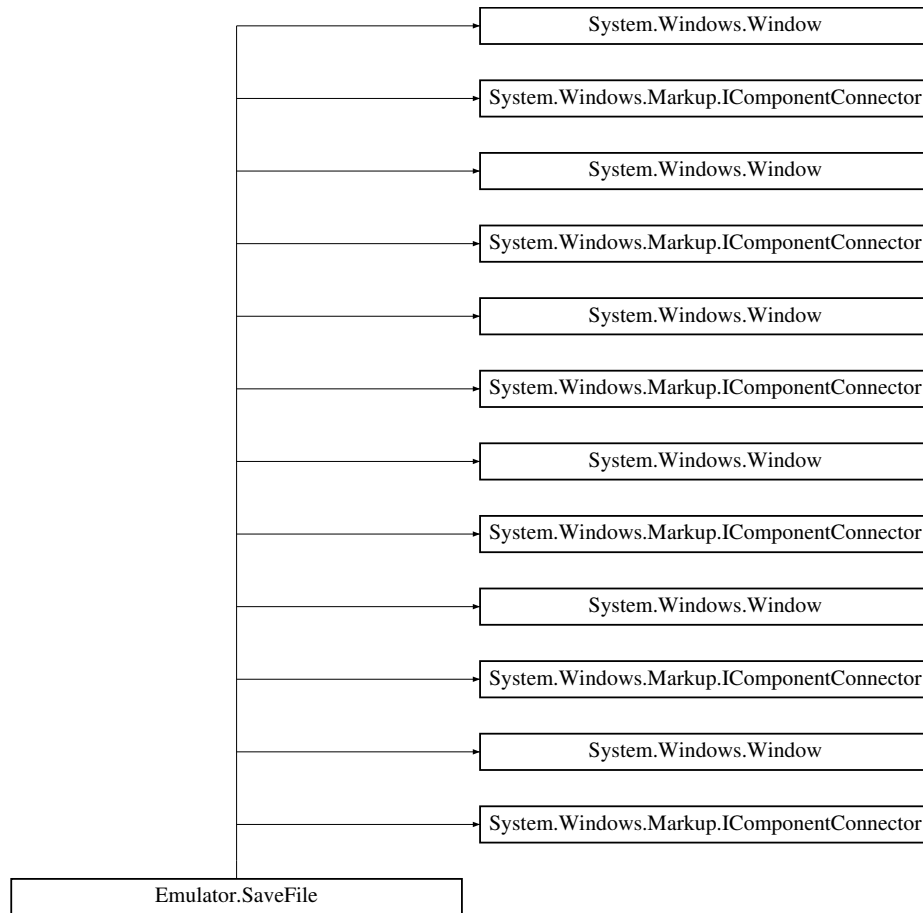
The documentation for this class was generated from the following file:

- Emulator/Model/[RomFileModel.cs](#)

6.29 Emulator.SaveFile Class Reference

SaveFile

Inheritance diagram for Emulator.SaveFile:



Public Member Functions

- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- [SaveFile](#) ()

Private Member Functions

- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void [NotificationMessageReceived](#) (NotificationMessage notificationMessage)

Private Attributes

- bool [_contentLoaded](#)

6.29.1 Detailed Description

[SaveFile](#)

Interaction logic for SaveState.xaml

Definition at line 40 of file [SaveFile.g.cs](#).

6.29.2 Constructor & Destructor Documentation

6.29.2.1 [SaveFile\(\)](#) Emulator.SaveFile.SaveFile () [inline]

Definition at line 10 of file [SaveFile.xaml.cs](#).

```
00011     {
00012         InitializeComponent ();
00013         Messenger.Default.Register<NotificationMessage>(this, NotificationMessageReceived);
00014     }
```

6.29.3 Member Function Documentation

6.29.3.1 [Connect\(\)](#) [1/6] void System.Windows.Markup.IComponentConnector. Emulator.SaveFile.↔

```
Connect (
    int connectionId,
    object target ) [inline], [private]
```

Definition at line 109 of file [SaveFile.g.cs](#).

```
00109     {
00110         switch (connectionId)
00111         {
00112             case 1:
00113                 this.SelectFile = ((System.Windows.Controls.Button) (target));
00114                 return;
00115             case 2:
00116                 this.FilePath = ((System.Windows.Controls.TextBox) (target));
00117                 return;
00118             case 3:
00119                 this.PathText = ((System.Windows.Controls.TextBlock) (target));
00120                 return;
00121             case 4:
00122                 this.CancelButton = ((System.Windows.Controls.Button) (target));
00123                 return;
00124             case 5:
00125                 this.LoadButton = ((System.Windows.Controls.Button) (target));
00126                 return;
00127         }
00128         this._contentLoaded = true;
00129     }
```

6.29.3.2 Connect() [2/6] void System.Windows.Markup.IComponentConnector. Emulator.SaveFile.↔

```
Connect (
    int connectionId,
    object target ) [inline], [private]
```

Definition at line 109 of file [SaveFile.g.i.cs](#).

```
00109 {
00110     switch (connectionId)
00111     {
00112     case 1:
00113         this.SelectFile = ((System.Windows.Controls.Button) (target));
00114         return;
00115     case 2:
00116         this.FilePath = ((System.Windows.Controls.TextBox) (target));
00117         return;
00118     case 3:
00119         this.PathText = ((System.Windows.Controls.TextBlock) (target));
00120         return;
00121     case 4:
00122         this.CancelButton = ((System.Windows.Controls.Button) (target));
00123         return;
00124     case 5:
00125         this.LoadButton = ((System.Windows.Controls.Button) (target));
00126         return;
00127     }
00128     this._contentLoaded = true;
00129 }
```

6.29.3.3 Connect() [3/6] void System.Windows.Markup.IComponentConnector. Emulator.SaveFile.↔

```
Connect (
    int connectionId,
    object target ) [inline], [private]
```

Definition at line 109 of file [SaveFile.g.cs](#).

```
00109 {
00110     switch (connectionId)
00111     {
00112     case 1:
00113         this.SelectFile = ((System.Windows.Controls.Button) (target));
00114         return;
00115     case 2:
00116         this.FilePath = ((System.Windows.Controls.TextBox) (target));
00117         return;
00118     case 3:
00119         this.PathText = ((System.Windows.Controls.TextBlock) (target));
00120         return;
00121     case 4:
00122         this.CancelButton = ((System.Windows.Controls.Button) (target));
00123         return;
00124     case 5:
00125         this.LoadButton = ((System.Windows.Controls.Button) (target));
00126         return;
00127     }
00128     this._contentLoaded = true;
00129 }
```

6.29.3.4 Connect() [4/6] void System.Windows.Markup.IComponentConnector. Emulator.SaveFile.↔

```
Connect (
    int connectionId,
    object target ) [inline], [private]
```

Definition at line 109 of file [SaveFile.g.i.cs](#).

```
00109 {
00110     switch (connectionId)
00111     {
00112     case 1:
00113         this.SelectFile = ((System.Windows.Controls.Button) (target));
00114         return;
```



```

00115         case 2:
00116             this.FilePath = ((System.Windows.Controls.TextBox) (target));
00117             return;
00118         case 3:
00119             this.PathText = ((System.Windows.Controls.TextBlock) (target));
00120             return;
00121         case 4:
00122             this.CancelButton = ((System.Windows.Controls.Button) (target));
00123             return;
00124         case 5:
00125             this.LoadButton = ((System.Windows.Controls.Button) (target));
00126             return;
00127     }
00128     this._contentLoaded = true;
00129 }

```

6.29.3.5 Connect() [5/6] void System.Windows.Markup.IComponentConnector. Emulator.SaveFile.↔
Connect (

```

    int connectionId,
    object target ) [inline], [private]

```

Definition at line 109 of file [SaveFile.g.cs](#).

```

00109                                                     {
00110         switch (connectionId)
00111         {
00112             case 1:
00113                 this.SelectFile = ((System.Windows.Controls.Button) (target));
00114                 return;
00115             case 2:
00116                 this.FilePath = ((System.Windows.Controls.TextBox) (target));
00117                 return;
00118             case 3:
00119                 this.PathText = ((System.Windows.Controls.TextBlock) (target));
00120                 return;
00121             case 4:
00122                 this.CancelButton = ((System.Windows.Controls.Button) (target));
00123                 return;
00124             case 5:
00125                 this.LoadButton = ((System.Windows.Controls.Button) (target));
00126                 return;
00127         }
00128         this._contentLoaded = true;
00129     }

```

6.29.3.6 Connect() [6/6] void System.Windows.Markup.IComponentConnector. Emulator.SaveFile.↔
Connect (

```

    int connectionId,
    object target ) [inline], [private]

```

Definition at line 109 of file [SaveFile.g.i.cs](#).

```

00109                                                     {
00110         switch (connectionId)
00111         {
00112             case 1:
00113                 this.SelectFile = ((System.Windows.Controls.Button) (target));
00114                 return;
00115             case 2:
00116                 this.FilePath = ((System.Windows.Controls.TextBox) (target));
00117                 return;
00118             case 3:
00119                 this.PathText = ((System.Windows.Controls.TextBlock) (target));
00120                 return;
00121             case 4:
00122                 this.CancelButton = ((System.Windows.Controls.Button) (target));
00123                 return;
00124             case 5:
00125                 this.LoadButton = ((System.Windows.Controls.Button) (target));
00126                 return;
00127         }
00128         this._contentLoaded = true;
00129     }

```

6.29.3.7 InitializeComponent() [1/6] void Emulator.SaveFile.InitializeComponent () [inline]

InitializeComponent

Definition at line 89 of file [SaveFile.g.cs](#).

```

00089                                     {
00090         if (_contentLoaded) {
00091             return;
00092         }
00093         _contentLoaded = true;
00094         System.Uri resourceLocator = new System.Uri("/Emulator;component/savefile.xaml",
System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097         System.Windows.Application.LoadComponent(this, resourceLocator);
00098
00099 #line default
00100 #line hidden
00101     }

```

6.29.3.8 InitializeComponent() [2/6] void Emulator.SaveFile.InitializeComponent () [inline]

InitializeComponent

Definition at line 89 of file [SaveFile.g.i.cs](#).

```

00089                                     {
00090         if (_contentLoaded) {
00091             return;
00092         }
00093         _contentLoaded = true;
00094         System.Uri resourceLocator = new System.Uri("/Emulator;component/savefile.xaml",
System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097         System.Windows.Application.LoadComponent(this, resourceLocator);
00098
00099 #line default
00100 #line hidden
00101     }

```

6.29.3.9 InitializeComponent() [3/6] void Emulator.SaveFile.InitializeComponent () [inline]

InitializeComponent

Definition at line 89 of file [SaveFile.g.cs](#).

```

00089                                     {
00090         if (_contentLoaded) {
00091             return;
00092         }
00093         _contentLoaded = true;
00094         System.Uri resourceLocator = new System.Uri("/Emulator;component/savefile.xaml",
System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097         System.Windows.Application.LoadComponent(this, resourceLocator);
00098
00099 #line default
00100 #line hidden
00101     }

```

6.29.3.10 InitializeComponent() [4/6] void Emulator.SaveFile.InitializeComponent () [inline]

InitializeComponent

Definition at line 89 of file [SaveFile.g.i.cs](#).

```

00089                                     {
00090         if (_contentLoaded) {
00091             return;
00092         }
00093         _contentLoaded = true;
00094         System.Uri resourceLocator = new System.Uri("/Emulator;component/savefile.xaml",
            System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097         System.Windows.Application.LoadComponent(this, resourceLocator);
00098
00099 #line default
00100 #line hidden
00101     }

```

6.29.3.11 InitializeComponent() [5/6] void Emulator.SaveFile.InitializeComponent () [inline]

InitializeComponent

Definition at line 89 of file [SaveFile.g.cs](#).

```

00089                                     {
00090         if (_contentLoaded) {
00091             return;
00092         }
00093         _contentLoaded = true;
00094         System.Uri resourceLocator = new System.Uri("/Emulator;component/savefile.xaml",
            System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097         System.Windows.Application.LoadComponent(this, resourceLocator);
00098
00099 #line default
00100 #line hidden
00101     }

```

6.29.3.12 InitializeComponent() [6/6] void Emulator.SaveFile.InitializeComponent () [inline]

InitializeComponent

Definition at line 89 of file [SaveFile.g.i.cs](#).

```

00089                                     {
00090         if (_contentLoaded) {
00091             return;
00092         }
00093         _contentLoaded = true;
00094         System.Uri resourceLocator = new System.Uri("/Emulator;component/savefile.xaml",
            System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097         System.Windows.Application.LoadComponent(this, resourceLocator);
00098
00099 #line default
00100 #line hidden
00101     }

```

6.29.3.13 NotificationMessageReceived() void Emulator.SaveFile.NotificationMessageReceived (NotificationMessage notificationMessage) [inline], [private]Definition at line 16 of file [SaveFile.xaml.cs](#).

```

00017     {
00018         if (notificationMessage.Notification == "CloseSaveFileWindow")
00019             Close();
00020     }

```

6.29.4 Member Data Documentation

6.29.4.1 `_contentLoaded` `bool` `Emulator.SaveFile._contentLoaded` `[private]`

Definition at line 82 of file [SaveFile.g.cs](#).

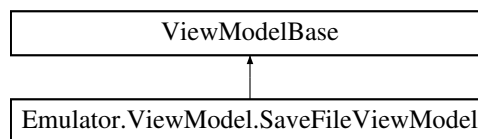
The documentation for this class was generated from the following files:

- [Emulator/obj/x86/Debug/SaveFile.g.cs](#)
- [Emulator/obj/x86/Debug/SaveFile.g.i.cs](#)
- [Emulator/obj/x86/Publish/SaveFile.g.cs](#)
- [Emulator/obj/x86/Publish/SaveFile.g.i.cs](#)
- [Emulator/obj/x86/Release/SaveFile.g.cs](#)
- [Emulator/obj/x86/Release/SaveFile.g.i.cs](#)
- [Emulator/SaveFile.xaml.cs](#)

6.30 Emulator.ViewModel.SaveFileViewModel Class Reference

The [ViewModel](#) Used by the [SaveFileView](#)

Inheritance diagram for `Emulator.ViewModel.SaveFileViewModel`:



Public Member Functions

- [SaveFileViewModel](#) ()
Instantiates a new instance of the [SaveFileViewModel](#). This is used by the IOC to create the default instance.
- [SaveFileViewModel](#) ([StateFileModel](#) stateFileModel)
Instantiates a new instance of the [SaveFileViewModel](#)

Properties

- RelayCommand [SaveFileCommand](#) `[get, set]`
The Relay Command called when saving a file
- RelayCommand [CloseCommand](#) `[get, set]`
The Relay Command called when closing a file
- RelayCommand [SelectFileCommand](#) `[get, set]`
The Relay Command called when Selecting a file
- string [Filename](#) `[get, set]`
The file to be saved
- bool [SaveEnabled](#) `[get]`
Tells the UI that that a file has been selected and can be saved.

Private Member Functions

- void [Save](#) ()
- void [Select](#) ()

Static Private Member Functions

- static void [Close](#) ()

Private Attributes

- readonly [StateFileModel](#) [_stateFileModel](#)

6.30.1 Detailed Description

The [ViewModel](#) Used by the [SaveFileView](#)

Definition at line 15 of file [SaveFileViewModel.cs](#).

6.30.2 Constructor & Destructor Documentation

6.30.2.1 [SaveFileViewModel\(\)](#) [1/2] `Emulator.ViewModel.SaveFileViewModel.SaveFileViewModel ()`
[inline]

Instantiates a new instance of the [SaveFileViewModel](#). This is used by the IOC to create the default instance.

Definition at line 51 of file [SaveFileViewModel.cs](#).

```
00052     {  
00053  
00054     }
```

6.30.2.2 [SaveFileViewModel\(\)](#) [2/2] `Emulator.ViewModel.SaveFileViewModel.SaveFileViewModel (StateFileModel stateFileModel)` [inline]

Instantiates a new instance of the [SaveFileViewModel](#)

Parameters

<i>stateFileModel</i>	The StateFileModel to be serialized to a file
-----------------------	---

Definition at line 60 of file [SaveFileViewModel.cs](#).

```
00061     {  
00062         SaveFileCommand = new RelayCommand(Save);  
00063         CloseCommand = new RelayCommand(Close);  
00064         SelectFileCommand = new RelayCommand(Select);  
00065         \_stateFileModel = stateFileModel;  
00066     }
```

6.30.3 Member Function Documentation

6.30.3.1 Close() static void Emulator.ViewModel.SaveFileViewModel.Close () [inline], [static], [private]

Definition at line 80 of file [SaveFileViewModel.cs](#).

```
00081     {
00082         Messenger.Default.Send(new NotificationMessage("CloseSaveFileWindow"));
00083     }
```

6.30.3.2 Save() void Emulator.ViewModel.SaveFileViewModel.Save () [inline], [private]

Definition at line 70 of file [SaveFileViewModel.cs](#).

```
00071     {
00072         var formatter = new BinaryFormatter();
00073         Stream stream = new FileStream(Filename, FileMode.Create, FileAccess.Write,
        FileShare.None);
00074         formatter.Serialize(stream, _stateFileModel);
00075         stream.Close();
00076
00077         Close();
00078     }
```

6.30.3.3 Select() void Emulator.ViewModel.SaveFileViewModel.Select () [inline], [private]

Definition at line 85 of file [SaveFileViewModel.cs](#).

```
00086     {
00087         var dialog = new SaveFileDialog { DefaultExt = ".6502", Filter = "WolfNet W65C02 Emulator
        Save State (*.6502)|*.6502" };
00088
00089         var result = dialog.ShowDialog();
00090
00091         if (result != true)
00092             return;
00093
00094         Filename = dialog.FileName;
00095         RaisePropertyChanged("Filename");
00096         RaisePropertyChanged("SaveEnabled");
00097
00098     }
```

6.30.4 Member Data Documentation

6.30.4.1 _stateFileModel readonly [StateFileModel](#) Emulator.ViewModel.SaveFileViewModel._state↔
FileModel [private]

Definition at line 17 of file [SaveFileViewModel.cs](#).

6.30.5 Property Documentation

6.30.5.1 CloseCommand `RelayCommand Emulator.ViewModel.SaveFileViewModel.CloseCommand [get], [set]`

The Relay Command called when closing a file

Definition at line 28 of file [SaveFileViewModel.cs](#).

```
00028 { get; set; }
```

6.30.5.2 Filename `string Emulator.ViewModel.SaveFileViewModel.Filename [get], [set]`

The file to be saved

Definition at line 38 of file [SaveFileViewModel.cs](#).

```
00038 { get; set; }
```

6.30.5.3 SaveEnabled `bool Emulator.ViewModel.SaveFileViewModel.SaveEnabled [get]`

Tells the UI that that a file has been selected and can be saved.

Definition at line 43 of file [SaveFileViewModel.cs](#).

```
00043 { get { return !string.IsNullOrEmpty(Filename); } }
```

6.30.5.4 SaveFileCommand `RelayCommand Emulator.ViewModel.SaveFileViewModel.SaveFileCommand [get], [set]`

The Relay Command called when saving a file

Definition at line 23 of file [SaveFileViewModel.cs](#).

```
00023 { get; set; }
```

6.30.5.5 SelectFileCommand `RelayCommand Emulator.ViewModel.SaveFileViewModel.SelectFile←
Command [get], [set]`

The Relay Command called when Selecting a file

Definition at line 33 of file [SaveFileViewModel.cs](#).

```
00033 { get; set; }
```

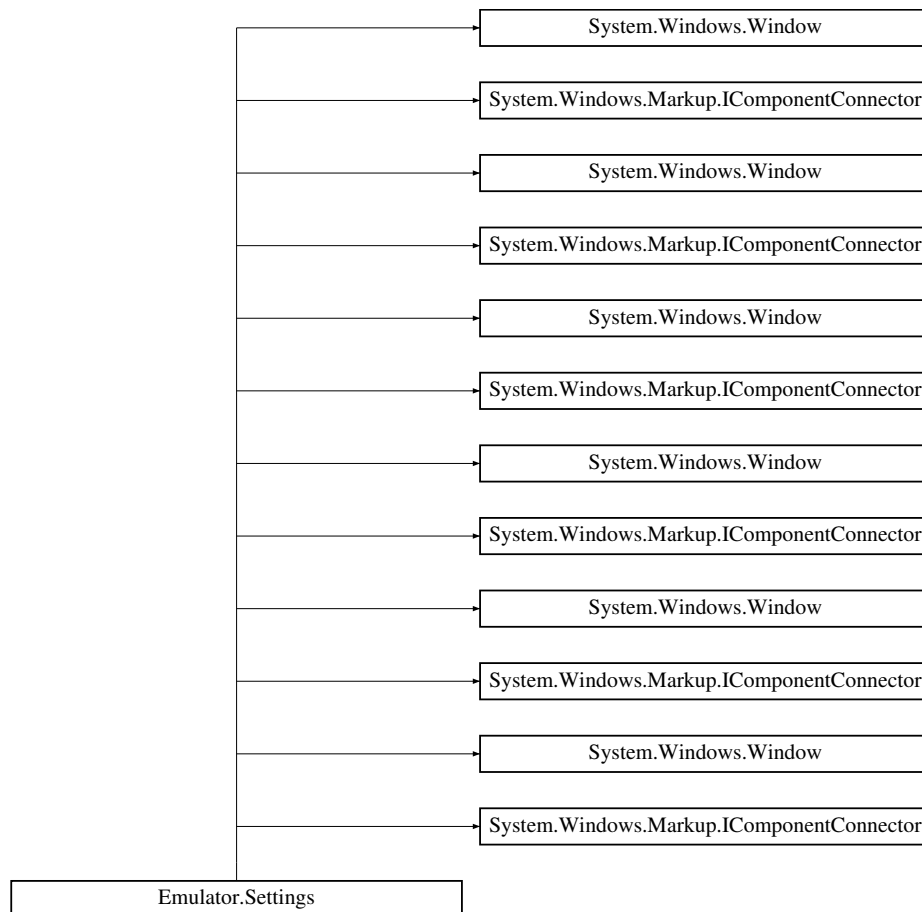
The documentation for this class was generated from the following file:

- Emulator/ViewModel/[SaveFileViewModel.cs](#)

6.31 Emulator.Settings Class Reference

Settings

Inheritance diagram for Emulator.Settings:



Public Member Functions

- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent
- [Settings](#) ()

Private Member Functions

- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void [NotificationMessageReceived](#) (NotificationMessage notificationMessage)
- void [NotificationMessageReceived](#) (NotificationMessage< [SettingsModel](#) > notificationMessage)
- void [PortSelectionDropDownClosed](#) (object sender, EventArgs e)

Private Attributes

- bool [_contentLoaded](#)

6.31.1 Detailed Description

[Settings](#)

Interaction logic for Settings.xaml

Definition at line 40 of file [Settings.g.cs](#).

6.31.2 Constructor & Destructor Documentation

6.31.2.1 [Settings\(\)](#) `Emulator.Settings.Settings () [inline]`

Definition at line 13 of file [Settings.xaml.cs](#).

```
00014     {
00015         InitializeComponent();
00016         Messenger.Default.Register<NotificationMessage>(this, NotificationMessageReceived);
00017         Messenger.Default.Register<NotificationMessage<SettingsModel>>(this,
00018             NotificationMessageReceived);
00018     }
```

6.31.3 Member Function Documentation

6.31.3.1 Connect() [1/6] void System.Windows.Markup.IComponentConnector. Emulator.Settings.↔

```
Connect (
    int connectionId,
    object target ) [inline], [private]
```

Definition at line 101 of file [Settings.g.cs](#).

```
00101                                     {
00102         switch (connectionId)
00103         {
00104             case 1:
00105                 this.ComPortCombo = ((System.Windows.Controls.ComboBox) (target));
00106
00107 #line 7 "..\..\..\Settings.xaml"
00108                 this.ComPortCombo.DropDownClosed += new
System.EventHandler(this.PortSelectionDropDownClosed);
00109
00110 #line default
00111 #line hidden
00112                 return;
00113             case 2:
00114                 this.PortText = ((System.Windows.Controls.TextBlock) (target));
00115                 return;
00116             case 3:
00117                 this.ApplyButton = ((System.Windows.Controls.Button) (target));
00118                 return;
00119             case 4:
00120                 this.CloseButton = ((System.Windows.Controls.Button) (target));
00121                 return;
00122             }
00123             this._contentLoaded = true;
00124         }
```

6.31.3.2 Connect() [2/6] void System.Windows.Markup.IComponentConnector. Emulator.Settings.↔

```
Connect (
    int connectionId,
    object target ) [inline], [private]
```

Definition at line 101 of file [Settings.g.i.cs](#).

```
00101                                     {
00102         switch (connectionId)
00103         {
00104             case 1:
00105                 this.ComPortCombo = ((System.Windows.Controls.ComboBox) (target));
00106
00107 #line 7 "..\..\..\Settings.xaml"
00108                 this.ComPortCombo.DropDownClosed += new
System.EventHandler(this.PortSelectionDropDownClosed);
00109
00110 #line default
00111 #line hidden
00112                 return;
00113             case 2:
00114                 this.PortText = ((System.Windows.Controls.TextBlock) (target));
00115                 return;
00116             case 3:
00117                 this.ApplyButton = ((System.Windows.Controls.Button) (target));
00118                 return;
00119             case 4:
00120                 this.CloseButton = ((System.Windows.Controls.Button) (target));
00121                 return;
00122             }
00123             this._contentLoaded = true;
00124         }
```

6.31.3.3 Connect() [3/6] void System.Windows.Markup.IComponentConnector. Emulator.Settings.↔

```
Connect (
    int connectionId,
    object target ) [inline], [private]
```

Definition at line 101 of file [Settings.g.cs](#).

```

00101                                                     {
00102         switch (connectionId)
00103         {
00104             case 1:
00105                 this.ComPortCombo = ((System.Windows.Controls.ComboBox) (target));
00106
00107             #line 7 "..\..\..\Settings.xaml"
00108                 this.ComPortCombo.DropDownClosed += new
System.EventHandler(this.PortSelectionDropDownClosed);
00109
00110             #line default
00111             #line hidden
00112                 return;
00113             case 2:
00114                 this.PortText = ((System.Windows.Controls.TextBlock) (target));
00115                 return;
00116             case 3:
00117                 this.ApplyButton = ((System.Windows.Controls.Button) (target));
00118                 return;
00119             case 4:
00120                 this.CloseButton = ((System.Windows.Controls.Button) (target));
00121                 return;
00122             }
00123             this._contentLoaded = true;
00124         }

```

6.31.3.4 Connect() [4/6] void System.Windows.Markup.IComponentConnector. Emulator.Settings.↔
Connect (

```

    int connectionId,
    object target ) [inline], [private]

```

Definition at line 101 of file [Settings.g.i.cs](#).

```

00101                                                     {
00102         switch (connectionId)
00103         {
00104             case 1:
00105                 this.ComPortCombo = ((System.Windows.Controls.ComboBox) (target));
00106
00107             #line 7 "..\..\..\Settings.xaml"
00108                 this.ComPortCombo.DropDownClosed += new
System.EventHandler(this.PortSelectionDropDownClosed);
00109
00110             #line default
00111             #line hidden
00112                 return;
00113             case 2:
00114                 this.PortText = ((System.Windows.Controls.TextBlock) (target));
00115                 return;
00116             case 3:
00117                 this.ApplyButton = ((System.Windows.Controls.Button) (target));
00118                 return;
00119             case 4:
00120                 this.CloseButton = ((System.Windows.Controls.Button) (target));
00121                 return;
00122             }
00123             this._contentLoaded = true;
00124         }

```

6.31.3.5 Connect() [5/6] void System.Windows.Markup.IComponentConnector. Emulator.Settings.↔
Connect (

```

    int connectionId,
    object target ) [inline], [private]

```

Definition at line 101 of file [Settings.g.cs](#).

```

00101                                                     {
00102         switch (connectionId)
00103         {
00104             case 1:
00105                 this.ComPortCombo = ((System.Windows.Controls.ComboBox) (target));
00106

```

```

00107 #line 7 "..\..\..\Settings.xaml"
00108     this.ComPortCombo.DropDownClosed += new
System.EventHandler(this.PortSelectionDropDownClosed);
00109
00110 #line default
00111 #line hidden
00112     return;
00113     case 2:
00114         this.PortText = ((System.Windows.Controls.TextBlock) (target));
00115         return;
00116     case 3:
00117         this.ApplyButton = ((System.Windows.Controls.Button) (target));
00118         return;
00119     case 4:
00120         this.CloseButton = ((System.Windows.Controls.Button) (target));
00121         return;
00122     }
00123     this._contentLoaded = true;
00124 }

```

6.31.3.6 Connect() [6/6] void System.Windows.Markup.IComponentConnector. Emulator.Settings.↔
Connect (

```

    int connectionId,
    object target ) [inline], [private]

```

Definition at line 101 of file [Settings.g.i.cs](#).

```

00101
00102     switch (connectionId)
00103     {
00104     case 1:
00105         this.ComPortCombo = ((System.Windows.Controls.ComboBox) (target));
00106
00107 #line 7 "..\..\..\Settings.xaml"
00108     this.ComPortCombo.DropDownClosed += new
System.EventHandler(this.PortSelectionDropDownClosed);
00109
00110 #line default
00111 #line hidden
00112     return;
00113     case 2:
00114         this.PortText = ((System.Windows.Controls.TextBlock) (target));
00115         return;
00116     case 3:
00117         this.ApplyButton = ((System.Windows.Controls.Button) (target));
00118         return;
00119     case 4:
00120         this.CloseButton = ((System.Windows.Controls.Button) (target));
00121         return;
00122     }
00123     this._contentLoaded = true;
00124 }

```

6.31.3.7 InitializeComponent() [1/6] void Emulator.Settings.InitializeComponent () [inline]

InitializeComponent

Definition at line 81 of file [Settings.g.cs](#).

```

00081
00082     if (_contentLoaded) {
00083         return;
00084     }
00085     _contentLoaded = true;
00086     System.Uri resourceLocator = new System.Uri("/Emulator;component/settings.xaml",
System.UriKind.Relative);
00087
00088 #line 1 "..\..\..\Settings.xaml"
00089     System.Windows.Application.LoadComponent(this, resourceLocator);
00090
00091 #line default
00092 #line hidden
00093 }

```

6.31.3.8 InitializeComponent() [2/6] void Emulator.Settings.InitializeComponent () [inline]

InitializeComponent

Definition at line 81 of file [Settings.g.i.cs](#).

```

00081                                     {
00082             if (_contentLoaded) {
00083                 return;
00084             }
00085             _contentLoaded = true;
00086             System.Uri resourceLocator = new System.Uri("/Emulator;component/settings.xaml",
System.UriKind.Relative);
00087
00088 #line 1 "..\..\..\Settings.xaml"
00089     System.Windows.Application.LoadComponent(this, resourceLocator);
00090
00091 #line default
00092 #line hidden
00093     }

```

6.31.3.9 InitializeComponent() [3/6] void Emulator.Settings.InitializeComponent () [inline]

InitializeComponent

Definition at line 81 of file [Settings.g.cs](#).

```

00081                                     {
00082             if (_contentLoaded) {
00083                 return;
00084             }
00085             _contentLoaded = true;
00086             System.Uri resourceLocator = new System.Uri("/Emulator;component/settings.xaml",
System.UriKind.Relative);
00087
00088 #line 1 "..\..\..\Settings.xaml"
00089     System.Windows.Application.LoadComponent(this, resourceLocator);
00090
00091 #line default
00092 #line hidden
00093     }

```

6.31.3.10 InitializeComponent() [4/6] void Emulator.Settings.InitializeComponent () [inline]

InitializeComponent

Definition at line 81 of file [Settings.g.i.cs](#).

```

00081                                     {
00082             if (_contentLoaded) {
00083                 return;
00084             }
00085             _contentLoaded = true;
00086             System.Uri resourceLocator = new System.Uri("/Emulator;component/settings.xaml",
System.UriKind.Relative);
00087
00088 #line 1 "..\..\..\Settings.xaml"
00089     System.Windows.Application.LoadComponent(this, resourceLocator);
00090
00091 #line default
00092 #line hidden
00093     }

```

6.31.3.11 InitializeComponent() [5/6] void Emulator.Settings.InitializeComponent () [inline]

InitializeComponent

Definition at line 81 of file [Settings.g.cs](#).

```

00081                                     {
00082             if (_contentLoaded) {
00083                 return;
00084             }
00085             _contentLoaded = true;
00086             System.Uri resourceLocater = new System.Uri("/Emulator;component/settings.xaml",
System.UriKind.Relative);
00087
00088 #line 1 "..\..\..\Settings.xaml"
00089             System.Windows.Application.LoadComponent(this, resourceLocater);
00090
00091 #line default
00092 #line hidden
00093     }

```

6.31.3.12 InitializeComponent() [6/6] void Emulator.Settings.InitializeComponent () [inline]

InitializeComponent

Definition at line 81 of file [Settings.g.i.cs](#).

```

00081                                     {
00082             if (_contentLoaded) {
00083                 return;
00084             }
00085             _contentLoaded = true;
00086             System.Uri resourceLocater = new System.Uri("/Emulator;component/settings.xaml",
System.UriKind.Relative);
00087
00088 #line 1 "..\..\..\Settings.xaml"
00089             System.Windows.Application.LoadComponent(this, resourceLocater);
00090
00091 #line default
00092 #line hidden
00093     }

```

6.31.3.13 NotificationMessageReceived() [1/2] void Emulator.Settings.NotificationMessageReceived (
NotificationMessage notificationMessage) [inline], [private]Definition at line 20 of file [Settings.xaml.cs](#).

```

00021     {
00022         if (notificationMessage.Notification == "CloseSettingsWindow")
00023         {
00024             Close();
00025         }
00026     }

```

6.31.3.14 NotificationMessageReceived() [2/2] void Emulator.Settings.NotificationMessageReceived (
NotificationMessage< SettingsModel > notificationMessage) [inline], [private]Definition at line 28 of file [Settings.xaml.cs](#).

```

00029     {
00030         if (notificationMessage.Notification == "SettingsWindow")
00031         {
00032             SettingsViewModel.SettingsModel = notificationMessage.Content;
00033             ComPortCombo.SelectedItem = notificationMessage.Content.ComPortName;
00034         }
00035     }

```

6.31.3.15 PortSelectionDropDownClosed() void Emulator.Settings.PortSelectionDropDownClosed (
object sender,
EventArgs e) [inline], [private]

Definition at line 37 of file [Settings.xaml.cs](#).

```
00038     {  
00039         if (!(ComPortCombo.SelectedValue == null))  
00040         {  
00041             string port = ComPortCombo.SelectedValue.ToString();  
00042             SettingsViewModel.ComPortSelection = port;  
00043         }  
00044     }
```

6.31.4 Member Data Documentation

6.31.4.1 _contentLoaded bool Emulator.Settings._contentLoaded [private]

Definition at line 74 of file [Settings.g.cs](#).

The documentation for this class was generated from the following files:

- Emulator/obj/x86/Debug/[Settings.g.cs](#)
- Emulator/obj/x86/Debug/[Settings.g.i.cs](#)
- Emulator/obj/x86/Publish/[Settings.g.cs](#)
- Emulator/obj/x86/Publish/[Settings.g.i.cs](#)
- Emulator/obj/x86/Release/[Settings.g.cs](#)
- Emulator/obj/x86/Release/[Settings.g.i.cs](#)
- Emulator/[Settings.xaml.cs](#)

6.32 Emulator.SettingsFile Class Reference

Static Public Member Functions

- static [SettingsModel CreateNew](#) ()

6.32.1 Detailed Description

Definition at line 5 of file [SettingsFile.cs](#).

6.32.2 Member Function Documentation

6.32.2.1 CreateNew() static `SettingsModel` Emulator.SettingsFile.CreateNew () [inline], [static]

Definition at line 7 of file [SettingsFile.cs](#).

```

00008      {
00009          // Create new settings file.
00010          SettingsModel _settings = new SettingsModel
00011          {
00012              SettingsVersionMajor = Versioning.SettingsFile.Major,
00013              SettingsVersionMinor = Versioning.SettingsFile.Minor,
00014              SettingsVersionBuild = Versioning.SettingsFile.Build,
00015              SettingsVersionRevision = Versioning.SettingsFile.Revision,
00016          #if DEBUG
00017              ComPortName = "COM9",
00018          #else
00019              ComPortName = "COM1",
00020          #endif
00021          };
00022          return _settings;
00023      }

```

The documentation for this class was generated from the following file:

- Emulator/Classes/[SettingsFile.cs](#)

6.33 Emulator.Versioning.SettingsFile Class Reference

Static Public Attributes

- const byte `Major` = 1
- const byte `Minor` = 0
- const byte `Build` = 0
- const byte `Revision` = 0

6.33.1 Detailed Description

Definition at line 18 of file [Versioning.cs](#).

6.33.2 Member Data Documentation

6.33.2.1 Build const byte Emulator.Versioning.SettingsFile.Build = 0 [static]

Definition at line 22 of file [Versioning.cs](#).

6.33.2.2 Major const byte Emulator.Versioning.SettingsFile.Major = 1 [static]

Definition at line 20 of file [Versioning.cs](#).

6.33.2.3 Minor `const byte Emulator.Versioning.SettingsFile.Minor = 0 [static]`

Definition at line 21 of file [Versioning.cs](#).

6.33.2.4 Revision `const byte Emulator.Versioning.SettingsFile.Revision = 0 [static]`

Definition at line 23 of file [Versioning.cs](#).

The documentation for this class was generated from the following file:

- Emulator/Classes/[Versioning.cs](#)

6.34 Emulator.Model.SettingsModel Class Reference

[Model](#) that contains the required information needed to save the current settings to disk

Properties

- byte [SettingsVersionMajor](#) [get, set]
The version of the file that is being saved
- byte [SettingsVersionMinor](#) [get, set]
The version of the file that is being saved
- byte [SettingsVersionBuild](#) [get, set]
The version of the file that is being saved
- byte [SettingsVersionRevision](#) [get, set]
The version of the file that is being saved
- string [ComPortName](#) [get, set]
The PC port that is being saved

6.34.1 Detailed Description

[Model](#) that contains the required information needed to save the current settings to disk

Definition at line 11 of file [SettingsModel.cs](#).

6.34.2 Property Documentation

6.34.2.1 ComPortName `string Emulator.Model.SettingsModel.ComPortName [get], [set]`

The PC port that is being saved

Definition at line 36 of file [SettingsModel.cs](#).

```
00036 { get; set; }
```

6.34.2.2 SettingsVersionBuild `byte Emulator.Model.SettingsModel.SettingsVersionBuild [get], [set]`

The version of the file that is being saved

Definition at line 26 of file [SettingsModel.cs](#).

```
00026 { get; set; }
```

6.34.2.3 SettingsVersionMajor `byte Emulator.Model.SettingsModel.SettingsVersionMajor [get], [set]`

The version of the file that is being saved

Definition at line 16 of file [SettingsModel.cs](#).

```
00016 { get; set; }
```

6.34.2.4 SettingsVersionMinor `byte Emulator.Model.SettingsModel.SettingsVersionMinor [get], [set]`

The version of the file that is being saved

Definition at line 21 of file [SettingsModel.cs](#).

```
00021 { get; set; }
```

6.34.2.5 SettingsVersionRevision `byte Emulator.Model.SettingsModel.SettingsVersionRevision [get], [set]`

The version of the file that is being saved

Definition at line 31 of file [SettingsModel.cs](#).

```
00031 { get; set; }
```

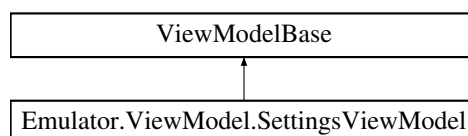
The documentation for this class was generated from the following file:

- [Emulator/Model/SettingsModel.cs](#)

6.35 Emulator.ViewModel.SettingsViewModel Class Reference

The [ViewModel](#) Used by the SaveFileView

Inheritance diagram for Emulator.ViewModel.SettingsViewModel:



Public Member Functions

- [SettingsViewModel](#) ()
Instantiates a new instance of the [SettingsViewModel](#). This is used by the IOC to create the default instance.
- [SettingsViewModel](#) ([SettingsModel](#) settingsModel)
Instantiates a new instance of the [SettingsViewModel](#)
- void [UpdatePortList](#) ()
Updates PortList with the COM ports available to the computer

Properties

- RelayCommand [ApplyCommand](#) [get, set]
The Relay Command called when saving a file
- RelayCommand [CloseCommand](#) [get, set]
The Relay Command called when closing a file
- bool [ApplyEnabled](#) [get]
Tells the UI that that a file has been selected and can be saved.
- ObservableCollection< string > [PortList](#) [get]
Creates a new instance of PortList, the list of all COM ports available to the computer
- static string [ComPortSelection](#) [get, set]
- static [SettingsModel](#) [SettingsModel](#) [get, set]

Private Member Functions

- void [Apply](#) ()

Static Private Member Functions

- static void [Close](#) ()

Private Attributes

- readonly ObservableCollection< string > [_PortList](#) = new ObservableCollection<string>()

6.35.1 Detailed Description

The [ViewModel](#) Used by the SaveFileView

Definition at line 14 of file [SettingsViewModel.cs](#).

6.35.2 Constructor & Destructor Documentation

6.35.2.1 SettingsViewModel() [1/2] `Emulator.ViewModel.SettingsViewModel.SettingsViewModel ()`
[inline]

Instantiates a new instance of the [SettingsViewModel](#). This is used by the IOC to create the default instance.

Definition at line 48 of file [SettingsViewModel.cs](#).

```
00049     {  
00050  
00051     }
```

6.35.2.2 SettingsViewModel() [2/2] `Emulator.ViewModel.SettingsViewModel.SettingsViewModel (
 SettingsModel settingsModel)` [inline]

Instantiates a new instance of the [SettingsViewModel](#)

Parameters

<i>settingsModel</i>	The SettingsFileModel to be serialized to a file
----------------------	--

Definition at line 57 of file [SettingsViewModel.cs](#).

```
00058     {
00059         ApplyCommand = new RelayCommand(Apply);
00060         CloseCommand = new RelayCommand(Close);
00061         ComPortSelection = settingsModel.ComPortName;
00062
00063         UpdatePortList();
00064     }
```

6.35.3 Member Function Documentation

6.35.3.1 Apply() void Emulator.ViewModel.SettingsViewModel.Apply () [inline], [private]

Definition at line 81 of file [SettingsViewModel.cs](#).

```
00082     {
00083         Messenger.Default.Send(new NotificationMessage<SettingsModel>(new SettingsModel
00084     {
00085         SettingsVersionMajor = Versioning.SettingsFile.Major,
00086         SettingsVersionMinor = Versioning.SettingsFile.Minor,
00087         SettingsVersionBuild = Versioning.SettingsFile.Build,
00088         SettingsVersionRevision = Versioning.SettingsFile.Revision,
00089         ComPortName = ComPortSelection,
00090     }, "SettingsApplied"));
00091         Messenger.Default.Send(new NotificationMessage("CloseSettingsWindow"));
00092     }
```

6.35.3.2 Close() static void Emulator.ViewModel.SettingsViewModel.Close () [inline], [static], [private]

Definition at line 94 of file [SettingsViewModel.cs](#).

```
00095     {
00096         Messenger.Default.Send(new NotificationMessage("CloseSettingsWindow"));
00097     }
```

6.35.3.3 UpdatePortList() void Emulator.ViewModel.SettingsViewModel.UpdatePortList () [inline]

Updates PortList with the COM ports available to the computer

Definition at line 69 of file [SettingsViewModel.cs](#).

```
00070     {
00071         PortList.Clear();
00072         foreach (string s in SerialPort.GetPortNames())
00073         {
00074             PortList.Add(s);
00075         }
00076         RaisePropertyChanged("PortList");
00077     }
```

6.35.4 Member Data Documentation

6.35.4.1 _PortList readonly ObservableCollection<string> Emulator.ViewModel.SettingsView↔
Model._PortList = new ObservableCollection<string>() [private]

Definition at line 37 of file [SettingsViewModel.cs](#).

6.35.5 Property Documentation

6.35.5.1 ApplyCommand RelayCommand Emulator.ViewModel.SettingsViewModel.ApplyCommand [get],
[set]

The Relay Command called when saving a file

Definition at line 20 of file [SettingsViewModel.cs](#).

```
00020 { get; set; }
```

6.35.5.2 ApplyEnabled bool Emulator.ViewModel.SettingsViewModel.ApplyEnabled [get]

Tells the UI that that a file has been selected and can be saved.

Definition at line 30 of file [SettingsViewModel.cs](#).

```
00030 { get { return !string.IsNullOrEmpty(Emulator.FileLocations.SettingsFile); } }
```

6.35.5.3 CloseCommand RelayCommand Emulator.ViewModel.SettingsViewModel.CloseCommand [get],
[set]

The Relay Command called when closing a file

Definition at line 25 of file [SettingsViewModel.cs](#).

```
00025 { get; set; }
```

6.35.5.4 ComPortSelection string Emulator.ViewModel.SettingsViewModel.ComPortSelection [static],
[get], [set]

Definition at line 39 of file [SettingsViewModel.cs](#).

```
00039 { get; set; }
```

6.35.5.5 PortList ObservableCollection<string> Emulator.ViewModel.SettingsViewModel.PortList
[get]

Creates a new instance of PortList, the list of all COM ports available to the computer

Definition at line 36 of file [SettingsViewModel.cs](#).

```
00036 { get { return _PortList; } }
```

6.35.5.6 SettingsModel `SettingsModel` `Emulator.ViewModel.SettingsViewModel.SettingsModel` `[static], [get], [set]`

Definition at line 40 of file [SettingsViewModel.cs](#).

```
00040 { get; set; }
```

The documentation for this class was generated from the following file:

- [Emulator/ViewModel/SettingsViewModel.cs](#)

6.36 Hardware.MemoryMap.SharedRom Class Reference

Static Public Attributes

- static byte `TotalBanks` = 1

Properties

- static int `Offset` `[get]`
- static int `Length` `[get]`

Static Private Attributes

- static int `_Offset` = 0xE000
- static int `_Length` = 0x1FFF

6.36.1 Detailed Description

Definition at line 44 of file [MemoryMap.cs](#).

6.36.2 Member Data Documentation

6.36.2.1 _Length `int` `Hardware.MemoryMap.SharedRom._Length` = 0x1FFF `[static], [private]`

Definition at line 47 of file [MemoryMap.cs](#).

6.36.2.2 _Offset `int` `Hardware.MemoryMap.SharedRom._Offset` = 0xE000 `[static], [private]`

Definition at line 46 of file [MemoryMap.cs](#).

6.36.2.3 TotalBanks `byte Hardware.MemoryMap.SharedRom.TotalBanks = 1 [static]`

Definition at line 49 of file [MemoryMap.cs](#).

6.36.3 Property Documentation

6.36.3.1 Length `int Hardware.MemoryMap.SharedRom.Length [static], [get]`

Definition at line 52 of file [MemoryMap.cs](#).

```
00052 { get { return _Length; } }
```

6.36.3.2 Offset `int Hardware.MemoryMap.SharedRom.Offset [static], [get]`

Definition at line 51 of file [MemoryMap.cs](#).

```
00051 { get { return _Offset; } }
```

The documentation for this class was generated from the following file:

- [Hardware/Classes/MemoryMap.cs](#)

6.37 Emulator.Model.StateFileModel Class Reference

[Model](#) that contains the required information needed to save the current state of the processor to disk

Properties

- `int NumberOfCycles [get, set]`
The Number of Cycles the Program has Ran so Far
- `IList< OutputLog > OutputLog [get, set]`
The output of the program
- `Hardware.W65C02 W65C02 [get, set]`
The Processor Object that is being saved
- `Hardware.W65C22 W65C22 [get, set]`
The first VIA Object that is being saved
- `Hardware.W65C22 MM65SIB [get, set]`
The second VIA Object that is being saved
- `Hardware.W65C51 W65C51 [get, set]`
The ACIA Object that is being saved
- `Hardware.AT28CXX AT28C010 [get, set]`
The Shared ROM Object that is being saved
- `Hardware.AT28CXX AT28C64 [get, set]`
The Banked ROM Object that is being saved

6.37.1 Detailed Description

[Model](#) that contains the required information needed to save the current state of the processor to disk

Definition at line 10 of file [StateFileModel.cs](#).

6.37.2 Property Documentation

6.37.2.1 AT28C010 [Hardware.AT28CXX](#) Emulator.Model.StateFileModel.AT28C010 [get], [set]

The Shared ROM Object that is being saved

Definition at line 45 of file [StateFileModel.cs](#).

```
00045 { get; set; }
```

6.37.2.2 AT28C64 [Hardware.AT28CXX](#) Emulator.Model.StateFileModel.AT28C64 [get], [set]

The Banked ROM Object that is being saved

Definition at line 50 of file [StateFileModel.cs](#).

```
00050 { get; set; }
```

6.37.2.3 MM65SIB [Hardware.W65C22](#) Emulator.Model.StateFileModel.MM65SIB [get], [set]

The second VIA Object that is being saved

Definition at line 35 of file [StateFileModel.cs](#).

```
00035 { get; set; }
```

6.37.2.4 NumberOfCycles [int](#) Emulator.Model.StateFileModel.NumberOfCycles [get], [set]

The Number of Cycles the Program has Ran so Far

Definition at line 15 of file [StateFileModel.cs](#).

```
00015 { get; set; }
```

6.37.2.5 OutputLog [IList<OutputLog>](#) Emulator.Model.StateFileModel.OutputLog [get], [set]

The output of the program

Definition at line 20 of file [StateFileModel.cs](#).

```
00020 { get; set; }
```

6.37.2.6 W65C02 [Hardware.W65C02](#) Emulator.Model.StateFileModel.W65C02 [get], [set]

The Processor Object that is being saved

Definition at line 25 of file [StateFileModel.cs](#).

```
00025 { get; set; }
```

6.37.2.7 W65C22 [Hardware.W65C22](#) Emulator.Model.StateFileModel.W65C22 [get], [set]

The first VIA Object that is being saved

Definition at line 30 of file [StateFileModel.cs](#).

```
00030 { get; set; }
```

6.37.2.8 W65C51 [Hardware.W65C51](#) Emulator.Model.StateFileModel.W65C51 [get], [set]

The ACIA Object that is being saved

Definition at line 40 of file [StateFileModel.cs](#).

```
00040 { get; set; }
```

The documentation for this class was generated from the following file:

- Emulator/Model/[StateFileModel.cs](#)

6.38 Hardware.Utility Class Reference

Static Public Member Functions

- static string [ConvertOpCodeIntoString](#) (this int i)

6.38.1 Detailed Description

Definition at line 5 of file [Utility.cs](#).

6.38.2 Member Function Documentation

6.38.2.1 ConvertOpCodeIntoString() static string Hardware.Utility.ConvertOpCodeIntoString (this int i) [inline], [static]

Definition at line 7 of file [Utility.cs](#).

```

00008     {
00009         switch (i)
00010         {
00011             case 0x69: //ãADCãImmediate
00012             case 0x65: //ãADCãZeroãPage
00013             case 0x75: //ãADCãZeroãPageãX
00014             case 0x6D: //ãADCãAbsolute
00015             case 0x7D: //ãADCãAbsoluteãX
00016             case 0x79: //ãADCãAbsoluteãY
00017             case 0x61: //ãADCãIndirectãX
00018             case 0x71: //ãADCãIndirectãY
00019             {
00020                 return "ADC";
00021             }
00022             case 0x29: //ãANDãImmediate
00023             case 0x25: //ãANDãZeroãPage
00024             case 0x35: //ãANDãZeroãPageãX
00025             case 0x2D: //ãANDãAbsolute
00026             case 0x3D: //ãANDãAbsoluteãX
00027             case 0x39: //ãANDãAbsoluteãY
00028             case 0x21: //ãANDãIndirectãX
00029             case 0x31: //ãANDãIndirectãY
00030             {
00031                 return "AND";
00032             }
00033             case 0x0A: //ãASLãAccumulator
00034             case 0x06: //ãASLãZeroãPage
00035             case 0x16: //ãASLãZeroãPageãX
00036             case 0x0E: //ãASLãAbsolute
00037             case 0x1E: //ãASLãAbsoluteãX
00038             {
00039                 return "ASL";
00040             }
00041             case 0x90: //ãBCCãRelative
00042             {
00043                 return "BCC";
00044             }
00045             case 0xB0: //ãBCSãRelative
00046             {
00047                 return "BCS";
00048             }
00049             case 0xF0: //ãBEQãRelative
00050             {
00051                 return "BEQ";
00052             }
00053             case 0x24: //ãBITãZeroãPage
00054             case 0x2C: //ãBITãAbsolute
00055             {
00056                 return "BIT";
00057             }
00058             case 0x30: //ãBMIãRelative
00059             {
00060                 return "BMI";
00061             }
00062             case 0xD0: //ãBNEãRelative
00063             {
00064                 return "BNE";
00065             }
00066             case 0x10: //ãBPLãRelative
00067             {
00068                 return "BPL";
00069             }
00070             case 0x00: //ãBRKãImplied
00071             {
00072                 return "BRK";
00073             }
00074             case 0x50: // BVC Relative
00075             {
00076                 return "BCV";
00077             }
00078             case 0x70: //BVS Relative
00079             {
00080                 return "BVS";
00081             }
00082             case 0x18: //ãCLCãImplied
00083             {
00084                 return "CLC";
00085             }
00086             case 0xD8: //ãCLDãImplied
00087             {
00088                 return "CLD";
00089             }

```

```

00090         case 0x58: //äCLIäImplied
00091         {
00092             return "CLI";
00093         }
00094         case 0xB8: //äCLVäImplied
00095         {
00096             return "CLV";
00097         }
00098         case 0xC9: //äCMPäImmediate
00099         case 0xC5: //äCMPäZeroPage
00100         case 0xD5: //äCMPäZeroäPageäX
00101         case 0xCD: //äCMPäAbsolute
00102         case 0xDD: //äCMPäAbsoluteäX
00103         case 0xD9: //äCMPäAbsoluteäY
00104         case 0xC1: //äCMPäIndirectäX
00105         case 0xD1: //äCMPäIndirectäY
00106         {
00107             return "CMP";
00108         }
00109         case 0xE0: //äCPXäImmediate
00110         case 0xE4: //äCPXäZeroPage
00111         case 0xEC: //äCPXäAbsolute
00112         {
00113             return "CPX";
00114         }
00115         case 0xC0: //äCPYäImmediate
00116         case 0xC4: //äCPYäZeroPage
00117         case 0xCC: //äCPYäAbsolute
00118         {
00119             return "CPY";
00120         }
00121         case 0xC6: //äDECäZeroäPage
00122         case 0xD6: //äDECäZeroäPageäX
00123         case 0xCE: //äDECäAbsolute
00124         case 0xDE: //äDECäAbsoluteäX
00125         {
00126             return "DEC";
00127         }
00128         case 0xCA: //äDEXäImplied
00129         {
00130             return "DEX";
00131         }
00132         case 0x88: //äDEYäImplied
00133         {
00134             return "DEY";
00135         }
00136         case 0x49: //äEORäImmediate
00137         case 0x45: //äEORäZeroäPage
00138         case 0x55: //äEORäZeroäPageäX
00139         case 0x4D: //äEORäAbsolute
00140         case 0x5D: //äEORäAbsoluteäX
00141         case 0x59: //äEORäAbsoluteäY
00142         case 0x41: //äEORäIndirectäX
00143         case 0x51: //äEORäIndirectäY
00144         {
00145             return "EOR";
00146         }
00147         case 0xE6: //äINCäZeroäPage
00148         case 0xF6: //äINCäZeroäPageäX
00149         {
00150             return "INC";
00151         }
00152         case 0xE8: //äINXäImplied
00153         {
00154             return "INX";
00155         }
00156         case 0xC8: //äINYäImplied
00157         {
00158             return "INY";
00159         }
00160         case 0xEE: //äINCäAbsolute
00161         case 0xFE: //äINCäAbsoluteäX
00162         {
00163             return "INC";
00164         }
00165         case 0x4C: //äJMPäAbsolute
00166         case 0x6C: //äJMPäIndirect
00167         {
00168             return "JMP";
00169         }
00170         case 0x20: //äJSRäAbsolute
00171         {
00172             return "JSR";
00173         }
00174         case 0xA9: //äLDAäImmediate
00175         case 0xA5: //äLDAäZeroäPage
00176         case 0xB5: //äLDAäZeroäPageäX

```

```

00177         case 0xAD: //äLDAäAbsolute
00178         case 0xBD: //äLDAäAbsoluteäX
00179         case 0xB9: //äLDAäAbsoluteäY
00180         case 0xA1: //äLDAäIndirectäX
00181         case 0xB1: //äLDAäIndirectäY
00182         {
00183             return "LDA";
00184         }
00185         case 0xA2: //äLDXäImmediate
00186         case 0xA6: //äLDXäZeroäPage
00187         case 0xB6: //äLDXäZeroäPageäY
00188         case 0xAE: //äLDXäAbsolute
00189         case 0xBE: //äLDXäAbsoluteäY
00190         {
00191             return "LDX";
00192         }
00193         case 0xA0: //äLDYäImmediate
00194         case 0xA4: //äLDYäZeroäPage
00195         case 0xB4: //äLDYäZeroäPageäY
00196         case 0xAC: //äLDYäAbsolute
00197         case 0xBC: //äLDYäAbsoluteäY
00198         {
00199             return "LDY";
00200         }
00201         case 0x4A: //äLSRäAccumulator
00202         case 0x46: //äLSRäZeroäPage
00203         case 0x56: //äLSRäZeroäPageäX
00204         case 0x4E: //äLSRäAbsolute
00205         case 0x5E: //äLSRäAbsoluteäX
00206         {
00207             return "LSR";
00208         }
00209         case 0xEA: //äNOPäImplied
00210         {
00211             return "NOP";
00212         }
00213         case 0x09: //äORAäImmediate
00214         case 0x05: //äORAäZeroäPage
00215         case 0x15: //äORAäZeroäPageäX
00216         case 0x0D: //äORAäAbsolute
00217         case 0x1D: //äORAäAbsoluteäX
00218         case 0x19: //äORAäAbsoluteäY
00219         case 0x01: //äORAäIndirectäX
00220         case 0x11: //äORAäIndirectäY
00221         {
00222             return "ORA";
00223         }
00224         case 0x48: //äPHAäImplied
00225         {
00226             return "PHA";
00227         }
00228         case 0x08: //äPHPäImplied
00229         {
00230             return "PHP";
00231         }
00232         case 0x68: //äPLAäImplied
00233         {
00234             return "PLA";
00235         }
00236         case 0x28: //äPLPäImplied
00237         {
00238             return "PLP";
00239         }
00240         case 0x2A: //äROLäAccumulator
00241         case 0x26: //äROLäZeroäPage
00242         case 0x36: //äROLäZeroäPageäX
00243         case 0x2E: //äROLäAbsolute
00244         case 0x3E: //äROLäAbsoluteäX
00245         {
00246             return "ROL";
00247         }
00248         case 0x6A: //äRORäAccumulator
00249         case 0x66: //äRORäZeroäPage
00250         case 0x76: //äRORäZeroäPageäX
00251         case 0x6E: //äRORäAbsolute
00252         case 0x7E: //äRORäAbsoluteäX
00253         {
00254             return "ROR";
00255         }
00256         case 0x40: //äRTIäImplied
00257         {
00258             return "RTI";
00259         }
00260         case 0x60: //äRTSäImplied
00261         {
00262             return "RTS";
00263         }

```

```

00264         case 0xE9: //äSBCäImmediate
00265         case 0xE5: //äSBCäZeroäPage
00266         case 0xF5: //äSBCäZeroäPageäX
00267         case 0xED: //äSBCäAbsolute
00268         case 0xFD: //äSBCäAbsoluteäX
00269         case 0xF9: //äSBCäAbsoluteäY
00270         case 0xE1: //äSBCäIndirectäX
00271         case 0xF1: //äSBCäIndirectäY
00272         {
00273             return "SBC";
00274         }
00275         case 0x38: //äSECäImplied
00276         {
00277             return "SEC";
00278         }
00279         case 0xF8: //äSEDäImplied
00280         {
00281             return "SED";
00282         }
00283         case 0x78: //äSEIäImplied
00284         {
00285             return "SEI";
00286         }
00287         case 0x85: //äSTAäZeroäPage
00288         case 0x95: //äSTAäZeroäPageäX
00289         case 0x8D: //äSTAäAbsolute
00290         case 0x9D: //äSTAäAbsoluteäX
00291         case 0x99: //äSTAäAbsoluteäY
00292         case 0x81: //äSTAäIndirectäX
00293         case 0x91: //äSTAäIndirectäY
00294         {
00295             return "STA";
00296         }
00297         case 0x86: //äSTXäZeroäPage
00298         case 0x96: //äSTXäZeroäPageäY
00299         case 0x8E: //äSTXäAbsolute
00300         {
00301             return "STX";
00302         }
00303         case 0x84: //äSTYäZeroäPage
00304         case 0x94: //äSTYäZeroäPageäX
00305         case 0x8C: //äSTYäAbsolute
00306         {
00307             return "STY";
00308         }
00309         case 0xAA: //äTAXäImplied
00310         {
00311             return "TAX";
00312         }
00313         case 0xA8: //äTAYäImplied
00314         {
00315             return "TAY";
00316         }
00317         case 0xBA: //äTSXäImplied
00318         {
00319             return "TSX";
00320         }
00321         case 0x8A: //äTXAäImplied
00322         {
00323             return "TXA";
00324         }
00325         case 0x9A: //äTXSäImplied
00326         {
00327             return "TXS";
00328         }
00329         case 0x98: //äTYAäImplied
00330         {
00331             return "TYA";
00332         }
00333         default:
00334             throw new InvalidEnumArgumentException(string.Format("A Valid Conversion does not
exist for OpCode {0}", i.ToString("X")));
00335     }
00336 }
00337

```

The documentation for this class was generated from the following file:

- [Hardware/Classes/Utility.cs](#)

6.39 Emulator.Versioning Class Reference

Classes

- class [Product](#)
- class [SettingsFile](#)

6.39.1 Detailed Description

Definition at line 3 of file [Versioning.cs](#).

The documentation for this class was generated from the following file:

- [Emulator/Classes/Versioning.cs](#)

6.40 Emulator.ViewModel.ViewModelLocator Class Reference

This class contains static references to all the view models in the application and provides an entry point for the bindings.

Public Member Functions

- [ViewModelLocator](#) ()
Initializes a new instance of the [ViewModelLocator](#) class.

Static Public Member Functions

- static void [Cleanup](#) ()
The Cleanup Method

Properties

- [MainViewModel Main](#) [get]
The [MainViewModel](#) Instance
- [SettingsViewModel Settings](#) [get]
The [SettingsViewModel](#) Instance
- [MemoryVisualViewModel MemoryVisual](#) [get]
The [SaveFileViewModel](#) Instance

6.40.1 Detailed Description

This class contains static references to all the view models in the application and provides an entry point for the bindings.

Definition at line 24 of file [ViewModelLocator.cs](#).

6.40.2 Constructor & Destructor Documentation

6.40.2.1 ViewModelLocator() `Emulator.ViewModel.ViewModelLocator.ViewModelLocator () [inline]`

Initializes a new instance of the [ViewModelLocator](#) class.

Definition at line 29 of file [ViewModelLocator.cs](#).

```
00030     {
00031         ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);
00032
00033         SimpleIoc.Default.Register<MainViewModel>();
00034         SimpleIoc.Default.Register<SettingsViewModel>();
00035         SimpleIoc.Default.Register<MemoryVisualViewModel>();
00036     }
```

6.40.3 Member Function Documentation

6.40.3.1 Cleanup() `static void Emulator.ViewModel.ViewModelLocator.Cleanup () [inline], [static]`

The Cleanup Method

<todo> Clear the ViewModels </todo>

Definition at line 65 of file [ViewModelLocator.cs](#).

```
00066     {
00067     /// <todo>
00068     /// Clear the ViewModels
00069     /// </todo>
00070 }
```

6.40.4 Property Documentation

6.40.4.1 Main `MainViewModel` `Emulator.ViewModel.ViewModelLocator.Main [get]`

The [MainViewModel](#) Instance

Definition at line 41 of file [ViewModelLocator.cs](#).

```
00042     {
00043         get { return ServiceLocator.Current.GetInstance<MainViewModel>(); }
00044     }
```


6.40.4.2 MemoryVisual [MemoryVisualViewModel](#) Emulator.ViewModel.ViewModelLocator.MemoryVisual [get]

The [SaveFileViewModel](#) Instance

Definition at line 57 of file [ViewModelLocator.cs](#).

```
00058     {
00059         get { return ServiceLocator.Current.GetInstance<MemoryVisualViewModel>(); }
00060     }
```

6.40.4.3 Settings [SettingsViewModel](#) Emulator.ViewModel.ViewModelLocator.Settings [get]

The [SettingsViewModel](#) Instance

Definition at line 49 of file [ViewModelLocator.cs](#).

```
00050     {
00051         get { return ServiceLocator.Current.GetInstance<SettingsViewModel>(); }
00052     }
```

The documentation for this class was generated from the following file:

- Emulator/ViewModel/[ViewModelLocator.cs](#)

6.41 Hardware.W65C02 Class Reference

An implementation of a [W65C02](#) Processor.

Public Member Functions

- [W65C02](#) ()
Default Constructor, Instantiates a new instance of the processor.
- void [Reset](#) ()
Initializes the processor to its default state.
- void [NextStep](#) ()
Performs the next step on the processor
- void [InterruptRequest](#) ()
The InterruptRequest or IRQ
- int [GetCycleCount](#) ()
Gets the Number of Cycles that have elapsed
- void [IncrementCycleCount](#) ()
Increments the Cycle Count, causes a CycleCountIncrementedAction to fire.
- void [ResetCycleCount](#) ()
Resets the Cycle Count back to 0
- void [AslOperation](#) ([AddressingMode](#) addressingMode)
The ASL - Shift Left One Bit (Memory or Accumulator)

Public Attributes

- bool [isRunning](#)
Checks shether the emulated computer is running or not.

Protected Member Functions

- void [SetNegativeFlag](#) (int value)
Sets the IsSignNegative register
- void [SetZeroFlag](#) (int value)
Sets the IsResultZero register
- int [GetAddressByAddressingMode](#) ([AddressingMode](#) addressingMode)
Uses the AddressingMode to return the correct address based on the mode. Note: This method will not increment the program counter for any mode. Note: This method will return an error if called for either the immediate or accumulator modes.
- void [AddWithCarryOperation](#) ([AddressingMode](#) addressingMode)
The ADC - Add Memory to Accumulator with Carry Operation
- void [SubtractWithBorrowOperation](#) ([AddressingMode](#) addressingMode)
The SBC operation. Performs a subtract with carry operation on the accumulator and a value in memory.

Properties

- int [Accumulator](#) [get, protected set]
The Accumulator. This value is implemented as an integer instead of a byte. This is done so we can detect wrapping of the value and set the correct number of cycles.
- int [XRegister](#) [get, private set]
The X Index Register
- int [YRegister](#) [get, private set]
The Y Index Register
- int [CurrentOpCode](#) [get, private set]
The Current Op Code being executed by the system
- [Disassembly CurrentDisassembly](#) [get, private set]
The disassembly of the current operation. This value is only set when the CPU is built in debug mode.
- int [ProgramCounter](#) [get, private set]
Points to the Current Address of the instruction being executed by the system. The PC wraps when the value is greater than 65535, or less than 0.
- int [StackPointer](#) [get, private set]
Points to the Current Position of the Stack. This value is a 00-FF value but is offset to point to the location in memory where the stack resides.
- Action [CycleCountIncrementedAction](#) [get, set]
An external action that occurs when the cycle count is incremented
- bool [CarryFlag](#) [get, protected set]
This is the carry flag. when adding, if the result is greater than 255 or 99 in BCD Mode, then this bit is enabled. In subtraction this is reversed and set to false if a borrow is required IE the result is less than 0
- bool [ZeroFlag](#) [get, private set]
Is true if one of the registers is set to zero.
- bool [DisableInterruptFlag](#) [get, private set]
This determines if Interrupts are currently disabled. This flag is turned on during a reset to prevent an interrupt from occurring during startup/initialization. If this flag is true, then the IRQ pin is ignored.
- bool [DecimalFlag](#) [get, private set]
Binary Coded Decimal Mode is set/cleared via this flag. when this mode is in effect, a byte represents a number from 0-99.
- bool [OverflowFlag](#) [get, protected set]
This property is set when an overflow occurs. An overflow happens if the high bit(7) changes during the operation. Remember that values from 128-256 are negative values as the high bit is set to 1. Examples: 64 + 64 = -128 -128 + -128 = 0
- bool [NegativeFlag](#) [get, private set]

Set to true if the result of an operation is negative in ADC and SBC operations. Remember that 128-256 represent negative numbers when doing signed math. In shift operations the sign holds the carry.

- bool [TriggerNmi](#) [get, set]

Set to true when an NMI should occur

- bool [TriggerIRQ](#) [get, private set]

Set to true when an IRQ has occurred and is being processed by the CPU.

Private Member Functions

- void [ExecuteOpCode](#) ()

Executes an Opcode

- void [MoveProgramCounterByRelativeValue](#) (byte valueToMove)

Moves the ProgramCounter in a given direction based on the value inputted

- byte [PeekStack](#) ()

Returns a the value from the stack without changing the position of the stack pointer

- void [PokeStack](#) (byte value)

Write a value directly to the stack without modifying the Stack Pointer

- byte [ConvertFlagsToByte](#) (bool setBreak)

Coverts the Flags into its byte representation.

- void [SetDisassembly](#) ()

- int [WrapProgramCounter](#) (int value)

- [AddressingMode](#) [GetAddressingMode](#) ()

- void [AndOperation](#) ([AddressingMode](#) addressingMode)

The AND - Compare Memory with Accumulator operation

- void [BranchOperation](#) (bool performBranch)

Performs the different branch operations.

- void [BitOperation](#) ([AddressingMode](#) addressingMode)

The bit operation, does an & comparison between a value in memory and the accumulator

- void [CompareOperation](#) ([AddressingMode](#) addressingMode, int comparisonValue)

The compare operation. This operation compares a value in memory with a value passed into it.

- void [ChangeMemoryByOne](#) ([AddressingMode](#) addressingMode, bool decrement)

Changes a value in memory by 1

- void [ChangeRegisterByOne](#) (bool useXRegister, bool decrement)

Changes a value in either the X or Y register by 1

- void [EorOperation](#) ([AddressingMode](#) addressingMode)

The EOR Operation, Performs an Exclusive OR Operation against the Accumulator and a value in memory

- void [LsrOperation](#) ([AddressingMode](#) addressingMode)

The LSR Operation. Performs a Left shift operation on a value in memory

- void [OrOperation](#) ([AddressingMode](#) addressingMode)

The Or Operation. Performs an Or Operation with the accumulator and a value in memory

- void [RolOperation](#) ([AddressingMode](#) addressingMode)

The ROL operation. Performs a rotate left operation on a value in memory.

- void [RorOperation](#) ([AddressingMode](#) addressingMode)

The ROR operation. Performs a rotate right operation on a value in memory.

- void [PushFlagsOperation](#) ()

The PSP Operation. Pushes the Status Flags to the stack

- void [PullFlagsOperation](#) ()

The PLP Operation. Pull the status flags off the stack on sets the flags accordingly.

- void [JumpToSubRoutineOperation](#) ()

The JSR routine. Jumps to a subroutine.

- void [ReturnFromSubRoutineOperation](#) ()

- *The RTS routine. Called when returning from a subroutine.*
- void [BreakOperation](#) (bool isBrk, int vector)
- *The BRK routine. Called when a BRK occurs.*
- void [ReturnFromInterruptOperation](#) ()
- *The RTI routine. Called when returning from a BRK operation. Note: when called after a BRK operation the Program Counter is not set to the location after the BRK, it is set +1*
- void [ProcessNMI](#) ()
- *This is ran anytime an NMI occurs*
- void [ProcessIRQ](#) ()
- *This is ran anytime an IRQ occurs*

Private Attributes

- readonly ILogger [_logger](#) = LogManager.GetLogger("Processor")
- int [_programCounter](#)
- int [_stackPointer](#)
- int [_cycleCount](#)
- bool [_previousInterrupt](#)
- bool [_interrupt](#)

6.41.1 Detailed Description

An implementation of a [W65C02](#) Processor.

Definition at line 12 of file [W65C02.cs](#).

6.41.2 Constructor & Destructor Documentation

6.41.2.1 W65C02() `Hardware.W65C02.W65C02 () [inline]`

Default Constructor, Instantiates a new instance of the processor.

Definition at line 142 of file [W65C02.cs](#).

```
00143     {
00144         StackPointer = 0x100;
00145         CycleCountIncrementedAction = () => { };
00146     }
```

6.41.3 Member Function Documentation

6.41.3.1 AddWithCarryOperation() `void Hardware.W65C02.AddWithCarryOperation (AddressingMode addressingMode) [inline], [protected]`

The ADC - Add Memory to Accumulator with Carry Operation

Parameters

<i>addressingMode</i>	The addressing mode used to perform this operation.
-----------------------	---

Definition at line 1883 of file [W65C02.cs](#).

```

01884     {
01885         //Accumulator, Carry = Accumulator + ValueInMemoryLocation + Carry
01886         var memoryValue = MemoryMap.Read(GetAddressByAddressingMode(addressingMode));
01887         var newValue = memoryValue + Accumulator + (CarryFlag ? 1 : 0);
01888
01889
01890         OverflowFlag = (((Accumulator ^ newValue) & 0x80) != 0) && (((Accumulator ^ memoryValue) &
0x80) == 0);
01891
01892         if (DecimalFlag)
01893         {
01894             newValue = int.Parse(memoryValue.ToString("x")) + int.Parse(Accumulator.ToString("x"))
+ (CarryFlag ? 1 : 0);
01895
01896             if (newValue > 99)
01897             {
01898                 CarryFlag = true;
01899                 newValue -= 100;
01900             }
01901             else
01902             {
01903                 CarryFlag = false;
01904             }
01905
01906             newValue = (int)Convert.ToInt64(string.Concat("0x", newValue), 16);
01907         }
01908         else
01909         {
01910             if (newValue > 255)
01911             {
01912                 CarryFlag = true;
01913                 newValue -= 256;
01914             }
01915             else
01916             {
01917                 CarryFlag = false;
01918             }
01919         }
01920
01921         SetZeroFlag(newValue);
01922         SetNegativeFlag(newValue);
01923
01924         Accumulator = newValue;
01925     }

```

6.41.3.2 AndOperation() void Hardware.W65C02.AndOperation (
AddressingMode *addressingMode*) [inline], [private]

The AND - Compare Memory with Accumulator operation

Parameters

<i>addressingMode</i>	The addressing mode being used
-----------------------	--------------------------------

Definition at line 1931 of file [W65C02.cs](#).

```

01932     {
01933         Accumulator = MemoryMap.Read(GetAddressByAddressingMode(addressingMode)) & Accumulator;
01934
01935         SetZeroFlag(Accumulator);
01936         SetNegativeFlag(Accumulator);
01937     }

```

6.41.3.3 AslOperation() void Hardware.W65C02.AslOperation (
 AddressingMode addressingMode) [inline]

The ASL - Shift Left One Bit (Memory or Accumulator)

Parameters

<i>addressingMode</i>	The addressing Mode being used
-----------------------	--------------------------------

Definition at line 1943 of file W65C02.cs.

```

01944     {
01945         int value;
01946         var memoryAddress = 0;
01947         if (addressingMode == AddressingMode.Accumulator)
01948         {
01949             MemoryMap.Read(ProgramCounter + 1);
01950             value = Accumulator;
01951         }
01952         else
01953         {
01954             memoryAddress = GetAddressByAddressingMode(addressingMode);
01955             value = MemoryMap.Read(memoryAddress);
01956         }
01957
01958         //Dummy Write
01959         if (addressingMode != AddressingMode.Accumulator)
01960         {
01961             MemoryMap.Write(memoryAddress, (byte)value);
01962         }
01963
01964         //If the 7th bit is set, then we have a carry
01965         CarryFlag = ((value & 0x80) != 0);
01966
01967         //The And here ensures that if the value is greater than 255 it wraps properly.
01968         value = (value << 1) & 0xFE;
01969
01970         SetNegativeFlag(value);
01971         SetZeroFlag(value);
01972
01973
01974         if (addressingMode == AddressingMode.Accumulator)
01975             Accumulator = value;
01976         else
01977         {
01978             MemoryMap.Write(memoryAddress, (byte)value);
01979         }
01980     }

```

6.41.3.4 BitOperation() void Hardware.W65C02.BitOperation (
 AddressingMode addressingMode) [inline], [private]

The bit operation, does an & comparison between a value in memory and the accumulator

Parameters

<i>addressingMode</i>	
-----------------------	--

Definition at line 2003 of file W65C02.cs.

```

02004     {
02005
02006         var memoryValue = MemoryMap.Read(GetAddressByAddressingMode(addressingMode));
02007         var valueToCompare = memoryValue & Accumulator;
02008
02009         OverflowFlag = (memoryValue & 0x40) != 0;
02010
02011         SetNegativeFlag(memoryValue);
02012         SetZeroFlag(valueToCompare);
02013     }

```

6.41.3.5 BranchOperation() void Hardware.W65C02.BranchOperation (
 bool *performBranch*) [inline], [private]

Performs the different branch operations.

Parameters

<i>performBranch</i>	Is a branch required
----------------------	----------------------

Definition at line 1986 of file W65C02.cs.

```

01987     {
01988         var value = MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.Relative));
01989
01990         if (!performBranch)
01991         {
01992             ProgramCounter++;
01993             return;
01994         }
01995
01996         MoveProgramCounterByRelativeValue(value);
01997     }

```

6.41.3.6 BreakOperation() void Hardware.W65C02.BreakOperation (
 bool *isBrk*,
 int *vector*) [inline], [private]

The BRK routine. Called when a BRK occurs.

Definition at line 2349 of file W65C02.cs.

```

02350     {
02351         MemoryMap.Read(++ProgramCounter);
02352
02353         //Put the high value on the stack
02354         //When we RTI the address will be incremented by one, and the address after a break will
not be used.
02355         PokeStack((byte)((ProgramCounter) >> 8) & 0xFF);
02356         StackPointer--;
02357         IncrementCycleCount();
02358
02359         //Put the low value on the stack
02360         PokeStack((byte)((ProgramCounter) & 0xFF));
02361         StackPointer--;
02362         IncrementCycleCount();
02363
02364         //We only set the Break Flag is a Break Occurs
02365         if (isBrk)
02366             PokeStack((byte)(ConvertFlagsToByte(true) | 0x10));
02367         else
02368             PokeStack(ConvertFlagsToByte(false));
02369
02370         StackPointer--;
02371         IncrementCycleCount();
02372
02373         DisableInterruptFlag = true;
02374
02375         ProgramCounter = (MemoryMap.Read(vector + 1) << 8) | MemoryMap.Read(vector);
02376
02377         _previousInterrupt = false;
02378     }

```

6.41.3.7 ChangeMemoryByOne() void Hardware.W65C02.ChangeMemoryByOne (
 AddressingMode *addressingMode*,
 bool *decrement*) [inline], [private]

Changes a value in memory by 1

Parameters

<i>addressingMode</i>	The addressing mode to use
<i>decrement</i>	If the operation is decrementing or incrementing the vaulue by 1

Definition at line 2039 of file [W65C02.cs](#).

```

02040     {
02041         var memoryLocation = GetAddressByAddressingMode(addressingMode);
02042         var memory = MemoryMap.Read(memoryLocation);
02043
02044         MemoryMap.Write(memoryLocation, memory);
02045
02046         if (decrement)
02047             memory -= 1;
02048         else
02049             memory += 1;
02050
02051         SetZeroFlag(memory);
02052         SetNegativeFlag(memory);
02053
02054
02055         MemoryMap.Write(memoryLocation, memory);
02056     }

```

6.41.3.8 ChangeRegisterByOne() void [Hardware.W65C02.ChangeRegisterByOne](#) (
 bool *useXRegister*,
 bool *decrement*) [inline], [private]

Changes a value in either the X or Y register by 1

Parameters

<i>useXRegister</i>	If the operation is using the X or Y register
<i>decrement</i>	If the operation is decrementing or incrementing the vaulue by 1

Definition at line 2063 of file [W65C02.cs](#).

```

02064     {
02065         var value = useXRegister ? XRegister : YRegister;
02066
02067         if (decrement)
02068             value -= 1;
02069         else
02070             value += 1;
02071
02072         if (value < 0x00)
02073             value += 0x100;
02074         else if (value > 0xFF)
02075             value -= 0x100;
02076
02077         SetZeroFlag(value);
02078         SetNegativeFlag(value);
02079         IncrementCycleCount();
02080
02081         if (useXRegister)
02082             XRegister = value;
02083         else
02084             YRegister = value;
02085     }

```

6.41.3.9 CompareOperation() void [Hardware.W65C02.CompareOperation](#) (
[AddressingMode](#) *addressingMode*,
 int *comparisonValue*) [inline], [private]

The compare operation. This operation compares a value in memory with a value passed into it.

Parameters

<i>addressingMode</i>	The addressing mode to use
<i>comparisonValue</i>	The value to compare against memory

Definition at line 2020 of file [W65C02.cs](#).

```

02021     {
02022         var memoryValue = MemoryMap.Read(GetAddressByAddressingMode(addressingMode));
02023         var comparedValue = comparisonValue - memoryValue;
02024
02025         if (comparedValue < 0)
02026             comparedValue += 0x10000;
02027
02028         SetZeroFlag(comparedValue);
02029
02030         CarryFlag = memoryValue <= comparisonValue;
02031         SetNegativeFlag(comparedValue);
02032     }

```

6.41.3.10 ConvertFlagsToByte() byte Hardware.W65C02.ConvertFlagsToByte (bool *setBreak*) [inline], [private]

Converts the Flags into its byte representation.

Parameters

<i>setBreak</i>	Determines if the break flag should be set during conversion. IRQ does not set the flag on the stack, but PHP and BRK do
-----------------	--

Returns

Definition at line 1521 of file [W65C02.cs](#).

```

01522     {
01523         return (byte)((CarryFlag ? 0x01 : 0) + (ZeroFlag ? 0x02 : 0) + (DisableInterruptFlag ?
01524             0x04 : 0) + (DecimalFlag ? 8 : 0) + (setBreak ? 0x10 : 0) + 0x20 + (OverflowFlag ? 0x40 : 0)
01525         + (NegativeFlag ? 0x80 : 0));
01526     }

```

6.41.3.11 EorOperation() void Hardware.W65C02.EorOperation (AddressingMode *addressingMode*) [inline], [private]

The EOR Operation, Performs an Exclusive OR Operation against the Accumulator and a value in memory

Parameters

<i>addressingMode</i>	The addressing mode to use
-----------------------	----------------------------

Definition at line 2091 of file [W65C02.cs](#).

```

02092     {
02093         Accumulator = Accumulator ^ MemoryMap.Read(GetAddressByAddressingMode(addressingMode));

```

```

02094
02095         SetNegativeFlag(Accumulator);
02096         SetZeroFlag(Accumulator);
02097     }

```

6.41.3.12 ExecuteOpCode() void Hardware.W65C02.ExecuteOpCode () [inline], [private]

Executes an Opcode

Definition at line 238 of file W65C02.cs.

```

00239     {
00240         //The x+ cycles denotes that if a page wrap occurs, then an additional cycle is consumed.
00241         //The x++ cycles denotes that 1 cycle is added when a branch occurs and it on the same
        page, and two cycles are added if its on a different page./
00242         //This is handled inside the GetValueFromMemory Method
00243         switch (CurrentOpCode)
00244         {
00245             #region Add / Subtract Operations
00246             //ADC Add With Carry, Immediate, 2 Bytes, 2 Cycles
00247             case 0x69:
00248             {
00249                 AddWithCarryOperation(AddressingMode.Immediate);
00250                 break;
00251             }
00252             //ADC Add With Carry, Zero Page, 2 Bytes, 3 Cycles
00253             case 0x65:
00254             {
00255                 AddWithCarryOperation(AddressingMode.ZeroPage);
00256                 break;
00257             }
00258             //ADC Add With Carry, Zero Page X, 2 Bytes, 4 Cycles
00259             case 0x75:
00260             {
00261                 AddWithCarryOperation(AddressingMode.ZeroPageX);
00262                 break;
00263             }
00264             //ADC Add With Carry, Absolute, 3 Bytes, 4 Cycles
00265             case 0x6D:
00266             {
00267                 AddWithCarryOperation(AddressingMode.Absolute);
00268                 break;
00269             }
00270             //ADC Add With Carry, Absolute X, 3 Bytes, 4+ Cycles
00271             case 0x7D:
00272             {
00273                 AddWithCarryOperation(AddressingMode.AbsoluteX);
00274                 break;
00275             }
00276             //ADC Add With Carry, Absolute Y, 3 Bytes, 4+ Cycles
00277             case 0x79:
00278             {
00279                 AddWithCarryOperation(AddressingMode.AbsoluteY);
00280                 break;
00281             }
00282             //ADC Add With Carry, Indexed Indirect, 2 Bytes, 6 Cycles
00283             case 0x61:
00284             {
00285                 AddWithCarryOperation(AddressingMode.IndirectX);
00286                 break;
00287             }
00288             //ADC Add With Carry, Indexed Indirect, 2 Bytes, 5+ Cycles
00289             case 0x71:
00290             {
00291                 AddWithCarryOperation(AddressingMode.IndirectY);
00292                 break;
00293             }
00294             //SBC Subtract with Borrow, Immediate, 2 Bytes, 2 Cycles
00295             case 0xE9:
00296             {
00297                 SubtractWithBorrowOperation(AddressingMode.Immediate);
00298                 break;
00299             }
00300             //SBC Subtract with Borrow, Zero Page, 2 Bytes, 3 Cycles
00301             case 0xE5:
00302             {
00303                 SubtractWithBorrowOperation(AddressingMode.ZeroPage);
00304                 break;
00305             }
00306             //SBC Subtract with Borrow, Zero Page X, 2 Bytes, 4 Cycles

```

```

00307         case 0xF5:
00308             {
00309                 SubtractWithBorrowOperation(AddressingMode.ZeroPageX);
00310                 break;
00311             }
00312         //SBC Subtract with Borrow, Absolute, 3 Bytes, 4 Cycles
00313         case 0xED:
00314             {
00315                 SubtractWithBorrowOperation(AddressingMode.Absolute);
00316                 break;
00317             }
00318         //SBC Subtract with Borrow, Absolute X, 3 Bytes, 4+ Cycles
00319         case 0xFD:
00320             {
00321                 SubtractWithBorrowOperation(AddressingMode.AbsoluteX);
00322                 break;
00323             }
00324         //SBC Subtract with Borrow, Absolute Y, 3 Bytes, 4+ Cycles
00325         case 0xF9:
00326             {
00327                 SubtractWithBorrowOperation(AddressingMode.AbsoluteY);
00328                 break;
00329             }
00330         //SBC Subtract with Borrow, Indexed Indirect, 2 Bytes, 6 Cycles
00331         case 0xE1:
00332             {
00333                 SubtractWithBorrowOperation(AddressingMode.IndirectX);
00334                 break;
00335             }
00336         //SBC Subtract with Borrow, Indexed Indirect, 2 Bytes, 5+ Cycles
00337         case 0xF1:
00338             {
00339                 SubtractWithBorrowOperation(AddressingMode.IndirectY);
00340                 break;
00341             }
00342     #endregion
00343
00344     #region Branch Operations
00345         //BCC Branch if Carry is Clear, Relative, 2 Bytes, 2++ Cycles
00346         case 0x90:
00347             {
00348                 BranchOperation(!CarryFlag);
00349                 break;
00350             }
00351         //BCS Branch if Carry is Set, Relative, 2 Bytes, 2++ Cycles
00352         case 0xB0:
00353             {
00354                 BranchOperation(CarryFlag);
00355                 break;
00356             }
00357         //BEQ Branch if Zero is Set, Relative, 2 Bytes, 2++ Cycles
00358         case 0xF0:
00359             {
00360                 BranchOperation(ZeroFlag);
00361                 break;
00362             }
00363         // BMI Branch if Negative Set
00364         case 0x30:
00365             {
00366                 BranchOperation(NegativeFlag);
00367                 break;
00368             }
00369         //BNE Branch if Zero is Not Set, Relative, 2 Bytes, 2++ Cycles
00370         case 0xD0:
00371             {
00372                 BranchOperation(!ZeroFlag);
00373                 break;
00374             }
00375         // BPL Branch if Negative Clear, 2 Bytes, 2++ Cycles
00376         case 0x10:
00377             {
00378                 BranchOperation(!NegativeFlag);
00379                 break;
00380             }
00381         // BVC Branch if Overflow Clear, 2 Bytes, 2++ Cycles
00382         case 0x50:
00383             {
00384                 BranchOperation(!OverflowFlag);
00385                 break;
00386             }
00387         // BVS Branch if Overflow Set, 2 Bytes, 2++ Cycles
00388         case 0x70:
00389             {
00390                 BranchOperation(OverflowFlag);
00391                 break;
00392             }
00393

```

```

00394     }
00395 #endregion
00396
00397 #region BitWise Comparison Operations
00398 //AND Compare Memory with Accumulator, Immediate, 2 Bytes, 2 Cycles
00399 case 0x29:
00400 {
00401     AndOperation(AddressingMode.Immediate);
00402     break;
00403 }
00404 //AND Compare Memory with Accumulator, Zero Page, 2 Bytes, 3 Cycles
00405 case 0x25:
00406 {
00407     AndOperation(AddressingMode.ZeroPage);
00408     break;
00409 }
00410 //AND Compare Memory with Accumulator, Zero PageX, 2 Bytes, 3 Cycles
00411 case 0x35:
00412 {
00413     AndOperation(AddressingMode.ZeroPageX);
00414     break;
00415 }
00416 //AND Compare Memory with Accumulator, Absolute, 3 Bytes, 4 Cycles
00417 case 0x2D:
00418 {
00419     AndOperation(AddressingMode.Absolute);
00420     break;
00421 }
00422 //AND Compare Memory with Accumulator, AbsoluteX 3 Bytes, 4+ Cycles
00423 case 0x3D:
00424 {
00425     AndOperation(AddressingMode.AbsoluteX);
00426     break;
00427 }
00428 //AND Compare Memory with Accumulator, AbsoluteY, 3 Bytes, 4+ Cycles
00429 case 0x39:
00430 {
00431     AndOperation(AddressingMode.AbsoluteY);
00432     break;
00433 }
00434 //AND Compare Memory with Accumulator, IndexedIndirect, 2 Bytes, 6 Cycles
00435 case 0x21:
00436 {
00437     AndOperation(AddressingMode.IndirectX);
00438     break;
00439 }
00440 //AND Compare Memory with Accumulator, IndirectIndexed, 2 Bytes, 5 Cycles
00441 case 0x31:
00442 {
00443     AndOperation(AddressingMode.IndirectY);
00444     break;
00445 }
00446 //BIT Compare Memory with Accumulator, Zero Page, 2 Bytes, 3 Cycles
00447 case 0x24:
00448 {
00449     BitOperation(AddressingMode.ZeroPage);
00450     break;
00451 }
00452 //BIT Compare Memory with Accumulator, Absolute, 2 Bytes, 4 Cycles
00453 case 0x2C:
00454 {
00455     BitOperation(AddressingMode.Absolute);
00456     break;
00457 }
00458 //EOR Exclusive OR Memory with Accumulator, Immediate, 2 Bytes, 2 Cycles
00459 case 0x49:
00460 {
00461     EorOperation(AddressingMode.Immediate);
00462     break;
00463 }
00464 //EOR Exclusive OR Memory with Accumulator, Zero Page, 2 Bytes, 3 Cycles
00465 case 0x45:
00466 {
00467     EorOperation(AddressingMode.ZeroPage);
00468     break;
00469 }
00470 //EOR Exclusive OR Memory with Accumulator, Zero Page X, 2 Bytes, 4 Cycles
00471 case 0x55:
00472 {
00473     EorOperation(AddressingMode.ZeroPageX);
00474     break;
00475 }
00476 //EOR Exclusive OR Memory with Accumulator, Absolute, 3 Bytes, 4 Cycles
00477 case 0x4D:
00478 {
00479     EorOperation(AddressingMode.Absolute);
00480     break;

```

```

00481     }
00482     //EOR Exclusive OR Memory with Accumulator, Absolute X, 3 Bytes, 4+ Cycles
00483     case 0x5D:
00484     {
00485         EorOperation(AddressingMode.AbsoluteX);
00486         break;
00487     }
00488     //EOR Exclusive OR Memory with Accumulator, Absolute Y, 3 Bytes, 4+ Cycles
00489     case 0x59:
00490     {
00491         EorOperation(AddressingMode.AbsoluteY);
00492         break;
00493     }
00494     //EOR Exclusive OR Memory with Accumulator, IndexedIndirect, 2 Bytes 6 Cycles
00495     case 0x41:
00496     {
00497         EorOperation(AddressingMode.IndirectX);
00498         break;
00499     }
00500     //EOR Exclusive OR Memory with Accumulator, IndirectIndexed, 2 Bytes 5 Cycles
00501     case 0x51:
00502     {
00503         EorOperation(AddressingMode.IndirectY);
00504         break;
00505     }
00506     //ORA Compare Memory with Accumulator, Immediate, 2 Bytes, 2 Cycles
00507     case 0x09:
00508     {
00509         OrOperation(AddressingMode.Immediate);
00510         break;
00511     }
00512     //ORA Compare Memory with Accumulator, Zero Page, 2 Bytes, 2 Cycles
00513     case 0x05:
00514     {
00515         OrOperation(AddressingMode.ZeroPage);
00516         break;
00517     }
00518     //ORA Compare Memory with Accumulator, Zero PageX, 2 Bytes, 4 Cycles
00519     case 0x15:
00520     {
00521         OrOperation(AddressingMode.ZeroPageX);
00522         break;
00523     }
00524     //ORA Compare Memory with Accumulator, Absolute, 3 Bytes, 4 Cycles
00525     case 0x0D:
00526     {
00527         OrOperation(AddressingMode.Absolute);
00528         break;
00529     }
00530     //ORA Compare Memory with Accumulator, AbsoluteX 3 Bytes, 4+ Cycles
00531     case 0x1D:
00532     {
00533         OrOperation(AddressingMode.AbsoluteX);
00534         break;
00535     }
00536     //ORA Compare Memory with Accumulator, AbsoluteY, 3 Bytes, 4+ Cycles
00537     case 0x19:
00538     {
00539         OrOperation(AddressingMode.AbsoluteY);
00540         break;
00541     }
00542     //ORA Compare Memory with Accumulator, IndexedIndirect, 2 Bytes, 6 Cycles
00543     case 0x01:
00544     {
00545         OrOperation(AddressingMode.IndirectX);
00546         break;
00547     }
00548     //ORA Compare Memory with Accumulator, IndirectIndexed, 2 Bytes, 5 Cycles
00549     case 0x11:
00550     {
00551         OrOperation(AddressingMode.IndirectY);
00552         break;
00553     }
00554 #endregion
00555
00556 #region Clear Flag Operations
00557     //CLC Clear Carry Flag, Implied, 1 Byte, 2 Cycles
00558     case 0x18:
00559     {
00560         CarryFlag = false;
00561         IncrementCycleCount();
00562         break;
00563     }
00564     //CLD Clear Decimal Flag, Implied, 1 Byte, 2 Cycles
00565     case 0xD8:
00566     {
00567         DecimalFlag = false;

```

```

00568             IncrementCycleCount();
00569             break;
00570         }
00571     }
00572     //CLI Clear Interrupt Flag, Implied, 1 Byte, 2 Cycles
00573     case 0x58:
00574     {
00575         DisableInterruptFlag = false;
00576         IncrementCycleCount();
00577         break;
00578     }
00579 }
00580 //CLV Clear Overflow Flag, Implied, 1 Byte, 2 Cycles
00581 case 0xB8:
00582 {
00583     OverflowFlag = false;
00584     IncrementCycleCount();
00585     break;
00586 }
00587 }
00588 #endregion
00589
00590 #region Compare Operations
00591 //CMP Compare Accumulator with Memory, Immediate, 2 Bytes, 2 Cycles
00592 case 0xC9:
00593 {
00594     CompareOperation(AddressingMode.Immediate, Accumulator);
00595     break;
00596 }
00597 //CMP Compare Accumulator with Memory, Zero Page, 2 Bytes, 3 Cycles
00598 case 0xC5:
00599 {
00600     CompareOperation(AddressingMode.ZeroPage, Accumulator);
00601     break;
00602 }
00603 //CMP Compare Accumulator with Memory, Zero Page X, 2 Bytes, 4 Cycles
00604 case 0xD5:
00605 {
00606     CompareOperation(AddressingMode.ZeroPageX, Accumulator);
00607     break;
00608 }
00609 //CMP Compare Accumulator with Memory, Absolute, 3 Bytes, 4 Cycles
00610 case 0xCD:
00611 {
00612     CompareOperation(AddressingMode.Absolute, Accumulator);
00613     break;
00614 }
00615 //CMP Compare Accumulator with Memory, Absolute X, 2 Bytes, 4 Cycles
00616 case 0xDD:
00617 {
00618     CompareOperation(AddressingMode.AbsoluteX, Accumulator);
00619     break;
00620 }
00621 //CMP Compare Accumulator with Memory, Absolute Y, 2 Bytes, 4 Cycles
00622 case 0xD9:
00623 {
00624     CompareOperation(AddressingMode.AbsoluteY, Accumulator);
00625     break;
00626 }
00627 //CMP Compare Accumulator with Memory, Indirect X, 2 Bytes, 6 Cycles
00628 case 0xC1:
00629 {
00630     CompareOperation(AddressingMode.IndirectX, Accumulator);
00631     break;
00632 }
00633 //CMP Compare Accumulator with Memory, Indirect Y, 2 Bytes, 5 Cycles
00634 case 0xD1:
00635 {
00636     CompareOperation(AddressingMode.IndirectY, Accumulator);
00637     break;
00638 }
00639 //CPX Compare Accumulator with X Register, Immediate, 2 Bytes, 2 Cycles
00640 case 0xE0:
00641 {
00642     CompareOperation(AddressingMode.Immediate, XRegister);
00643     break;
00644 }
00645 //CPX Compare Accumulator with X Register, Zero Page, 2 Bytes, 3 Cycles
00646 case 0xE4:
00647 {
00648     CompareOperation(AddressingMode.ZeroPage, XRegister);
00649     break;
00650 }
00651 //CPX Compare Accumulator with X Register, Absolute, 3 Bytes, 4 Cycles
00652 case 0xEC:
00653 {
00654     CompareOperation(AddressingMode.Absolute, XRegister);

```

```

00655         break;
00656     }
00657     //CPY Compare Accumulator with Y Register, Immediate, 2 Bytes, 2 Cycles
00658     case 0xC0:
00659     {
00660         CompareOperation(AddressingMode.Immediate, YRegister);
00661         break;
00662     }
00663     //CPY Compare Accumulator with Y Register, Zero Page, 2 Bytes, 3 Cycles
00664     case 0xC4:
00665     {
00666         CompareOperation(AddressingMode.ZeroPage, YRegister);
00667         break;
00668     }
00669     //CPY Compare Accumulator with Y Register, Absolute, 3 Bytes, 4 Cycles
00670     case 0xCC:
00671     {
00672         CompareOperation(AddressingMode.Absolute, YRegister);
00673         break;
00674     }
00675 #endregion
00676
00677 #region Increment/Decrement Operations
00678     //DEC Decrement Memory by One, Zero Page, 2 Bytes, 5 Cycles
00679     case 0xC6:
00680     {
00681         ChangeMemoryByOne(AddressingMode.ZeroPage, true);
00682         break;
00683     }
00684     //DEC Decrement Memory by One, Zero Page X, 2 Bytes, 6 Cycles
00685     case 0xD6:
00686     {
00687         ChangeMemoryByOne(AddressingMode.ZeroPageX, true);
00688         break;
00689     }
00690     //DEC Decrement Memory by One, Absolute, 3 Bytes, 6 Cycles
00691     case 0xCE:
00692     {
00693         ChangeMemoryByOne(AddressingMode.Absolute, true);
00694         break;
00695     }
00696     //DEC Decrement Memory by One, Absolute X, 3 Bytes, 7 Cycles
00697     case 0xDE:
00698     {
00699         ChangeMemoryByOne(AddressingMode.AbsoluteX, true);
00700         IncrementCycleCount();
00701         break;
00702     }
00703     //DEX Decrement X Register by One, Implied, 1 Bytes, 2 Cycles
00704     case 0xCA:
00705     {
00706         ChangeRegisterByOne(true, true);
00707         break;
00708     }
00709     //DEY Decrement Y Register by One, Implied, 1 Bytes, 2 Cycles
00710     case 0x88:
00711     {
00712         ChangeRegisterByOne(false, true);
00713         break;
00714     }
00715     //INC Increment Memory by One, Zero Page, 2 Bytes, 5 Cycles
00716     case 0xE6:
00717     {
00718         ChangeMemoryByOne(AddressingMode.ZeroPage, false);
00719         break;
00720     }
00721     //INC Increment Memory by One, Zero Page X, 2 Bytes, 6 Cycles
00722     case 0xF6:
00723     {
00724         ChangeMemoryByOne(AddressingMode.ZeroPageX, false);
00725         break;
00726     }
00727     //INC Increment Memory by One, Absolute, 3 Bytes, 6 Cycles
00728     case 0xEE:
00729     {
00730         ChangeMemoryByOne(AddressingMode.Absolute, false);
00731         break;
00732     }
00733     //INC Increment Memory by One, Absolute X, 3 Bytes, 7 Cycles
00734     case 0xFE:
00735     {
00736         ChangeMemoryByOne(AddressingMode.AbsoluteX, false);
00737         IncrementCycleCount();
00738         break;
00739     }
00740     //INX Increment X Register by One, Implied, 1 Bytes, 2 Cycles
00741     case 0xE8:

```



```

00742         {
00743             ChangeRegisterByOne(true, false);
00744             break;
00745         }
00746         //INY Increment Y Register by One, Implied, 1 Bytes, 2 Cycles
00747         case 0xC8:
00748         {
00749             ChangeRegisterByOne(false, false);
00750             break;
00751         }
00752     #endregion
00753
00754     #region GOTO and GOSUB Operations
00755     //JMP Jump to New Location, Absolute 3 Bytes, 3 Cycles
00756     case 0x4C:
00757     {
00758         ProgramCounter = GetAddressByAddressingMode(AddressingMode.Absolute);
00759         break;
00760     }
00761     //JMP Jump to New Location, Indirect 3 Bytes, 5 Cycles
00762     case 0x6C:
00763     {
00764         ProgramCounter = GetAddressByAddressingMode(AddressingMode.Absolute);
00765
00766         if ((ProgramCounter & 0xFF) == 0xFF)
00767         {
00768             //Get the first half of the address
00769             int address = MemoryMap.Read(ProgramCounter);
00770
00771             //Get the second half of the address, due to the issue with page boundary
00772             //it reads from the wrong location!
00773             address += 256 * MemoryMap.Read(ProgramCounter - 255);
00774             ProgramCounter = address;
00775         }
00776         else
00777         {
00778             ProgramCounter = GetAddressByAddressingMode(AddressingMode.Absolute);
00779         }
00780         break;
00781     }
00782     //JSR Jump to SubRoutine, Absolute, 3 Bytes, 6 Cycles
00783     case 0x20:
00784     {
00785         JumpToSubRoutineOperation();
00786         break;
00787     }
00788     //BRK Simulate IRQ, Implied, 1 Byte, 7 Cycles
00789     case 0x00:
00790     {
00791         BreakOperation(true, 0xFFFE);
00792         break;
00793     }
00794     //RTI Return From Interrupt, Implied, 1 Byte, 6 Cycles
00795     case 0x40:
00796     {
00797         ReturnFromInterruptOperation();
00798         break;
00799     }
00800     //RTS Return From Subroutine, Implied, 1 Byte, 6 Cycles
00801     case 0x60:
00802     {
00803         ReturnFromSubRoutineOperation();
00804         break;
00805     }
00806 #endregion
00807
00808     #region Load Value From Memory Operations
00809     //LDA Load Accumulator with Memory, Immediate, 2 Bytes, 2 Cycles
00810     case 0xA9:
00811     {
00812         Accumulator =
00813         MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.Immediate));
00814         SetZeroFlag(Accumulator);
00815         SetNegativeFlag(Accumulator);
00816         break;
00817     }
00818     //LDA Load Accumulator with Memory, Zero Page, 2 Bytes, 3 Cycles
00819     case 0xA5:
00820     {
00821         Accumulator =
00822         MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.ZeroPage));
00823         SetZeroFlag(Accumulator);
00824         SetNegativeFlag(Accumulator);
00825         break;
00826     }
00827     //LDA Load Accumulator with Memory, Zero Page X, 2 Bytes, 4 Cycles

```

```

00826         case 0xB5:
00827         {
00828             Accumulator =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.ZeroPageX));
00829             SetZeroFlag(Accumulator);
00830             SetNegativeFlag(Accumulator);
00831             break;
00832         }
00833         //LDA Load Accumulator with Memory, Absolute, 3 Bytes, 4 Cycles
00834         case 0xAD:
00835         {
00836             Accumulator =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.Absolute));
00837             SetZeroFlag(Accumulator);
00838             SetNegativeFlag(Accumulator);
00839             break;
00840         }
00841         //LDA Load Accumulator with Memory, Absolute X, 3 Bytes, 4+ Cycles
00842         case 0xBD:
00843         {
00844             Accumulator =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.AbsoluteX));
00845             SetZeroFlag(Accumulator);
00846             SetNegativeFlag(Accumulator);
00847             break;
00848         }
00849         //LDA Load Accumulator with Memory, Absolute Y, 3 Bytes, 4+ Cycles
00850         case 0xB9:
00851         {
00852             Accumulator =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.AbsoluteY));
00853             SetZeroFlag(Accumulator);
00854             SetNegativeFlag(Accumulator);
00855             break;
00856         }
00857         //LDA Load Accumulator with Memory, Index Indirect, 2 Bytes, 6 Cycles
00858         case 0xA1:
00859         {
00860             Accumulator =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.IndirectX));
00861             SetZeroFlag(Accumulator);
00862             SetNegativeFlag(Accumulator);
00863             break;
00864         }
00865         //LDA Load Accumulator with Memory, Indirect Index, 2 Bytes, 5+ Cycles
00866         case 0xB1:
00867         {
00868             Accumulator =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.IndirectY));
00869             SetZeroFlag(Accumulator);
00870             SetNegativeFlag(Accumulator);
00871             break;
00872         }
00873         //LDX Load X with memory, Immediate, 2 Bytes, 2 Cycles
00874         case 0xA2:
00875         {
00876             XRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.Immediate));
00877             SetZeroFlag(XRegister);
00878             SetNegativeFlag(XRegister);
00879             break;
00880         }
00881         //LDX Load X with memory, Zero Page, 2 Bytes, 3 Cycles
00882         case 0xA6:
00883         {
00884             XRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.ZeroPage));
00885             SetZeroFlag(XRegister);
00886             SetNegativeFlag(XRegister);
00887             break;
00888         }
00889         //LDX Load X with memory, Zero Page Y, 2 Bytes, 4 Cycles
00890         case 0xB6:
00891         {
00892             XRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.ZeroPageY));
00893             SetZeroFlag(XRegister);
00894             SetNegativeFlag(XRegister);
00895             break;
00896         }
00897         //LDX Load X with memory, Absolute, 3 Bytes, 4 Cycles
00898         case 0xAE:
00899         {
00900             XRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.Absolute));
00901             SetZeroFlag(XRegister);
00902             SetNegativeFlag(XRegister);

```

```

00903             break;
00904         }
00905         //LDX Load X with memory, Absolute Y, 3 Bytes, 4+ Cycles
00906         case 0xBE:
00907         {
00908             XRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.AbsoluteY));
00909             SetZeroFlag(XRegister);
00910             SetNegativeFlag(XRegister);
00911             break;
00912         }
00913         //LDY Load Y with memory, Immediate, 2 Bytes, 2 Cycles
00914         case 0xA0:
00915         {
00916             YRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.Immediate));
00917             SetZeroFlag(YRegister);
00918             SetNegativeFlag(YRegister);
00919             break;
00920         }
00921         //LDY Load Y with memory, Zero Page, 2 Bytes, 3 Cycles
00922         case 0xA4:
00923         {
00924             YRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.ZeroPage));
00925             SetZeroFlag(YRegister);
00926             SetNegativeFlag(YRegister);
00927             break;
00928         }
00929         //LDY Load Y with memory, Zero Page X, 2 Bytes, 4 Cycles
00930         case 0xB4:
00931         {
00932             YRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.ZeroPageX));
00933             SetZeroFlag(YRegister);
00934             SetNegativeFlag(YRegister);
00935             break;
00936         }
00937         //LDY Load Y with memory, Absolute, 3 Bytes, 4 Cycles
00938         case 0xAC:
00939         {
00940             YRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.Absolute));
00941             SetZeroFlag(YRegister);
00942             SetNegativeFlag(YRegister);
00943             break;
00944         }
00945         //LDY Load Y with memory, Absolute X, 3 Bytes, 4+ Cycles
00946         case 0xBC:
00947         {
00948             YRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.AbsoluteX));
00949             SetZeroFlag(YRegister);
00950             SetNegativeFlag(YRegister);
00951             break;
00952         }
00953         #endregion
00954
00955         #region Push/Pull Stack
00956         //PHA Push Accumulator onto Stack, Implied, 1 Byte, 3 Cycles
00957         case 0x48:
00958         {
00959             MemoryMap.Read(ProgramCounter + 1);
00960
00961             PokeStack((byte)Accumulator);
00962             StackPointer--;
00963             IncrementCycleCount();
00964             break;
00965         }
00966
00967         //PHP Push Flags onto Stack, Implied, 1 Byte, 3 Cycles
00968         case 0x08:
00969         {
00970             MemoryMap.Read(ProgramCounter + 1);
00971
00972             PushFlagsOperation();
00973             StackPointer--;
00974             IncrementCycleCount();
00975             break;
00976         }
00977         //PLA Pull Accumulator from Stack, Implied, 1 Byte, 4 Cycles
00978         case 0x68:
00979         {
00980             MemoryMap.Read(ProgramCounter + 1);
00981             StackPointer++;
00982             IncrementCycleCount();
00983

```

```

00984         Accumulator = PeekStack();
00985         SetNegativeFlag(Accumulator);
00986         SetZeroFlag(Accumulator);
00987
00988         IncrementCycleCount();
00989         break;
00990     }
00991     //PLP Pull Flags from Stack, Implied, 1 Byte, 4 Cycles
00992     case 0x28:
00993     {
00994         MemoryMap.Read(ProgramCounter + 1);
00995
00996         StackPointer++;
00997         IncrementCycleCount();
00998
00999         PullFlagsOperation();
01000
01001         IncrementCycleCount();
01002         break;
01003     }
01004     //TSX Transfer Stack Pointer to X Register, 1 Bytes, 2 Cycles
01005     case 0xBA:
01006     {
01007         XRegister = StackPointer;
01008
01009         SetNegativeFlag(XRegister);
01010         SetZeroFlag(XRegister);
01011         IncrementCycleCount();
01012         break;
01013     }
01014     //TXS Transfer X Register to Stack Pointer, 1 Bytes, 2 Cycles
01015     case 0x9A:
01016     {
01017         StackPointer = (byte)XRegister;
01018         IncrementCycleCount();
01019         break;
01020     }
01021 #endregion
01022
01023 #region Set Flag Operations
01024     //SEC Set Carry, Implied, 1 Bytes, 2 Cycles
01025     case 0x38:
01026     {
01027         CarryFlag = true;
01028         IncrementCycleCount();
01029         break;
01030     }
01031     //SED Set Interrupt, Implied, 1 Bytes, 2 Cycles
01032     case 0xF8:
01033     {
01034         DecimalFlag = true;
01035         IncrementCycleCount();
01036         break;
01037     }
01038     //SEI Set Interrupt, Implied, 1 Bytes, 2 Cycles
01039     case 0x78:
01040     {
01041         DisableInterruptFlag = true;
01042         IncrementCycleCount();
01043         break;
01044     }
01045 #endregion
01046
01047 #region Shift/Rotate Operations
01048     //ASL Shift Left 1 Bit Memory or Accumulator, Accumulator, 1 Bytes, 2 Cycles
01049     case 0x0A:
01050     {
01051         AslOperation(AddressingMode.Accumulator);
01052         break;
01053     }
01054     //ASL Shift Left 1 Bit Memory or Accumulator, Zero Page, 2 Bytes, 5 Cycles
01055     case 0x06:
01056     {
01057         AslOperation(AddressingMode.ZeroPage);
01058         break;
01059     }
01060     //ASL Shift Left 1 Bit Memory or Accumulator, Zero PageX, 2 Bytes, 6 Cycles
01061     case 0x16:
01062     {
01063         AslOperation(AddressingMode.ZeroPageX);
01064         break;
01065     }
01066     //ASL Shift Left 1 Bit Memory or Accumulator, Absolute, 3 Bytes, 6 Cycles
01067     case 0x0E:
01068     {
01069         AslOperation(AddressingMode.Absolute);
01070         break;

```

```

01071     }
01072     //ASL Shift Left 1 Bit Memory or Accumulator, AbsoluteX, 3 Bytes, 7 Cycles
01073     case 0x1E:
01074     {
01075         AslOperation(AddressingMode.AbsoluteX);
01076         IncrementCycleCount();
01077         break;
01078     }
01079     //LSR Shift Left 1 Bit Memory or Accumulator, Accumulator, 1 Bytes, 2 Cycles
01080     case 0x4A:
01081     {
01082         LsrOperation(AddressingMode.Accumulator);
01083         break;
01084     }
01085     //LSR Shift Left 1 Bit Memory or Accumulator, Zero Page, 2 Bytes, 5 Cycles
01086     case 0x46:
01087     {
01088         LsrOperation(AddressingMode.ZeroPage);
01089         break;
01090     }
01091     //LSR Shift Left 1 Bit Memory or Accumulator, Zero PageX, 2 Bytes, 6 Cycles
01092     case 0x56:
01093     {
01094         LsrOperation(AddressingMode.ZeroPageX);
01095         break;
01096     }
01097     //LSR Shift Left 1 Bit Memory or Accumulator, Absolute, 3 Bytes, 6 Cycles
01098     case 0x4E:
01099     {
01100         LsrOperation(AddressingMode.Absolute);
01101         break;
01102     }
01103     //LSR Shift Left 1 Bit Memory or Accumulator, AbsoluteX, 3 Bytes, 7 Cycles
01104     case 0x5E:
01105     {
01106         LsrOperation(AddressingMode.AbsoluteX);
01107         IncrementCycleCount();
01108         break;
01109     }
01110     //ROL Rotate Left 1 Bit Memory or Accumulator, Accumulator, 1 Bytes, 2 Cycles
01111     case 0x2A:
01112     {
01113         RolOperation(AddressingMode.Accumulator);
01114         break;
01115     }
01116     //ROL Rotate Left 1 Bit Memory or Accumulator, Zero Page, 2 Bytes, 5 Cycles
01117     case 0x26:
01118     {
01119         RolOperation(AddressingMode.ZeroPage);
01120         break;
01121     }
01122     //ROL Rotate Left 1 Bit Memory or Accumulator, Zero PageX, 2 Bytes, 6 Cycles
01123     case 0x36:
01124     {
01125         RolOperation(AddressingMode.ZeroPageX);
01126         break;
01127     }
01128     //ROL Rotate Left 1 Bit Memory or Accumulator, Absolute, 3 Bytes, 6 Cycles
01129     case 0x2E:
01130     {
01131         RolOperation(AddressingMode.Absolute);
01132         break;
01133     }
01134     //ROL Rotate Left 1 Bit Memory or Accumulator, AbsoluteX, 3 Bytes, 7 Cycles
01135     case 0x3E:
01136     {
01137         RolOperation(AddressingMode.AbsoluteX);
01138         IncrementCycleCount();
01139         break;
01140     }
01141     //ROR Rotate Right 1 Bit Memory or Accumulator, Accumulator, 1 Bytes, 2 Cycles
01142     case 0x6A:
01143     {
01144         RorOperation(AddressingMode.Accumulator);
01145         break;
01146     }
01147     //ROR Rotate Right 1 Bit Memory or Accumulator, Zero Page, 2 Bytes, 5 Cycles
01148     case 0x66:
01149     {
01150         RorOperation(AddressingMode.ZeroPage);
01151         break;
01152     }
01153     //ROR Rotate Right 1 Bit Memory or Accumulator, Zero PageX, 2 Bytes, 6 Cycles
01154     case 0x76:
01155     {
01156         RorOperation(AddressingMode.ZeroPageX);
01157         break;

```

```

01158         }
01159         //ROR Rotate Right 1 Bit Memory or Accumulator, Absolute, 3 Bytes, 6 Cycles
01160         case 0x6E:
01161         {
01162             RorOperation(AddressingMode.Absolute);
01163             break;
01164         }
01165         //ROR Rotate Right 1 Bit Memory or Accumulator, AbsoluteX, 3 Bytes, 7 Cycles
01166         case 0x7E:
01167         {
01168             RorOperation(AddressingMode.AbsoluteX);
01169             IncrementCycleCount();
01170             break;
01171         }
01172     #endregion
01173
01174     #region Store Value In Memory Operations
01175     //STA Store Accumulator In Memory, Zero Page, 2 Bytes, 3 Cycles
01176     case 0x85:
01177     {
01178         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.ZeroPage),
01179             (byte)Accumulator);
01180         break;
01181     }
01182     //STA Store Accumulator In Memory, Zero Page X, 2 Bytes, 4 Cycles
01183     case 0x95:
01184     {
01185         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.ZeroPageX),
01186             (byte)Accumulator);
01187         break;
01188     }
01189     //STA Store Accumulator In Memory, Absolute, 3 Bytes, 4 Cycles
01190     case 0x8D:
01191     {
01192         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.Absolute),
01193             (byte)Accumulator);
01194         break;
01195     }
01196     //STA Store Accumulator In Memory, Absolute X, 3 Bytes, 5 Cycles
01197     case 0x9D:
01198     {
01199         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.AbsoluteX),
01200             (byte)Accumulator);
01201         IncrementCycleCount();
01202         break;
01203     }
01204     //STA Store Accumulator In Memory, Absolute Y, 3 Bytes, 5 Cycles
01205     case 0x99:
01206     {
01207         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.AbsoluteY),
01208             (byte)Accumulator);
01209         IncrementCycleCount();
01210         break;
01211     }
01212     //STA Store Accumulator In Memory, Indexed Indirect, 2 Bytes, 6 Cycles
01213     case 0x81:
01214     {
01215         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.IndirectX),
01216             (byte)Accumulator);
01217         break;
01218     }
01219     //STA Store Accumulator In Memory, Indirect Indexed, 2 Bytes, 6 Cycles
01220     case 0x91:
01221     {
01222         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.IndirectY),
01223             (byte)Accumulator);
01224         IncrementCycleCount();
01225         break;
01226     }
01227     //STX Store Index X, Zero Page, 2 Bytes, 3 Cycles
01228     case 0x86:
01229     {
01230         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.ZeroPage),
01231             (byte)XRegister);
01232         break;
01233     }
01234     //STX Store Index X, Zero Page Y, 2 Bytes, 4 Cycles
01235     case 0x96:
01236     {
01237         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.ZeroPageY),
01238             (byte)XRegister);
01239         break;
01240     }
01241     //STX Store Index X, Absolute, 3 Bytes, 4 Cycles
01242     case 0x8E:
01243     {
01244         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.Absolute),
01245             (byte)XRegister);
01246         break;
01247     }

```

```

01236         (byte)XRegister);
01237             break;
01238         }
01239         //STY Store Index Y, Zero Page, 2 Bytes, 3 Cycles
01240         case 0x84:
01241         {
01242             MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.ZeroPage),
01243             (byte)YRegister);
01244             break;
01245         }
01246         //STY Store Index Y, Zero Page X, 2 Bytes, 4 Cycles
01247         case 0x94:
01248         {
01249             MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.ZeroPageX),
01250             (byte)YRegister);
01251             break;
01252         }
01253         //STY Store Index Y, Absolute, 2 Bytes, 4 Cycles
01254         case 0x8C:
01255         {
01256             MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.Absolute),
01257             (byte)YRegister);
01258             break;
01259         }
01260     }
01261 #endregion
01262 #region Transfer Operations
01263 //TAX Transfer Accumulator to X Register, Implied, 1 Bytes, 2 Cycles
01264 case 0xAA:
01265 {
01266     IncrementCycleCount();
01267     XRegister = Accumulator;
01268     SetNegativeFlag(XRegister);
01269     SetZeroFlag(XRegister);
01270     break;
01271 }
01272 //TAY Transfer Accumulator to Y Register, 1 Bytes, 2 Cycles
01273 case 0xA8:
01274 {
01275     IncrementCycleCount();
01276     YRegister = Accumulator;
01277     SetNegativeFlag(YRegister);
01278     SetZeroFlag(YRegister);
01279     break;
01280 }
01281 //TXA Transfer X Register to Accumulator, Implied, 1 Bytes, 2 Cycles
01282 case 0x8A:
01283 {
01284     IncrementCycleCount();
01285     Accumulator = XRegister;
01286     SetNegativeFlag(Accumulator);
01287     SetZeroFlag(Accumulator);
01288     break;
01289 }
01290 //TYA Transfer Y Register to Accumulator, Implied, 1 Bytes, 2 Cycles
01291 case 0x98:
01292 {
01293     IncrementCycleCount();
01294     Accumulator = YRegister;
01295     SetNegativeFlag(Accumulator);
01296     SetZeroFlag(Accumulator);
01297     break;
01298 }
01299 #endregion
01300 //NOP Operation, Implied, 1 Byte, 2 Cycles
01301 case 0xEA:
01302 {
01303     IncrementCycleCount();
01304     break;
01305 }
01306 default:
01307     throw new NotSupportedException(string.Format("The OpCode {0} is not supported",
01308     CurrentOpCode));
01309 }
01310 }
01311 }

```

6.41.3.13 GetAddressByAddressingMode() int Hardware.W65C02.GetAddressByAddressingMode (

```
AddressingMode addressingMode ) [inline], [protected]
```

Uses the AddressingMode to return the correct address based on the mode. Note: This method will not increment the program counter for any mode. Note: This method will return an error if called for either the immediate or accumulator modes.

Parameters

<i>addressingMode</i>	The addressing Mode to use
-----------------------	----------------------------

Returns

The memory Location

Definition at line 1339 of file W65C02.cs.

```
01340     {
01341         int address;
01342         int highByte;
01343         switch (addressingMode)
01344         {
01345             case (AddressingMode.Absolute):
01346             {
01347                 return (MemoryMap.Read(ProgramCounter++) | (MemoryMap.Read(ProgramCounter++) <<
01348 8));
01349             }
01350             case AddressingMode.AbsoluteX:
01351             {
01352                 //Get the low half of the address
01353                 address = MemoryMap.Read(ProgramCounter++);
01354                 //Get the high byte
01355                 highByte = MemoryMap.Read(ProgramCounter++);
01356                 //We crossed a page boundary, so an extra read has occurred.
01357                 //However, if this is an ASL, LSR, DEC, INC, ROR, ROL or STA operation, we do
01358                 not decrease it by 1.
01359                 if (address + XRegister > 0xFF)
01360                 {
01361                     switch (CurrentOpCode)
01362                     {
01363                         case 0x1E:
01364                         case 0xDE:
01365                         case 0xFE:
01366                         case 0x5E:
01367                         case 0x3E:
01368                         case 0x7E:
01369                         case 0x9D:
01370                         {
01371                             //This is a MemoryMap.Read Fetch Write Operation, so we don't
01372                             make the extra read.
01373                             return ((highByte << 8 | address) + XRegister) & 0xFFFF;
01374                         }
01375                         default:
01376                         {
01377                             MemoryMap.Read((((highByte << 8 | address) + XRegister) - 0xFF)
01378                             & 0xFFFF);
01379                             break;
01380                         }
01381                     }
01382                     return ((highByte << 8 | address) + XRegister) & 0xFFFF;
01383                 }
01384             case AddressingMode.AbsoluteY:
01385             {
01386                 //Get the low half of the address
01387                 address = MemoryMap.Read(ProgramCounter++);
01388                 //Get the high byte
01389                 highByte = MemoryMap.Read(ProgramCounter++);
01390                 //We crossed a page boundary, so decrease the number of cycles by 1 if the
01391                 operation is not STA
01392                 if (address + YRegister > 0xFF && CurrentOpCode != 0x99)
01393                 {
01394                     MemoryMap.Read((((highByte << 8 | address) + YRegister) - 0xFF) & 0xFFFF);
01395                 }
01396             }
```



```

01397
01398         //Bitshift the high byte into place, AND with FFFF to handle wrapping.
01399         return ((highByte << 8 | address) + YRegister) & 0xFFFF;
01400     }
01401     case AddressingMode.Immediate:
01402     {
01403         return ProgramCounter++;
01404     }
01405     case AddressingMode.IndirectX:
01406     {
01407         //Get the location of the address to retrieve
01408         address = MemoryMap.Read(ProgramCounter++);
01409         MemoryMap.Read(address);
01410
01411         address += XRegister;
01412
01413         //Now get the final Address. The is not a zero page address either.
01414         var finalAddress = MemoryMap.Read((address & 0xFF) | (MemoryMap.Read((address
+ 1) & 0xFF) << 8));
01415         return finalAddress;
01416     }
01417     case AddressingMode.IndirectY:
01418     {
01419         address = MemoryMap.Read(ProgramCounter++);
01420
01421         var finalAddress = MemoryMap.Read(address) + (MemoryMap.Read((address + 1) &
0xFF) << 8);
01422
01423         if ((finalAddress & 0xFF) + YRegister > 0xFF && CurrentOpCode != 0x91)
01424         {
01425             MemoryMap.Read((finalAddress + YRegister - 0xFF) & 0xFFFF);
01426         }
01427
01428         return (finalAddress + YRegister) & 0xFFFF;
01429     }
01430     case AddressingMode.Relative:
01431     {
01432         return ProgramCounter;
01433     }
01434     case (AddressingMode.ZeroPage):
01435     {
01436         address = MemoryMap.Read(ProgramCounter++);
01437         return address;
01438     }
01439     case (AddressingMode.ZeroPageX):
01440     {
01441         address = MemoryMap.Read(ProgramCounter++);
01442         MemoryMap.Read(address);
01443
01444         address += XRegister;
01445         address &= 0xFF;
01446
01447         //This address wraps if its greater than 0xFF
01448         if (address > 0xFF)
01449         {
01450             address -= 0x100;
01451             return address;
01452         }
01453
01454         return address;
01455     }
01456     case (AddressingMode.ZeroPageY):
01457     {
01458         address = MemoryMap.Read(ProgramCounter++);
01459         MemoryMap.Read(address);
01460
01461         address += YRegister;
01462         address &= 0xFF;
01463
01464         return address;
01465     }
01466     default:
01467         throw new InvalidOperationException(string.Format("The Address Mode '{0}' does not
require an address", addressingMode));
01468     }
01469 }

```

6.41.3.14 GetAddressingMode() `AddressingMode Hardware.W65C02.GetAddressingMode () [inline], [private]`

Definition at line 1680 of file [W65C02.cs](#).

```

01681     {
01682         switch (CurrentOpCode)
01683         {
01684             case 0x0D: //ORA
01685             case 0x2D: //AND
01686             case 0x4D: //EOR
01687             case 0x6D: //ADC
01688             case 0x8D: //STA
01689             case 0xAD: //LDA
01690             case 0xCD: //CMP
01691             case 0xED: //SBC
01692             case 0x0E: //ASL
01693             case 0x2E: //ROL
01694             case 0x4E: //LSR
01695             case 0x6E: //ROR
01696             case 0x8E: //SDX
01697             case 0xAE: //LDX
01698             case 0xCE: //DEC
01699             case 0xEE: //INC
01700             case 0x2C: //Bit
01701             case 0x4C: //JMP
01702             case 0x8C: //STY
01703             case 0xAC: //LDY
01704             case 0xCC: //CPY
01705             case 0xEC: //CPX
01706             case 0x20: //JSR
01707             {
01708                 return AddressingMode.Absolute;
01709             }
01710             case 0x1D: //ORA
01711             case 0x3D: //AND
01712             case 0x5D: //EOR
01713             case 0x7D: //ADC
01714             case 0x9D: //STA
01715             case 0xBD: //LDA
01716             case 0xDD: //CMP
01717             case 0xFD: //SBC
01718             case 0xBC: //LDY
01719             case 0xFE: //INC
01720             case 0x1E: //ASL
01721             case 0x3E: //ROL
01722             case 0x5E: //LSR
01723             case 0x7E: //ROR
01724             {
01725                 return AddressingMode.AbsoluteX;
01726             }
01727             case 0x19: //ORA
01728             case 0x39: //AND
01729             case 0x59: //EOR
01730             case 0x79: //ADC
01731             case 0x99: //STA
01732             case 0xB9: //LDA
01733             case 0xD9: //CMP
01734             case 0xF9: //SBC
01735             case 0xBE: //LDX
01736             {
01737                 return AddressingMode.AbsoluteY;
01738             }
01739             case 0x0A: //ASL
01740             case 0x4A: //LSR
01741             case 0x2A: //ROL
01742             case 0x6A: //ROR
01743             {
01744                 return AddressingMode.Accumulator;
01745             }
01746             case 0x09: //ORA
01747             case 0x29: //AND
01748             case 0x49: //EOR
01749             case 0x69: //ADC
01750             case 0xA0: //LDY
01751             case 0xC0: //CPY
01752             case 0xE0: //CMP
01753             case 0xA2: //LDX
01754             case 0xA9: //LDA
01755             case 0xC9: //CMP
01756             case 0xE9: //SBC
01757             {
01758                 return AddressingMode.Immediate;
01759             }
01760             case 0x00: //BRK
01761             case 0x18: //CLC
01762             case 0xD8: //CLD
01763             case 0x58: //CLI
01764             case 0xB8: //CLV
01765             case 0xDE: //DEC
01766             case 0xCA: //DEX

```

```

01768         case 0x88: //DEY
01769         case 0xE8: //INX
01770         case 0xC8: //INY
01771         case 0xEA: //NOP
01772         case 0x48: //PHA
01773         case 0x08: //PHP
01774         case 0x68: //PLA
01775         case 0x28: //PLP
01776         case 0x40: //RTI
01777         case 0x60: //RTS
01778         case 0x38: //SEC
01779         case 0xF8: //SED
01780         case 0x78: //SEI
01781         case 0xAA: //TAX
01782         case 0xA8: //TAY
01783         case 0xBA: //TSX
01784         case 0x8A: //TXA
01785         case 0x9A: //TXS
01786         case 0x98: //TYA
01787         {
01788             return AddressingMode.Implied;
01789         }
01790         case 0x6C:
01791         {
01792             return AddressingMode.Indirect;
01793         }
01794
01795         case 0x61: //ADC
01796         case 0x21: //AND
01797         case 0xC1: //CMP
01798         case 0x41: //EOR
01799         case 0xA1: //LDA
01800         case 0x01: //ORA
01801         case 0xE1: //SBC
01802         case 0x81: //STA
01803         {
01804             return AddressingMode.IndirectX;
01805         }
01806
01807         case 0x71: //ADC
01808         case 0x31: //AND
01809         case 0xD1: //CMP
01810         case 0x51: //EOR
01811         case 0xB1: //LDA
01812         case 0x11: //ORA
01813         case 0xF1: //SBC
01814         case 0x91: //STA
01815         {
01816             return AddressingMode.IndirectY;
01817         }
01818
01819         case 0x90: //BCC
01820         case 0xB0: //BCS
01821         case 0xF0: //BEQ
01822         case 0x30: //BMI
01823         case 0xD0: //BNE
01824         case 0x10: //BPL
01825         case 0x50: //BVC
01826         case 0x70: //BVS
01827         {
01828             return AddressingMode.Relative;
01829         }
01830
01831         case 0x65: //ADC
01832         case 0x25: //AND
01833         case 0x06: //ASL
01834         case 0x24: //BIT
01835         case 0xC5: //CMP
01836         case 0xE4: //CPX
01837         case 0xC4: //CPY
01838         case 0xC6: //DEC
01839         case 0x45: //EOR
01840         case 0xE6: //INC
01841         case 0xA5: //LDA
01842         case 0xA6: //LDX
01843         case 0xA4: //LDY
01844         case 0x46: //LSR
01845         case 0x05: //ORA
01846         case 0x26: //ROL
01847         case 0x66: //ROR
01848         case 0xE5: //SBC
01849         case 0x85: //STA
01850         case 0x86: //STX
01851         case 0x84: //STY
01852         {
01853             return AddressingMode.ZeroPage;
01854         }
01855
01856         case 0x75: //ADC
01857         case 0x35: //AND
01858         case 0x16: //ASL

```

```

01855         case 0xD5: //CMP
01856         case 0xD6: //DEC
01857         case 0x55: //EOR
01858         case 0xF6: //INC
01859         case 0xB5: //LDA
01860         case 0xB6: //LDX
01861         case 0xB4: //LDY
01862         case 0x56: //LSR
01863         case 0x15: //ORA
01864         case 0x36: //ROL
01865         case 0x76: //ROR
01866         case 0xF5: //SBC
01867         case 0x95: //STA
01868         case 0x96: //STX
01869         case 0x94: //STY
01870     {
01871         return AddressingMode.ZeroPageX;
01872     }
01873     default:
01874         throw new NotSupportedException(string.Format("Opcode {0} is not supported",
CurrentOpCode));
01875     }
01876 }

```

6.41.3.15 GetCycleCount() int Hardware.W65C02.GetCycleCount () [inline]

Gets the Number of Cycles that have elapsed

Returns

The number of elapsed cycles

Definition at line 208 of file [W65C02.cs](#).

```

00209     {
00210         return _cycleCount;
00211     }

```

6.41.3.16 IncrementCycleCount() void Hardware.W65C02.IncrementCycleCount () [inline]

Increments the Cycle Count, causes a CycleCountIncrementedAction to fire.

Definition at line 216 of file [W65C02.cs](#).

```

00217     {
00218         _cycleCount++;
00219         CycleCountIncrementedAction();
00220
00221         _previousInterrupt = _interrupt;
00222         _interrupt = TriggerNmi || (TriggerIRQ && !DisableInterruptFlag);
00223     }

```

6.41.3.17 InterruptRequest() void Hardware.W65C02.InterruptRequest () [inline]

The InterruptRequest or IRQ

Definition at line 199 of file [W65C02.cs](#).

```

00200     {
00201         TriggerIRQ = true;
00202     }

```

6.41.3.18 JumpToSubRoutineOperation() void Hardware.W65C02.JumpToSubRoutineOperation ()
[inline], [private]

The JSR routine. Jumps to a subroutine.

Definition at line 2308 of file W65C02.cs.

```
02309     {
02310         IncrementCycleCount();
02311
02312         //Put the high value on the stack, this should be the address after our operation -1
02313         //The RTS operation increments the PC by 1 which is why we don't move 2
02314         PokeStack((byte)((ProgramCounter + 1) >> 8) & 0xFF);
02315         StackPointer--;
02316         IncrementCycleCount();
02317
02318         PokeStack((byte)((ProgramCounter + 1) & 0xFF));
02319         StackPointer--;
02320         IncrementCycleCount();
02321
02322         ProgramCounter = GetAddressByAddressingMode(AddressingMode.Absolute);
02323     }
```

6.41.3.19 LsrOperation() void Hardware.W65C02.LsrOperation (
AddressingMode addressingMode) [inline], [private]

The LSR Operation. Performs a Left shift operation on a value in memory

Parameters

<i>addressingMode</i>	The addressing mode to use
-----------------------	----------------------------

Definition at line 2103 of file W65C02.cs.

```
02104     {
02105         int value;
02106         var memoryAddress = 0;
02107         if (addressingMode == AddressingMode.Accumulator)
02108         {
02109             MemoryMap.Read(ProgramCounter + 1);
02110             value = Accumulator;
02111         }
02112         else
02113         {
02114             memoryAddress = GetAddressByAddressingMode(addressingMode);
02115             value = MemoryMap.Read(memoryAddress);
02116         }
02117
02118         //Dummy Write
02119         if (addressingMode != AddressingMode.Accumulator)
02120         {
02121             MemoryMap.Write(memoryAddress, (byte)value);
02122         }
02123
02124         NegativeFlag = false;
02125
02126         //If the Zero bit is set, we have a carry
02127         CarryFlag = (value & 0x01) != 0;
02128
02129         value = (value >> 1);
02130
02131         SetZeroFlag(value);
02132         if (addressingMode == AddressingMode.Accumulator)
02133             Accumulator = value;
02134         else
02135         {
02136             MemoryMap.Write(memoryAddress, (byte)value);
02137         }
02138     }
```

6.41.3.20 MoveProgramCounterByRelativeValue() void Hardware.W65C02.MoveProgramCounterByRelativeValue (byte valueToMove) [inline], [private]

Moves the ProgramCounter in a given direction based on the value inputted

Definition at line 1475 of file W65C02.cs.

```
01476     {
01477         var movement = valueToMove > 127 ? (valueToMove - 255) : valueToMove;
01478
01479         var newProgramCounter = ProgramCounter + movement;
01480
01481         //This makes sure that we always land on the correct spot for a positive number
01482         if (movement >= 0)
01483             newProgramCounter++;
01484
01485         //We Crossed a Page Boundary. So we increment the cycle counter by one. The +1 is
01486         //because we always check from the end of the instruction not the beginning
01487         if (((ProgramCounter + 1 ^ newProgramCounter) & 0xff00) != 0x0000)
01488         {
01489             IncrementCycleCount();
01490         }
01491
01492         ProgramCounter = newProgramCounter;
01493         MemoryMap.Read(ProgramCounter);
01494     }
```

6.41.3.21 NextStep() void Hardware.W65C02.NextStep () [inline]

Performs the next step on the processor

Definition at line 170 of file W65C02.cs.

```
00171     {
00172         SetDisassembly();
00173
00174         //Have to read this first otherwise it causes tests to fail on a NES
00175         CurrentOpCode = MemoryMap.Read(ProgramCounter);
00176
00177         ProgramCounter++;
00178
00179         ExecuteOpCode();
00180
00181         if (_previousInterrupt)
00182         {
00183             if (TriggerNmi)
00184             {
00185                 ProcessNMI();
00186                 TriggerNmi = false;
00187             }
00188             else if (TriggerIRQ)
00189             {
00190                 ProcessIRQ();
00191                 TriggerIRQ = false;
00192             }
00193         }
00194     }
```

6.41.3.22 OrOperation() void Hardware.W65C02.OrOperation (AddressingMode addressingMode) [inline], [private]

The Or Operation. Performs an Or Operation with the accumulator and a value in memory

Parameters

<i>addressingMode</i>	The addressing mode to use
-----------------------	----------------------------

Definition at line 2144 of file W65C02.cs.

```
02145     {
02146         Accumulator = Accumulator | MemoryMap.Read(GetAddressByAddressingMode(addressingMode));
02147     }
02148     SetNegativeFlag(Accumulator);
02149     SetZeroFlag(Accumulator);
02150 }
```

6.41.3.23 PeekStack() byte Hardware.W65C02.PeekStack () [inline], [private]

Returns a the value from the stack without changing the position of the stack pointer

Returns

The value at the current Stack Pointer

Definition at line 1499 of file W65C02.cs.

```
01500     {
01501         //The stack lives at 0x100-0x1FF, but the value is only a byte so it needs to be
01502         translated return MemoryMap.Read(StackPointer + 0x100);
01503     }
```

6.41.3.24 PokeStack() void Hardware.W65C02.PokeStack (byte value) [inline], [private]

Write a value directly to the stack without modifying the Stack Pointer

Parameters

<i>value</i>	The value to be written to the stack
--------------	--------------------------------------

Definition at line 1510 of file W65C02.cs.

```
01511     {
01512         //The stack lives at 0x100-0x1FF, but the value is only a byte so it needs to be
01513         translated MemoryMap.Write(StackPointer + 0x100, value);
01514     }
```

6.41.3.25 ProcessIRQ() void Hardware.W65C02.ProcessIRQ () [inline], [private]

This is ran anytime an IRQ occurs

Definition at line 2420 of file W65C02.cs.

```
02421     {
02422         if (DisableInterruptFlag)
02423             return;
02424     }
02425     ProgramCounter--;
02426     BreakOperation(false, 0xFFFE);
02427     CurrentOpCode = MemoryMap.Read(ProgramCounter);
02428     SetDisassembly();
02429 }
02430 }
```

6.41.3.26 ProcessNMI() void Hardware.W65C02.ProcessNMI () [inline], [private]

This is ran anytime an NMI occurs

Definition at line 2408 of file W65C02.cs.

```
02409     {
02410         ProgramCounter--;
02411         BreakOperation(false, 0xFFFA);
02412         CurrentOpCode = MemoryMap.Read(ProgramCounter);
02413
02414         SetDisassembly();
02415     }
```

6.41.3.27 PullFlagsOperation() void Hardware.W65C02.PullFlagsOperation () [inline], [private]

The PLP Operation. Pull the status flags off the stack on sets the flags accordingly.

Definition at line 2292 of file W65C02.cs.

```
02293     {
02294         var flags = PeekStack();
02295         CarryFlag = (flags & 0x01) != 0;
02296         ZeroFlag = (flags & 0x02) != 0;
02297         DisableInterruptFlag = (flags & 0x04) != 0;
02298         DecimalFlag = (flags & 0x08) != 0;
02299         OverflowFlag = (flags & 0x40) != 0;
02300         NegativeFlag = (flags & 0x80) != 0;
02301
02302
02303     }
```

6.41.3.28 PushFlagsOperation() void Hardware.W65C02.PushFlagsOperation () [inline], [private]

The PSP Operation. Pushes the Status Flags to the stack

Definition at line 2284 of file W65C02.cs.

```
02285     {
02286         PokeStack(ConvertFlagsToByte(true));
02287     }
```

6.41.3.29 Reset() void Hardware.W65C02.Reset () [inline]

Initializes the processor to its default state.

Definition at line 151 of file W65C02.cs.

```
00152     {
00153         ResetCycleCount();
00154         StackPointer = 0x1FD;
00155         //Set the Program Counter to the Reset Vector Address.
00156         ProgramCounter = 0xFFFC;
00157         //Reset the Program Counter to the Address contained in the Reset Vector
00158         ProgramCounter = (MemoryMap.Read(ProgramCounter) | (MemoryMap.Read(ProgramCounter + 1) <<
00159     8));
00159         CurrentOpCode = MemoryMap.Read(ProgramCounter);
00160         //SetDisassembly();
00161         DisableInterruptFlag = true;
00162         _previousInterrupt = false;
00163         TriggerNmi = false;
00164         TriggerIRQ = false;
00165     }
```


6.41.3.30 ResetCycleCount() void Hardware.W65C02.ResetCycleCount () [inline]

Resets the Cycle Count back to 0

Definition at line 228 of file W65C02.cs.

```
00229     {
00230         _cycleCount = 0;
00231     }
```

6.41.3.31 ReturnFromInterruptOperation() void Hardware.W65C02.ReturnFromInterruptOperation () [inline], [private]

The RTI routine. Called when returning from a BRK operation. Note: when called after a BRK operation the Program Counter is not set to the location after the BRK, it is set +1

Definition at line 2385 of file W65C02.cs.

```
02386     {
02387         MemoryMap.Read(++ProgramCounter);
02388         StackPointer++;
02389         IncrementCycleCount();
02390
02391         PullFlagsOperation();
02392         StackPointer++;
02393         IncrementCycleCount();
02394
02395         var lowBit = PeekStack();
02396         StackPointer++;
02397         IncrementCycleCount();
02398
02399         var highBit = PeekStack() << 8;
02400         IncrementCycleCount();
02401
02402         ProgramCounter = (highBit | lowBit);
02403     }
```

6.41.3.32 ReturnFromSubRoutineOperation() void Hardware.W65C02.ReturnFromSubRoutineOperation () [inline], [private]

The RTS routine. Called when returning from a subroutine.

Definition at line 2328 of file W65C02.cs.

```
02329     {
02330         MemoryMap.Read(++ProgramCounter);
02331         StackPointer++;
02332         IncrementCycleCount();
02333
02334         var lowBit = PeekStack();
02335         StackPointer++;
02336         IncrementCycleCount();
02337
02338         var highBit = PeekStack() << 8;
02339         IncrementCycleCount();
02340
02341         ProgramCounter = (highBit | lowBit) + 1;
02342         IncrementCycleCount();
02343     }
```

6.41.3.33 RolOperation() void Hardware.W65C02.RolOperation (AddressingMode addressingMode) [inline], [private]

The ROL operation. Performs a rotate left operation on a value in memory.

Parameters

<i>addressingMode</i>	The addressing mode to use
-----------------------	----------------------------

Definition at line 2156 of file W65C02.cs.

```

02157     {
02158         int value;
02159         var memoryAddress = 0;
02160         if (addressingMode == AddressingMode.Accumulator)
02161         {
02162             //Dummy MemoryMap.Read
02163             MemoryMap.Read(ProgramCounter + 1);
02164             value = Accumulator;
02165         }
02166         else
02167         {
02168             memoryAddress = GetAddressByAddressingMode(addressingMode);
02169             value = MemoryMap.Read(memoryAddress);
02170         }
02171
02172         //Dummy Write
02173         if (addressingMode != AddressingMode.Accumulator)
02174         {
02175             MemoryMap.Write(memoryAddress, (byte)value);
02176         }
02177
02178         //Store the carry flag before shifting it
02179         var newCarry = (0x80 & value) != 0;
02180
02181         //The And here ensures that if the value is greater than 255 it wraps properly.
02182         value = (value << 1) & 0xFE;
02183
02184         if (CarryFlag)
02185             value = value | 0x01;
02186
02187         CarryFlag = newCarry;
02188
02189         SetZeroFlag(value);
02190         SetNegativeFlag(value);
02191
02192
02193         if (addressingMode == AddressingMode.Accumulator)
02194             Accumulator = value;
02195         else
02196         {
02197             MemoryMap.Write(memoryAddress, (byte)value);
02198         }
02199     }

```

6.41.3.34 RorOperation() void Hardware.W65C02.RorOperation (
AddressingMode *addressingMode*) [inline], [private]

The ROR operation. Performs a rotate right operation on a value in memory.

Parameters

<i>addressingMode</i>	The addressing mode to use
-----------------------	----------------------------

Definition at line 2205 of file W65C02.cs.

```

02206     {
02207         int value;
02208         var memoryAddress = 0;
02209         if (addressingMode == AddressingMode.Accumulator)
02210         {
02211             //Dummy MemoryMap.Read
02212             MemoryMap.Read(ProgramCounter + 1);
02213             value = Accumulator;
02214         }
02215         else
02216         {
02217             memoryAddress = GetAddressByAddressingMode(addressingMode);

```

```

02218         value = MemoryMap.Read(memoryAddress);
02219     }
02220
02221     //Dummy Write
02222     if (addressingMode != AddressingMode.Accumulator)
02223     {
02224         MemoryMap.Write(memoryAddress, (byte)value);
02225     }
02226
02227     //Store the carry flag before shifting it
02228     var newCarry = (0x01 & value) != 0;
02229
02230     value = (value » 1);
02231
02232     //If the carry flag is set then 0x
02233     if (CarryFlag)
02234         value = value | 0x80;
02235
02236     CarryFlag = newCarry;
02237
02238     SetZeroFlag(value);
02239     SetNegativeFlag(value);
02240
02241     if (addressingMode == AddressingMode.Accumulator)
02242         Accumulator = value;
02243     else
02244     {
02245         MemoryMap.Write(memoryAddress, (byte)value);
02246     }
02247 }

```

6.41.3.35 SetDisassembly() void Hardware.W65C02.SetDisassembly () [inline], [private]

Definition at line 1527 of file W65C02.cs.

```

01528     {
01529         var addressMode = GetAddressingMode();
01530
01531         var currentProgramCounter = ProgramCounter;
01532
01533         currentProgramCounter = WrapProgramCounter(++currentProgramCounter);
01534         int? address1 = MemoryMap.Read(currentProgramCounter);
01535
01536         currentProgramCounter = WrapProgramCounter(++currentProgramCounter);
01537         int? address2 = MemoryMap.Read(currentProgramCounter);
01538
01539         string disassembledStep = string.Empty;
01540
01541         switch (addressMode)
01542         {
01543             case AddressingMode.Absolute:
01544             {
01545                 disassembledStep = string.Format("${0}{1}",
01546 address2.Value.ToString("X").PadLeft(2, '0'), address1.Value.ToString("X").PadLeft(2, '0'));
01547                 break;
01548             }
01549             case AddressingMode.AbsoluteX:
01550             {
01551                 disassembledStep = string.Format("${0}{1},X",
01552 address2.Value.ToString("X").PadLeft(2, '0'), address1.Value.ToString("X").PadLeft(2, '0'));
01553                 break;
01554             }
01555             case AddressingMode.AbsoluteY:
01556             {
01557                 disassembledStep = string.Format("${0}{1},Y",
01558 address2.Value.ToString("X").PadLeft(2, '0'), address1.Value.ToString("X").PadLeft(2, '0'));
01559                 break;
01560             }
01561             case AddressingMode.Accumulator:
01562             {
01563                 address1 = null;
01564                 address2 = null;
01565
01566                 disassembledStep = "A";
01567                 break;
01568             }
01569             case AddressingMode.Immediate:
01570             {
01571                 disassembledStep = string.Format("#${0}",
01572 address1.Value.ToString("X").PadLeft(4, '0'));
01573                 address2 = null;
01574             }
01575         }
01576     }
01577 }

```

```

01570             break;
01571         }
01572         case AddressingMode.Implied:
01573         {
01574             address1 = null;
01575             address2 = null;
01576             break;
01577         }
01578         case AddressingMode.Indirect:
01579         {
01580             disassembledStep = string.Format("{0}{1}",
address2.Value.ToString("X").PadLeft(2, '0'), address1.Value.ToString("X").PadLeft(2, '0'));
01581             break;
01582         }
01583         case AddressingMode.IndirectX:
01584         {
01585             address2 = null;
01586             disassembledStep = string.Format("{0},X",
address1.Value.ToString("X").PadLeft(2, '0'));
01587             break;
01588         }
01589         case AddressingMode.IndirectY:
01590         {
01591             address2 = null;
01592             disassembledStep = string.Format("{0},Y",
address1.Value.ToString("X").PadLeft(2, '0'));
01593             break;
01594         }
01595         case AddressingMode.Relative:
01596         {
01597             var valueToMove = (byte)address1.Value;
01598             var movement = valueToMove > 127 ? (valueToMove - 255) : valueToMove;
01599             var newProgramCounter = ProgramCounter + movement;
01600             //This makes sure that we always land on the correct spot for a positive
01601             number
01602             if (movement >= 0)
01603                 newProgramCounter++;
01604             var stringAddress = ProgramCounter.ToString("X").PadLeft(4, '0');
01605             address1 = int.Parse(stringAddress.Substring(0, 2),
NumberStyles.AllowHexSpecifier);
01606             address2 = int.Parse(stringAddress.Substring(2, 2),
NumberStyles.AllowHexSpecifier);
01607             disassembledStep = string.Format("{0}",
newProgramCounter.ToString("X").PadLeft(4, '0'));
01608             break;
01609         }
01610         case AddressingMode.ZeroPage:
01611         {
01612             address2 = null;
01613             disassembledStep = string.Format("{0}",
address1.Value.ToString("X").PadLeft(2, '0'));
01614             break;
01615         }
01616         case AddressingMode.ZeroPageX:
01617         {
01618             address2 = null;
01619             disassembledStep = string.Format("{0},X",
address1.Value.ToString("X").PadLeft(2, '0'));
01620             break;
01621         }
01622         case AddressingMode.ZeroPageY:
01623         {
01624             address2 = null;
01625             disassembledStep = string.Format("{0},Y",
address1.Value.ToString("X").PadLeft(4, '0'));
01626             break;
01627         }
01628         default:
01629             throw new InvalidEnumArgumentException("Invalid Addressing Mode");
01630     }
01631     CurrentDisassembly = new Disassembly
01632     {

```

```

01647         HighAddress = address2.HasValue ? address2.Value.ToString("X").PadLeft(2, '0') :
string.Empty,
01648         LowAddress = address1.HasValue ? address1.Value.ToString("X").PadLeft(2, '0') :
string.Empty,
01649         OpCodeString = CurrentOpCode.ConvertOpCodeIntoString(),
01650         DisassemblyOutput = disassembledStep
01651     };
01652
01653     _logger.Debug("{0} : {1}{2}{3} {4} {5} A: {6} X: {7} Y: {8} SP {9} N: {10} V: {11} B:
{12} D: {13} I: {14} Z: {15} C: {16}",
01654         ProgramCounter.ToString("X").PadLeft(4, '0'),
01655         CurrentOpCode.ToString("X").PadLeft(2, '0'),
01656         CurrentDisassembly.LowAddress,
01657         CurrentDisassembly.HighAddress,
01658
01659         CurrentDisassembly.OpCodeString,
01660         CurrentDisassembly.DisassemblyOutput.PadRight(10, ' '),
01661
01662         Accumulator.ToString("X").PadLeft(3, '0'),
01663         XRegister.ToString("X").PadLeft(3, '0'),
01664         YRegister.ToString("X").PadLeft(3, '0'),
01665         StackPointer.ToString("X").PadLeft(3, '0'),
01666         Convert.ToInt16(NegativeFlag),
01667         Convert.ToInt16(OverflowFlag),
01668         0,
01669         Convert.ToInt16(DecimalFlag),
01670         Convert.ToInt16(DisableInterruptFlag),
01671         Convert.ToInt16(ZeroFlag),
01672         Convert.ToInt16(CarryFlag));
01673     }

```

6.41.3.36 SetNegativeFlag() void Hardware.W65C02.SetNegativeFlag (
int value) [inline], [protected]

Sets the IsSignNegative register

Parameters

<i>value</i>	
--------------	--

Definition at line 1317 of file W65C02.cs.

```

01318     {
01319         //on the 6502, any value greater than 127 is negative. 128 = 1000000 in Binary. the 8th
bit is set, therefore the number is a negative number.
01320         NegativeFlag = value > 127;
01321     }

```

6.41.3.37 SetZeroFlag() void Hardware.W65C02.SetZeroFlag (
int value) [inline], [protected]

Sets the IsResultZero register

Parameters

<i>value</i>	
--------------	--

Definition at line 1327 of file W65C02.cs.

```

01328     {
01329         ZeroFlag = value == 0;
01330     }

```

6.41.3.38 SubtractWithBorrowOperation() void Hardware.W65C02.SubtractWithBorrowOperation (AddressingMode addressingMode) [inline], [protected]

The SBC operation. Performs a subtract with carry operation on the accumulator and a value in memory.

Parameters

<i>addressingMode</i>	The addressing mode to use
-----------------------	----------------------------

Definition at line 2253 of file W65C02.cs.

```

02254     {
02255         var memoryValue = MemoryMap.Read(GetAddressByAddressingMode(addressingMode));
02256         var newValue = DecimalFlag ? int.Parse(Accumulator.ToString("x")) -
int.Parse(memoryValue.ToString("x")) - (CarryFlag ? 0 : 1) : Accumulator - memoryValue - (CarryFlag
? 0 : 1);
02257
02258         CarryFlag = newValue >= 0;
02259
02260         if (DecimalFlag)
02261         {
02262             if (newValue < 0)
02263                 newValue += 100;
02264
02265             newValue = (int)Convert.ToInt64(string.Concat("0x", newValue), 16);
02266         }
02267         else
02268         {
02269             OverflowFlag = (((Accumulator ^ newValue) & 0x80) != 0) && (((Accumulator ^
memoryValue) & 0x80) != 0);
02270
02271             if (newValue < 0)
02272                 newValue += 256;
02273         }
02274         SetNegativeFlag(newValue);
02275         SetZeroFlag(newValue);
02276         Accumulator = newValue;
02277     }
02278
02279

```

6.41.3.39 WrapProgramCounter() int Hardware.W65C02.WrapProgramCounter (int value) [inline], [private]

Definition at line 1675 of file W65C02.cs.

```

01676     {
01677         return value & 0xFFFF;
01678     }

```

6.41.4 Member Data Documentation

6.41.4.1 _cycleCount int Hardware.W65C02._cycleCount [private]

Definition at line 18 of file W65C02.cs.

6.41.4.2 _interrupt bool Hardware.W65C02._interrupt [private]

Definition at line 20 of file W65C02.cs.

6.41.4.3 `_logger` `readonly ILogger Hardware.W65C02._logger = LogManager.GetLogger("Processor")`
[private]

Definition at line 15 of file [W65C02.cs](#).

6.41.4.4 `_previousInterrupt` `bool Hardware.W65C02._previousInterrupt` [private]

Definition at line 19 of file [W65C02.cs](#).

6.41.4.5 `_programCounter` `int Hardware.W65C02._programCounter` [private]

Definition at line 16 of file [W65C02.cs](#).

6.41.4.6 `_stackPointer` `int Hardware.W65C02._stackPointer` [private]

Definition at line 17 of file [W65C02.cs](#).

6.41.4.7 `isRunning` `bool Hardware.W65C02.isRunning`

Checks shether the emulated computer is running or not.

Definition at line 25 of file [W65C02.cs](#).

6.41.5 Property Documentation

6.41.5.1 `Accumulator` `int Hardware.W65C02.Accumulator` [get], [protected set]

The Accumulator. This value is implemented as an integer instead of a byte. This is done so we can detect wrapping of the value and set the correct number of cycles.

Definition at line 33 of file [W65C02.cs](#).

```
00033 { get; protected set; }
```

6.41.5.2 CarryFlag `bool Hardware.W65C02.CarryFlag [get], [protected set]`

This is the carry flag. when adding, if the result is greater than 255 or 99 in BCD Mode, then this bit is enabled. In subtraction this is reversed and set to false if a borrow is required IE the result is less than 0

Definition at line 93 of file [W65C02.cs](#).

```
00093 { get; protected set; }
```

6.41.5.3 CurrentDisassembly `Disassembly Hardware.W65C02.CurrentDisassembly [get], [private set]`

The disassembly of the current operation. This value is only set when the CPU is built in debug mode.

Definition at line 53 of file [W65C02.cs](#).

```
00053 { get; private set; }
```

6.41.5.4 CurrentOpCode `int Hardware.W65C02.CurrentOpCode [get], [private set]`

The Current Op Code being executed by the system

Definition at line 48 of file [W65C02.cs](#).

```
00048 { get; private set; }
```

6.41.5.5 CycleCountIncrementedAction `Action Hardware.W65C02.CycleCountIncrementedAction [get], [set]`

An external action that occurs when the cycle count is incremented

Definition at line 86 of file [W65C02.cs](#).

```
00086 { get; set; }
```

6.41.5.6 DecimalFlag `bool Hardware.W65C02.DecimalFlag [get], [private set]`

Binary Coded Decimal Mode is set/cleared via this flag. when this mode is in effect, a byte represents a number from 0-99.

Definition at line 111 of file [W65C02.cs](#).

```
00111 { get; private set; }
```

6.41.5.7 DisableInterruptFlag `bool Hardware.W65C02.DisableInterruptFlag [get], [private set]`

This determines if Interrupts are currently disabled. This flag is turned on during a reset to prevent an interrupt from occurring during startup/initialization. If this flag is true, then the IRQ pin is ignored.

Definition at line 105 of file [W65C02.cs](#).

```
00105 { get; private set; }
```


6.41.5.8 NegativeFlag `bool Hardware.W65C02.NegativeFlag [get], [private set]`

Set to true if the result of an operation is negative in ADC and SBC operations. Remember that 128-256 represent negative numbers when doing signed math. In shift operations the sign holds the carry.

Definition at line 127 of file [W65C02.cs](#).

```
00127 { get; private set; }
```

6.41.5.9 OverflowFlag `bool Hardware.W65C02.OverflowFlag [get], [protected set]`

This property is set when an overflow occurs. An overflow happens if the high bit(7) changes during the operation. Remember that values from 128-256 are negative values as the high bit is set to 1. Examples: $64 + 64 = -128$ $-128 + -128 = 0$

Definition at line 120 of file [W65C02.cs](#).

```
00120 { get; protected set; }
```

6.41.5.10 ProgramCounter `int Hardware.W65C02.ProgramCounter [get], [private set]`

Points to the Current Address of the instruction being executed by the system. The PC wraps when the value is greater than 65535, or less than 0.

Definition at line 59 of file [W65C02.cs](#).

```
00060 {
00061     get { return _programCounter; }
00062     private set { _programCounter = WrapProgramCounter(value); }
00063 }
```

6.41.5.11 StackPointer `int Hardware.W65C02.StackPointer [get], [private set]`

Points to the Current Position of the Stack. This value is a 00-FF value but is offset to point to the location in memory where the stack resides.

Definition at line 69 of file [W65C02.cs](#).

```
00070 {
00071     get { return _stackPointer; }
00072     private set
00073     {
00074         if (value > 0xFF)
00075             _stackPointer = value - 0x100;
00076         else if (value < 0x00)
00077             _stackPointer = value + 0x100;
00078         else
00079             _stackPointer = value;
00080     }
00081 }
```

6.41.5.12 TriggerIRQ `bool Hardware.W65C02.TriggerIRQ [get], [private set]`

Set to true when an IRQ has occurred and is being processed by the CPU.

Definition at line 135 of file [W65C02.cs](#).

```
00135 { get; private set; }
```

6.41.5.13 TriggerNmi `bool Hardware.W65C02.TriggerNmi [get], [set]`

Set to true when an NMI should occur

Definition at line 132 of file [W65C02.cs](#).

```
00132 { get; set; }
```

6.41.5.14 XRegister `int Hardware.W65C02.XRegister [get], [private set]`

The X Index Register

Definition at line 38 of file [W65C02.cs](#).

```
00038 { get; private set; }
```

6.41.5.15 YRegister `int Hardware.W65C02.YRegister [get], [private set]`

The Y Index Register

Definition at line 43 of file [W65C02.cs](#).

```
00043 { get; private set; }
```

6.41.5.16 ZeroFlag `bool Hardware.W65C02.ZeroFlag [get], [private set]`

Is true if one of the registers is set to zero.

Definition at line 98 of file [W65C02.cs](#).

```
00098 { get; private set; }
```

The documentation for this class was generated from the following file:

- [Hardware/Hardware/W65C02.cs](#)

6.42 Hardware.W65C22 Class Reference

An implementation of a [W65C22](#) VIA.

Public Member Functions

- [W65C22](#) ([W65C02](#) processor, byte offset, int length)
- void [Reset](#) ()
Reset routine called whenever the emulated computer is reset.
- void [Init](#) (double timer)
Initialization routine for the VIA.
- void [T1Init](#) (double value)
T1 counter initialization routine.
- void [T2Init](#) (double value)
T2 counter initialization routine.
- byte [Read](#) (int address)
Routine to read from local memory.
- void [Write](#) (int address, byte data)
Writes data to the specified address in local memory.

Public Attributes

- readonly bool [T1IsIRQ](#) = false
- readonly bool [T2IsIRQ](#) = true
- int [T1CL](#) = 0x04
- int [T1CH](#) = 0x05
- int [T2CL](#) = 0x08
- int [T2CH](#) = 0x09
- int [ACR](#) = 0x0B
- int [IFR](#) = 0x0D
- int [IER](#) = 0x0E
- byte [ACR_T1TC](#) = (byte)(1 << 7)
- byte [ACR_T2TC](#) = (byte)(1 << 6)
- byte [IFR_T2](#) = (byte)(1 << 5)
- byte [IFR_T1](#) = (byte)(1 << 6)
- byte [IFR_INT](#) = (byte)(1 << 7)
- byte [IER_T2](#) = (byte)(1 << 5)
- byte [IER_T1](#) = (byte)(1 << 6)
- byte [IER_EN](#) = (byte)(1 << 7)

Properties

- byte[] [Memory](#) [get, set]
The memory area.
- int [Offset](#) [get, set]
The memory offset of the device.
- int [Length](#) [get, set]
The length of the device memory.
- int [End](#) [get]
The end of memory
- bool [T1TimerControl](#) [get, set]
T1 timer control
- bool [T2TimerControl](#) [get, set]
T2 timer control.
- bool [T1IsEnabled](#) [get, set]
Enable or check whether timer 1 is enabled or not.
- bool [T2IsEnabled](#) [get, set]
Enable or check whether timer 2 is enabled or not.
- double [T1Interval](#) [get]
Set or check the timer 1 interval.
- double [T2Interval](#) [get]
Set or check the timer 2 interval.
- Timer [T1Object](#) [get, set]
Set or get the timer 1 object.
- Timer [T2Object](#) [get, set]
Set or get the timer 2 object.
- [W65C02 Processor](#) [get, set]
Local referemce to the processor object.

Private Member Functions

- void [OnT1Timeout](#) (object sender, ElapsedEventArgs e)
Called whenever System.Timers.Timer event elapses.
- void [OnT2Timeout](#) (object sender, ElapsedEventArgs e)
Called whenever System.Timers.Timer event elapses

6.42.1 Detailed Description

An implementation of a [W65C22](#) VIA.

Definition at line 10 of file [W65C22.cs](#).

6.42.2 Constructor & Destructor Documentation

6.42.2.1 W65C22() `Hardware.W65C22.W65C22 (`
`W65C02 processor,`
`byte offset,`
`int length) [inline]`

Definition at line 122 of file [W65C22.cs](#).

```
00123     {
00124         if (offset > MemoryMap.DeviceArea.Length)
00125             throw new ArgumentException(String.Format("The offset: {0} is greater than the device
area: {1}", offset, MemoryMap.DeviceArea.Length));
00126         T1Init(1000);
00127         T2Init(1000);
00128
00129         Offset = MemoryMap.DeviceArea.Offset | offset;
00130         Memory = new byte[length + 1];
00131         Length = length;
00132         Processor = processor;
00133     }
```

6.42.3 Member Function Documentation

6.42.3.1 Init() `void Hardware.W65C22.Init (`
`double timer) [inline]`

Initialization routine for the VIA.

Parameters

<i>timer</i>	Amount of time to set timers for.
--------------	-----------------------------------

Definition at line 150 of file [W65C22.cs](#).

```
00151     {
00152         T1Init(timer);
00153         T2Init(timer);
```

```
00154      }
```

6.42.3.2 OnT1Timeout() `void Hardware.W65C22.OnT1Timeout (`
 `object sender,`
 `ElapsedEventArgs e) [inline], [private]`

Called whenever System.Timers.Timer event elapses.

Parameters

<i>sender</i>	
<i>e</i>	

Definition at line 247 of file [W65C22.cs](#).

```
00248      {
00249          if (Processor.isRunning)
00250          {
00251              if (T1IsEnabled)
00252              {
00253                  Write(IFR, (byte)(IFR_T1 & IFR_INT));
00254                  if (T1IsIRQ)
00255                  {
00256                      Processor.InterruptRequest();
00257                  }
00258                  else
00259                  {
00260                      Processor.TriggerNmi = true;
00261                  }
00262              }
00263          }
00264      }
```

6.42.3.3 OnT2Timeout() `void Hardware.W65C22.OnT2Timeout (`
 `object sender,`
 `ElapsedEventArgs e) [inline], [private]`

Called whenever System.Timers.Timer event elapses

Parameters

<i>sender</i>	
<i>e</i>	

Definition at line 272 of file [W65C22.cs](#).

```
00273      {
00274          if (Processor.isRunning)
00275          {
00276              if (T2IsEnabled)
00277              {
00278                  Write(IFR, (byte)(IFR_T2 & IFR_INT));
00279                  if (T2IsIRQ)
00280                  {
00281                      Processor.InterruptRequest();
00282                  }
00283                  else
00284                  {
00285                      Processor.TriggerNmi = true;
00286                  }
00287              }
00288          }
```

```
00289     }
```

6.42.3.4 Read() `byte Hardware.W65C22.Read (int address) [inline]`

Routine to read from local memory.

Parameters

<i>address</i>	Address to read from.
----------------	-----------------------

Returns

Byte value stored in the local memory.

Definition at line 191 of file [W65C22.cs](#).

```
00192     {
00193         if ((Offset <= address) && (address <= End))
00194         {
00195             byte data = 0x00;
00196             if (T1TimerControl)
00197             {
00198                 data = (byte)(data | ACR_T1TC);
00199             }
00200             else if (T2TimerControl)
00201             {
00202                 data = (byte)(data | ACR_T2TC);
00203             }
00204             return data;
00205         }
00206         else
00207         {
00208             return Memory[address - Offset];
00209         }
00210     }
```

6.42.3.5 Reset() `void Hardware.W65C22.Reset () [inline]`

Reset routine called whenever the emulated computer is reset.

Definition at line 138 of file [W65C22.cs](#).

```
00139     {
00140         T1TimerControl = false;
00141         T1IsEnabled = false;
00142         T2TimerControl = false;
00143         T2IsEnabled = false;
00144     }
```

6.42.3.6 T1Init() `void Hardware.W65C22.T1Init (double value) [inline]`

T1 counter initialization routine.

Parameters

<i>value</i>	Timer initialization value in milliseconds.
--------------	---

Definition at line 161 of file [W65C22.cs](#).

```

00162     {
00163         T1Object = new Timer(value);
00164         T1Object.Start();
00165         T1Object.Elapsed += OnT1Timeout;
00166         T1TimerControl = true;
00167         T1IsEnabled = false;
00168     }

```

6.42.3.7 T2Init() void Hardware.W65C22.T2Init (
 double value) [inline]

T2 counter initialization routine.

Parameters

<i>value</i>	Timer initialization value in milliseconds.
--------------	---

Definition at line 175 of file [W65C22.cs](#).

```

00176     {
00177         T2Object = new Timer(value);
00178         T2Object.Start();
00179         T2Object.Elapsed += OnT2Timeout;
00180         T2TimerControl = true;
00181         T2IsEnabled = false;
00182     }

```

6.42.3.8 Write() void Hardware.W65C22.Write (
 int address,
 byte data) [inline]

Writes data to the specified address in local memory.

Parameters

<i>address</i>	The address to write data to.
<i>data</i>	The data to be written.

Definition at line 218 of file [W65C22.cs](#).

```

00219     {
00220         if ((address == Offset + ACR) && ((data | ACR_T1TC) == ACR_T1TC))
00221         {
00222             T1TimerControl = true;
00223         }
00224         else if ((address == Offset + ACR) && ((data | ACR_T2TC) == ACR_T2TC))
00225         {
00226             T2TimerControl = true;
00227         }
00228         else if ((address == Offset + IER) && ((data | IER_T1) == IER_T1) && ((data | IER_EN) ==
IER_EN))
00229         {
00230             T1Init(T1Interval);
00231         }

```

```
00232         else if ((address == Offset + IER) && ((data | IER_T2) == IER_T2) && ((data | IER_EN) ==  
00233             IER_EN))  
00234         {  
00235             T2Init(T2Interval);  
00236             Memory[address - Offset] = data;  
00237         }
```

6.42.4 Member Data Documentation

6.42.4.1 ACR `int Hardware.W65C22.ACR = 0x0B`

Definition at line 19 of file [W65C22.cs](#).

6.42.4.2 ACR_T1TC `byte Hardware.W65C22.ACR_T1TC = (byte) (1 << 7)`

Definition at line 23 of file [W65C22.cs](#).

6.42.4.3 ACR_T2TC `byte Hardware.W65C22.ACR_T2TC = (byte) (1 << 6)`

Definition at line 24 of file [W65C22.cs](#).

6.42.4.4 IER `int Hardware.W65C22.IER = 0x0E`

Definition at line 21 of file [W65C22.cs](#).

6.42.4.5 IER_EN `byte Hardware.W65C22.IER_EN = (byte) (1 << 7)`

Definition at line 32 of file [W65C22.cs](#).

6.42.4.6 IER_T1 `byte Hardware.W65C22.IER_T1 = (byte) (1 << 6)`

Definition at line 31 of file [W65C22.cs](#).

6.42.4.7 IER_T2 `byte Hardware.W65C22.IER_T2 = (byte) (1 << 5)`

Definition at line 30 of file [W65C22.cs](#).

6.42.4.8 IFR `int Hardware.W65C22.IFR = 0x0D`

Definition at line 20 of file [W65C22.cs](#).

6.42.4.9 IFR_INT `byte Hardware.W65C22.IFR_INT = (byte) (1 << 7)`

Definition at line 28 of file [W65C22.cs](#).

6.42.4.10 IFR_T1 `byte Hardware.W65C22.IFR_T1 = (byte) (1 << 6)`

Definition at line 27 of file [W65C22.cs](#).

6.42.4.11 IFR_T2 `byte Hardware.W65C22.IFR_T2 = (byte) (1 << 5)`

Definition at line 26 of file [W65C22.cs](#).

6.42.4.12 T1CH `int Hardware.W65C22.T1CH = 0x05`

Definition at line 16 of file [W65C22.cs](#).

6.42.4.13 T1CL `int Hardware.W65C22.T1CL = 0x04`

Definition at line 15 of file [W65C22.cs](#).

6.42.4.14 T1IsIRQ `readonly bool Hardware.W65C22.T1IsIRQ = false`

Definition at line 13 of file [W65C22.cs](#).

6.42.4.15 T2CH `int Hardware.W65C22.T2CH = 0x09`

Definition at line 18 of file [W65C22.cs](#).

6.42.4.16 T2CL `int Hardware.W65C22.T2CL = 0x08`

Definition at line 17 of file [W65C22.cs](#).

6.42.4.17 T2IsIRQ `readonly bool Hardware.W65C22.T2IsIRQ = true`

Definition at line 14 of file [W65C22.cs](#).

6.42.5 Property Documentation

6.42.5.1 End `int Hardware.W65C22.End [get]`

The end of memory

Definition at line 54 of file [W65C22.cs](#).

```
00054 { get { return Offset + Length; } }
```

6.42.5.2 Length `int Hardware.W65C22.Length [get], [set]`

The length of the device memory.

Definition at line 49 of file [W65C22.cs](#).

```
00049 { get; set; }
```

6.42.5.3 Memory `byte [] Hardware.W65C22.Memory [get], [set]`

The memory area.

Definition at line 39 of file [W65C22.cs](#).

```
00039 { get; set; }
```

6.42.5.4 Offset `int Hardware.W65C22.Offset [get], [set]`

The memory offset of the device.

Definition at line 44 of file [W65C22.cs](#).

```
00044 { get; set; }
```

6.42.5.5 Processor `W65C02 Hardware.W65C22.Processor [get], [set], [private]`

Local reference to the processor object.

Definition at line 118 of file [W65C22.cs](#).

```
00118 { get; set; }
```

6.42.5.6 T1Interval `double Hardware.W65C22.T1Interval [get]`

Set or check the timer 1 interval.

Definition at line 95 of file [W65C22.cs](#).

```
00095 { get { return (int)(Read(T1CL) | (Read(T1CH) << 8)); } }
```

6.42.5.7 T1IsEnabled `bool Hardware.W65C22.T1IsEnabled [get], [set]`

Enable or check whether timer 1 is enabled or not.

Definition at line 77 of file [W65C22.cs](#).

```
00078 {
00079     { get { return T1Object.Enabled; }
00080       set { T1Object.Enabled = value; }
00081 }
```

6.42.5.8 T1Object `Timer Hardware.W65C22.T1Object [get], [set]`

Set or get the timer 1 object.

Definition at line 108 of file [W65C22.cs](#).

```
00108 { get; set; }
```

6.42.5.9 T1TimerControl `bool Hardware.W65C22.T1TimerControl [get], [set]`

T1 timer control

Definition at line 59 of file [W65C22.cs](#).

```
00060 {
00061     { get { return T1Object.AutoReset; }
00062       set { T1Object.AutoReset = value; }
00063 }
```

6.42.5.10 T2Interval double Hardware.W65C22.T2Interval [get]

Set or check the timer 2 interval.

Definition at line 100 of file [W65C22.cs](#).

```
00101     {
00102         get { return (int)(Read(T2CL) | (Read(T2CH) << 8)); }
00103     }
```

6.42.5.11 T2IsEnabled bool Hardware.W65C22.T2IsEnabled [get], [set]

Enable or check whether timer 2 is enabled or not.

Definition at line 86 of file [W65C22.cs](#).

```
00087     {
00088         get { return T2Object.Enabled; }
00089         set { T2Object.Enabled = value; }
00090     }
```

6.42.5.12 T2Object Timer Hardware.W65C22.T2Object [get], [set]

Set or get the timer 2 object.

Definition at line 113 of file [W65C22.cs](#).

```
00113 { get; set; }
```

6.42.5.13 T2TimerControl bool Hardware.W65C22.T2TimerControl [get], [set]

T2 timer control.

Definition at line 68 of file [W65C22.cs](#).

```
00069     {
00070         get { return T2Object.AutoReset; }
00071         set { T2Object.AutoReset = value; }
00072     }
```

The documentation for this class was generated from the following file:

- [Hardware/Hardware/W65C22.cs](#)

6.43 Hardware.W65C51 Class Reference

An implementation of a [W65C51](#) ACIA.

Public Member Functions

- [W65C51](#) ([W65C02](#) processor, byte offset)
- void [Reset](#) ()
- void [Init](#) (string port)
Default Constructor, Instantiates a new instance of COM Port I/O.
- void [Init](#) (string port, int baudRate)
Default Constructor, Instantiates a new instance of COM Port I/O.
- void [Fini](#) ()
Called when the window is closed.
- byte [Read](#) (int address)
Returns the byte at a given address.
- void [Write](#) (int address, byte data)
Writes data to the given address.
- void [WriteCOM](#) (byte data)
Called in order to write to the serial port.

Public Attributes

- readonly int [defaultBaudRate](#) = 115200
- byte [byteIn](#)

Properties

- byte[] [Memory](#) [get, set]
- bool [IsEnabled](#) [get, set]
- SerialPort [Object](#) [get, set]
- string [ObjectName](#) [get, set]
- [W65C02 Processor](#) [get, set]
- BackgroundWorker [_backgroundWorker](#) [get, set]
- int [Offset](#) [get, set]
- int [Length](#) [get, set]
- bool [DataRead](#) [get, set]
- bool [EchoMode](#) [get, set]
- bool [InterruptDisabled](#) [get, set]
- bool [Interrupted](#) [get, set]
- bool [Overrun](#) [get, set]
- bool [ParityEnabled](#) [get, set]
- bool [ReceiverFull](#) [get, set]
- byte [RtsControl](#) [get, set]

Private Member Functions

- void [ComInit](#) (SerialPort serialPort)
Called whenever the ACIA is initialized.
- void [ComFini](#) (SerialPort serialPort)
Called when the window is closed.
- void [SerialDataReceived](#) (object sender, SerialDataReceivedEventArgs e)
Called whenever SerialDataReceivedEventHandler event occurs.
- void [HardwarePreWrite](#) (int address, byte data)
- void [HardwarePreRead](#) (int address)
- void [CommandRegister](#) (byte data)
- void [CommandRegisterUpdate](#) ()
- void [ControlRegister](#) (byte data)
- void [ControlRegisterUpdate](#) ()
- void [StatusRegisterUpdate](#) ()
- void [BackgroundWorkerDoWork](#) (object sender, DoWorkEventArgs e)

6.43.1 Detailed Description

An implementation of a [W65C51](#) ACIA.

Definition at line 12 of file [W65C51.cs](#).

6.43.2 Constructor & Destructor Documentation

6.43.2.1 W65C51() Hardware.W65C51.W65C51 (
 [W65C02](#) processor,
 byte offset) [inline]

Definition at line 40 of file [W65C51.cs](#).

```
00041     {
00042         if (offset > MemoryMap.DeviceArea.Length)
00043             throw new ArgumentException(String.Format("The offset: {0} is greater than the device
area: {1}", offset, MemoryMap.DeviceArea.Length));
00044
00045         Processor = processor;
00046
00047         Offset = MemoryMap.DeviceArea.Offset | offset;
00048         Length = 0x04;
00049         Memory = new byte[Length + 1];
00050
00051         _backgroundWorker = new BackgroundWorker
00052         {
00053             WorkerSupportsCancellation = true
00054         };
00055         _backgroundWorker.DoWork += BackgroundWorkerDoWork;
00056         _backgroundWorker.RunWorkerAsync();
00057     }
```

6.43.3 Member Function Documentation

6.43.3.1 BackgroundWorkerDoWork() void Hardware.W65C51.BackgroundWorkerDoWork (
 object sender,
 DoWorkEventArgs e) [inline], [private]

Definition at line 677 of file [W65C51.cs](#).

```
00678     {
00679         var worker = sender as BackgroundWorker;
00680
00681         while (true)
00682         {
00683             if (worker != null && worker.CancellationPending)
00684             {
00685                 e.Cancel = true;
00686                 return;
00687             }
00688
00689             if (Processor.isRunning)
00690             {
00691                 if (ReceiverFull || Overrun)
00692                 {
00693                     Memory[Offset + 1] = (byte)(Memory[Offset + 1] | 0x80);
00694                     Interrupted = true;
00695                     Processor.InterruptRequest();
00696                 }
00697
00698                 if (DataRead)
00699                 {
00700                     ReceiverFull = false;
00701                     Interrupted = false;
00702                     Overrun = false;
00703                     DataRead = false;
00704                 }
00705             }
00706         }
00707     }
```

6.43.3.2 ComFini() `void Hardware.W65C51.ComFini (`
`SerialPort serialPort) [inline], [private]`

Called when the window is closed.

Parameters

<i>serialPort</i>	SerialPort Object to close
-------------------	----------------------------

Definition at line 195 of file [W65C51.cs](#).

```
00196     {
00197         if (serialPort != null)
00198         {
00199             serialPort.Close();
00200         }
00201
00202         _backgroundWorker.CancelAsync();
00203         _backgroundWorker.DoWork -= BackgroundWorkerDoWork;
00204     }
```

6.43.3.3 ComInit() `void Hardware.W65C51.ComInit (`
`SerialPort serialPort) [inline], [private]`

Called whenever the ACIA is initialized.

Parameters

<i>serialPort</i>	SerialPort object to initialize.
-------------------	----------------------------------

Definition at line 147 of file [W65C51.cs](#).

```
00148     {
00149         try
00150         {
00151             serialPort.Open();
00152         }
00153         catch (UnauthorizedAccessException w)
00154         {
00155             FileStream file = new FileStream(FileLocations.ErrorFile, FileMode.OpenOrCreate,
FileAccess.ReadWrite);
00156             StreamWriter stream = new StreamWriter(file);
00157             stream.WriteLine(w.Message);
00158             stream.WriteLine(w.Source);
00159             stream.Flush();
00160             file.Flush();
00161             stream.Close();
00162             file.Close();
00163             return;
00164         }
00165         serialPort.ReadTimeout = 50;
00166         serialPort.WriteTimeout = 50;
00167         serialPort.DataReceived += new SerialDataReceivedEventHandler(SerialDataReceived);
00168         try
00169         {
00170             serialPort.Write("-----\r\n");
00171             serialPort.Write(" WolfNet 6502 WBC Emulator\r\n");
00172             serialPort.Write("-----\r\n");
00173             serialPort.Write("\r\n");
00174         }
00175         catch (TimeoutException t)
00176         {
00177             _ = t;
00178             FileStream file = new FileStream(FileLocations.ErrorFile, FileMode.OpenOrCreate,
FileAccess.ReadWrite);
00179             StreamWriter stream = new StreamWriter(file);
00180             stream.WriteLine("Read/Write error: Port timed out!");
00181             stream.WriteLine("Please ensure all cables are connected properly!");
00182             stream.Flush();
00183             file.Flush();
00184         }
```

```

00184         stream.Close();
00185         file.Close();
00186         return;
00187     }
00188 }

```

6.43.3.4 CommandRegister() void Hardware.W65C51.CommandRegister (byte data) [inline], [private]

Definition at line 296 of file [W65C51.cs](#).

```

00297 {
00298     byte test = (byte)(data & 0x20);
00299     if (test == 0x20)
00300     {
00301         throw new ArgumentException("Parity must NEVER be enabled!");
00302     }
00303
00304     test = (byte)(data & 0x10);
00305     if (test == 0x10)
00306     {
00307         EchoMode = true;
00308     }
00309     else
00310     {
00311         EchoMode = false;
00312     }
00313
00314     test = (byte)(data & 0x0C);
00315     if (test == 0x00)
00316     {
00317         Object.Handshake = Handshake.None;
00318         Object.RtsEnable = true;
00319         Object.Handshake = Handshake.RequestToSend;
00320     }
00321     else if (test == 0x04)
00322     {
00323         Object.Handshake = Handshake.None;
00324         Object.RtsEnable = false;
00325     }
00326     else if ((test == 0x08) || (test == 0x0C))
00327     {
00328         throw new NotImplementedException("This cannot be emulated on windows!");
00329     }
00330     else
00331     {
00332         throw new ArgumentOutOfRangeException("RtsControl is invalid!");
00333     }
00334
00335     test = (byte)(data & 0x02);
00336     if (test == 0x02)
00337     {
00338         InterruptDisabled = true;
00339     }
00340     else
00341     {
00342         InterruptDisabled = false;
00343     }
00344
00345     test = (byte)(data & 0x01);
00346     if (test == 0x01)
00347     {
00348         Object.DtrEnable = true;
00349     }
00350     else
00351     {
00352         Object.DtrEnable = false;
00353     }
00354 }

```

6.43.3.5 CommandRegisterUpdate() void Hardware.W65C51.CommandRegisterUpdate () [inline], [private]

Definition at line 356 of file [W65C51.cs](#).


```

00357     {
00358         byte data = Memory[Offset + 2];
00359
00360         if (ParityEnabled)
00361         {
00362             data |= 0x20;
00363         }
00364         else
00365         {
00366             data &= 0xD0;
00367         }
00368
00369         if (EchoMode)
00370         {
00371             data |= 0x10;
00372         }
00373         else
00374         {
00375             data &= 0xE0;
00376         }
00377
00378         data &= RtsControl;
00379
00380         if (InterruptDisabled)
00381         {
00382             data |= 0x02;
00383         }
00384         else
00385         {
00386             data &= 0x0D;
00387         }
00388         if (Object.DtrEnable)
00389         {
00390             data |= 0x01;
00391         }
00392         else
00393         {
00394             data &= 0x0E;
00395         }
00396
00397         Memory[Offset + 2] = data;
00398     }

```

6.43.3.6 ControlRegister() void Hardware.W65C51.ControlRegister (byte data) [inline], [private]

Definition at line 400 of file [W65C51.cs](#).

```

00401     {
00402         byte test = (byte)(data & 0x80);
00403         if (test == 0x80)
00404         {
00405             test = (byte)(data & 0x60);
00406             if (test == 0x60)
00407             {
00408                 Object.StopBits = StopBits.OnePointFive;
00409             }
00410             else
00411             {
00412                 Object.StopBits = StopBits.Two;
00413             }
00414         }
00415         else
00416         {
00417             Object.StopBits = StopBits.One;
00418         }
00419
00420         test = (byte)(data & 0x60);
00421         if (test == 0x20)
00422         {
00423             Object.DataBits = 7;
00424         }
00425         else if (test == 0x40)
00426         {
00427             Object.DataBits = 6;
00428         }
00429         else if (test == 0x60)
00430         {
00431             Object.DataBits = 5;
00432         }
00433         else

```

```

00434         {
00435             Object.DataBits = 8;
00436         }
00437
00438         test = (byte)(data & 0x10);
00439         if (!(test == 0x10))
00440         {
00441             throw new ArgumentException("External clock rate not available on the WolfNet 65C02
WBC!");
00442         }
00443
00444         test = (byte)(data & 0x0F);
00445         if (test == 0x00)
00446         {
00447             Object.BaudRate = 115200;
00448         }
00449         else if (test == 0x01)
00450         {
00451             Object.BaudRate = 50;
00452         }
00453         else if (test == 0x02)
00454         {
00455             Object.BaudRate = 75;
00456         }
00457         else if (test == 0x03)
00458         {
00459             Object.BaudRate = 110;
00460         }
00461         else if (test == 0x04)
00462         {
00463             Object.BaudRate = 135;
00464         }
00465         else if (test == 0x05)
00466         {
00467             Object.BaudRate = 150;
00468         }
00469         else if (test == 0x06)
00470         {
00471             Object.BaudRate = 300;
00472         }
00473         else if (test == 0x07)
00474         {
00475             Object.BaudRate = 600;
00476         }
00477         else if (test == 0x08)
00478         {
00479             Object.BaudRate = 1200;
00480         }
00481         else if (test == 0x09)
00482         {
00483             Object.BaudRate = 1800;
00484         }
00485         else if (test == 0x0A)
00486         {
00487             Object.BaudRate = 2400;
00488         }
00489         else if (test == 0x0B)
00490         {
00491             Object.BaudRate = 3600;
00492         }
00493         else if (test == 0x0C)
00494         {
00495             Object.BaudRate = 4800;
00496         }
00497         else if (test == 0x0D)
00498         {
00499             Object.BaudRate = 7200;
00500         }
00501         else if (test == 0x0E)
00502         {
00503             Object.BaudRate = 9600;
00504         }
00505         else
00506         {
00507             Object.BaudRate = 19200;
00508         }
00509     }

```

6.43.3.7 ControlRegisterUpdate() void Hardware.W65C51.ControlRegisterUpdate () [inline],
[private]

Definition at line 511 of file W65C51.cs.

```

00512     {
00513         byte controlRegister = Memory[Offset + 3];
00514
00515         if (Object.StopBits == StopBits.Two)
00516         {
00517             controlRegister |= 0x80;
00518         }
00519         else if ((Object.StopBits == StopBits.OnePointFive) && (Object.DataBits == 5) ||
00520 (Object.StopBits == StopBits.One))
00521         {
00522             controlRegister &= 0x7F;
00523         }
00524         else
00525         {
00526             throw new ArgumentOutOfRangeException("StopBits or combination of StopBits and
DataBits is invalid!");
00527         }
00528         if (Object.DataBits == 8)
00529         {
00530             controlRegister &= 0x9F;
00531         }
00532         else if (Object.DataBits == 7)
00533         {
00534             controlRegister |= 0x20;
00535         }
00536         else if (Object.DataBits == 6)
00537         {
00538             controlRegister |= 0x40;
00539         }
00540         else if (Object.DataBits == 5)
00541         {
00542             controlRegister |= 0x60;
00543         }
00544         else
00545         {
00546             throw new ArgumentOutOfRangeException("DataBits is out of range!");
00547         }
00548         if (Object.BaudRate == 115200)
00549         {
00550             controlRegister &= 0xF0;
00551         }
00552         else if (Object.BaudRate == 50)
00553         {
00554             controlRegister |= 0x01;
00555         }
00556         else if (Object.BaudRate == 75)
00557         {
00558             controlRegister |= 0x02;
00559         }
00560         else if (Object.BaudRate == 110)
00561         {
00562             controlRegister |= 0x03;
00563         }
00564         else if (Object.BaudRate == 135)
00565         {
00566             controlRegister |= 0x04;
00567         }
00568         else if (Object.BaudRate == 150)
00569         {
00570             controlRegister |= 0x05;
00571         }
00572         else if (Object.BaudRate == 300)
00573         {
00574             controlRegister |= 0x06;
00575         }
00576         else if (Object.BaudRate == 600)
00577         {
00578             controlRegister |= 0x07;
00579         }
00580         else if (Object.BaudRate == 1200)
00581         {
00582             controlRegister |= 0x08;
00583         }
00584         else if (Object.BaudRate == 1800)
00585         {
00586             controlRegister |= 0x09;
00587         }
00588         else if (Object.BaudRate == 2400)
00589         {
00590             controlRegister |= 0x0A;
00591         }
00592         else if (Object.BaudRate == 3600)
00593         {
00594             controlRegister |= 0x0B;
00595         }

```

```

00596         }
00597         else if (Object.BaudRate == 4800)
00598         {
00599             controlRegister |= 0x0C;
00600         }
00601         else if (Object.BaudRate == 7200)
00602         {
00603             controlRegister |= 0x0D;
00604         }
00605         else if (Object.BaudRate == 9600)
00606         {
00607             controlRegister |= 0x0E;
00608         }
00609         else if (Object.BaudRate == 19200)
00610         {
00611             controlRegister |= 0x0F;
00612         }
00613         else
00614         {
00615             throw new ArgumentOutOfRangeException("BaudRate is outside the range of Baud Rates
supported by the W65C51!");
00616         }
00617         Memory[Offset + 3] = controlRegister;
00618     }
00619 }

```

6.43.3.8 Fini() void Hardware.W65C51.Fini () [inline]

Called when the window is closed.

Definition at line 94 of file [W65C51.cs](#).

```

00095     {
00096         ComFini(Object);
00097     }

```

6.43.3.9 HardwarePreRead() void Hardware.W65C51.HardwarePreRead (int address) [inline], [private]

Definition at line 273 of file [W65C51.cs](#).

```

00274     {
00275         if (address == Offset)
00276         {
00277             Interrupted = false;
00278             Overrun = false;
00279             ReceiverFull = false;
00280         }
00281         else if (address == Offset + 1)
00282         {
00283             StatusRegisterUpdate();
00284         }
00285         else if (address == Offset + 2)
00286         {
00287             CommandRegisterUpdate();
00288         }
00289         else if (address == Offset + 3)
00290         {
00291             ControlRegisterUpdate();
00292         }
00293     }
00294 }

```

6.43.3.10 HardwarePreWrite() `void Hardware.W65C51.HardwarePreWrite (`
 `int address,`
 `byte data) [inline], [private]`

Definition at line 253 of file [W65C51.cs](#).

```
00254     {  
00255         if (address == Offset)  
00256         {  
00257             WriteCOM(data);  
00258         }  
00259         else if (address == Offset + 1)  
00260         {  
00261             Reset();  
00262         }  
00263         else if (address == Offset + 2)  
00264         {  
00265             CommandRegister(data);  
00266         }  
00267         else if (address == Offset + 3)  
00268         {  
00269             ControlRegister(data);  
00270         }  
00271     }
```

6.43.3.11 Init() [1/2] `void Hardware.W65C51.Init (`
 `string port) [inline]`

Default Constructor, Instantiates a new instance of COM Port I/O.

Parameters

<i>port</i>	COM Port to use for I/O
-------------	-------------------------

Definition at line 69 of file [W65C51.cs](#).

```
00070     {  
00071         Object = new SerialPort(port, defaultBaudRate, Parity.None, 8, StopBits.One);  
00072         ObjectName = port;  
00073         ComInit(Object);  
00074     }  
00075 }
```

6.43.3.12 Init() [2/2] `void Hardware.W65C51.Init (`
 `string port,`
 `int baudRate) [inline]`

Default Constructor, Instantiates a new instance of COM Port I/O.

Parameters

<i>port</i>	COM Port to use for I/O
<i>baudRate</i>	Baud Rate to use for I/O

Definition at line 83 of file [W65C51.cs](#).

```
00084     {  
00085         Object = new SerialPort(port, baudRate, Parity.None, 8, StopBits.One);  
00086         ObjectName = port;  
00087         ComInit(Object);  
00088     }  
00089 }
```

6.43.3.13 Read() `byte Hardware.W65C51.Read (int address) [inline]`

Returns the byte at a given address.

Parameters

<i>address</i>	
----------------	--

Returns

the byte being returned

Definition at line 106 of file [W65C51.cs](#).

```
00107     {
00108         HardwarePreRead(address);
00109         byte data = Memory[address - Offset];
00110         DataRead = true;
00111         return data;
00112     }
```

6.43.3.14 Reset() `void Hardware.W65C51.Reset () [inline]`

Definition at line 59 of file [W65C51.cs](#).

```
00060     {
00061         IsEnabled = false;
00062     }
```

6.43.3.15 SerialDataReceived() `void Hardware.W65C51.SerialDataReceived (object sender, SerialDataReceivedEventArgs e) [inline], [private]`

Called whenever SerialDataReceivedEventHandler event occurs.

Parameters

<i>sender</i>	
<i>e</i>	

Definition at line 212 of file [W65C51.cs](#).

```
00213     {
00214         try
00215         {
00216             if (EchoMode)
00217             {
00218                 WriteCOM(Convert.ToByte(Object.ReadByte()));
00219             }
00220             else
00221             {
00222                 if (!ReceiverFull)
00223                 {
00224                     ReceiverFull = true;
```

```

00225         }
00226         else
00227         {
00228             Overrun = true;
00229         }
00230         Memory[0] = Convert.ToByte(Object.ReadByte());
00231     }
00232
00233     if (!InterruptDisabled)
00234     {
00235         Interrupted = true;
00236         Processor.InterruptRequest();
00237     }
00238 }
00239 catch (Win32Exception w)
00240 {
00241     FileStream file = new FileStream(FileLocations.ErrorFile, FileMode.OpenOrCreate,
    FileAccess.ReadWrite);
00242     StreamWriter stream = new StreamWriter(file);
00243     stream.WriteLine(w.Message);
00244     stream.WriteLine(w.ErrorCode.ToString());
00245     stream.WriteLine(w.Source);
00246     stream.Flush();
00247     stream.Close();
00248     file.Flush();
00249     file.Close();
00250 }
00251 }

```

6.43.3.16 StatusRegisterUpdate() void Hardware.W65C51.StatusRegisterUpdate () [inline], [private]

Definition at line 621 of file W65C51.cs.

```

00622 {
00623     byte statusRegister = Memory[Offset + 1];
00624
00625     if (Interrupted)
00626     {
00627         statusRegister |= 0x80;
00628     }
00629     else
00630     {
00631         statusRegister &= 0x7F;
00632     }
00633
00634     if (Object.DsrHolding == false)
00635     {
00636         statusRegister |= 0x40;
00637     }
00638     else
00639     {
00640         statusRegister &= 0xBF;
00641     }
00642
00643     if (Object.CDHolding)
00644     {
00645         statusRegister |= 0x20;
00646     }
00647     else
00648     {
00649         statusRegister &= 0xDF;
00650     }
00651
00652     statusRegister |= 0x10;
00653
00654     if (ReceiverFull)
00655     {
00656         statusRegister |= 0x08;
00657     }
00658     else
00659     {
00660         statusRegister &= 0xF7;
00661     }
00662
00663     if (Overrun)
00664     {
00665         statusRegister |= 0x04;
00666     }
00667     else
00668     {
00669         statusRegister &= 0xFB;

```

```

00670         }
00671
00672         statusRegister &= 0xFC;
00673
00674         Memory[Offset + 1] = statusRegister;
00675     }

```

6.43.3.17 Write() void Hardware.W65C51.Write (

```

    int address,
    byte data ) [inline]

```

Writes data to the given address.

Parameters

<i>address</i>	The address to write data to
<i>data</i>	The data to write

Definition at line 120 of file [W65C51.cs](#).

```

00121     {
00122         HardwarePreWrite(address, data);
00123         if (!(address == Offset) || (address == Offset + 1))
00124         {
00125             Memory[address - Offset] = data;
00126         }
00127     }

```

6.43.3.18 WriteCOM() void Hardware.W65C51.WriteCOM (

```

    byte data ) [inline]

```

Called in order to write to the serial port.

Parameters

<i>data</i>	Byte of data to send
-------------	----------------------

Definition at line 134 of file [W65C51.cs](#).

```

00135     {
00136         byte[] writeByte = new byte[] { data };
00137         Object.Write(writeByte, 0, 1);
00138     }

```

6.43.4 Member Data Documentation

6.43.4.1 byteIn byte Hardware.W65C51.byteIn

Definition at line 16 of file [W65C51.cs](#).

6.43.4.2 defaultBaudRate readonly int Hardware.W65C51.defaultBaudRate = 115200

Definition at line 15 of file [W65C51.cs](#).

6.43.5 Property Documentation

6.43.5.1 _backgroundWorker BackgroundWorker Hardware.W65C51._backgroundWorker [get], [set], [private]

Definition at line 25 of file [W65C51.cs](#).

```
00025 { get; set; }
```

6.43.5.2 DataRead bool Hardware.W65C51.DataRead [get], [set], [private]

Definition at line 29 of file [W65C51.cs](#).

```
00029 { get; set; }
```

6.43.5.3 EchoMode bool Hardware.W65C51.EchoMode [get], [set], [private]

Definition at line 30 of file [W65C51.cs](#).

```
00030 { get; set; }
```

6.43.5.4 InterruptDisabled bool Hardware.W65C51.InterruptDisabled [get], [set], [private]

Definition at line 31 of file [W65C51.cs](#).

```
00031 { get; set; }
```

6.43.5.5 Interrupted bool Hardware.W65C51.Interrupted [get], [set], [private]

Definition at line 32 of file [W65C51.cs](#).

```
00032 { get; set; }
```

6.43.5.6 IsEnabled bool Hardware.W65C51.IsEnabled [get], [set]

Definition at line 21 of file [W65C51.cs](#).

```
00021 { get; set; }
```

6.43.5.7 Length `int Hardware.W65C51.Length [get], [set]`

Definition at line 27 of file [W65C51.cs](#).

```
00027 { get; set; }
```

6.43.5.8 Memory `byte [] Hardware.W65C51.Memory [get], [set]`

Definition at line 20 of file [W65C51.cs](#).

```
00020 { get; set; }
```

6.43.5.9 Object `SerialPort Hardware.W65C51.Object [get], [set]`

Definition at line 22 of file [W65C51.cs](#).

```
00022 { get; set; }
```

6.43.5.10 ObjectName `string Hardware.W65C51.ObjectName [get], [set]`

Definition at line 23 of file [W65C51.cs](#).

```
00023 { get; set; }
```

6.43.5.11 Offset `int Hardware.W65C51.Offset [get], [set]`

Definition at line 26 of file [W65C51.cs](#).

```
00026 { get; set; }
```

6.43.5.12 Overrun `bool Hardware.W65C51.Overrun [get], [set], [private]`

Definition at line 33 of file [W65C51.cs](#).

```
00033 { get; set; }
```

6.43.5.13 ParityEnabled `bool Hardware.W65C51.ParityEnabled [get], [set], [private]`

Definition at line 34 of file [W65C51.cs](#).

```
00034 { get; set; }
```

6.43.5.14 Processor [W65C02](#) Hardware.W65C51.Processor [get], [set], [private]

Definition at line 24 of file [W65C51.cs](#).

```
00024 { get; set; }
```

6.43.5.15 ReceiverFull bool Hardware.W65C51.ReceiverFull [get], [set], [private]

Definition at line 35 of file [W65C51.cs](#).

```
00035 { get; set; }
```

6.43.5.16 RtsControl byte Hardware.W65C51.RtsControl [get], [set], [private]

Definition at line 36 of file [W65C51.cs](#).

```
00036 { get; set; }
```

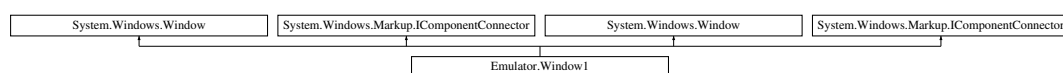
The documentation for this class was generated from the following file:

- Hardware/Hardware/[W65C51.cs](#)

6.44 Emulator.Window1 Class Reference

[Window1](#)

Inheritance diagram for Emulator.Window1:



Public Member Functions

- void [InitializeComponent](#) ()
InitializeComponent
- void [InitializeComponent](#) ()
InitializeComponent

Private Member Functions

- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)

Private Attributes

- bool [_contentLoaded](#)

6.44.1 Detailed Description

Window1

Definition at line 41 of file [MemoryMap.g.i.cs](#).

6.44.2 Member Function Documentation

6.44.2.1 Connect() [1/2] void System.Windows.Markup.IComponentConnector. Emulator.Window1.↔

```
Connect (
    int connectionId,
    object target ) [inline], [private]
```

Definition at line 78 of file [MemoryMap.g.i.cs](#).

```
00078                                     {
00079         switch (connectionId)
00080         {
00081             case 1:
00082                 this.MemoryMap = ((System.Windows.Controls.DataGrid) (target));
00083                 return;
00084             }
00085         this._contentLoaded = true;
00086     }
```

6.44.2.2 Connect() [2/2] void System.Windows.Markup.IComponentConnector. Emulator.Window1.↔

```
Connect (
    int connectionId,
    object target ) [inline], [private]
```

Definition at line 75 of file [Window1.g.i.cs](#).

```
00076     {
00077         this._contentLoaded = true;
00078     }
```

6.44.2.3 InitializeComponent() [1/2] void Emulator.Window1.InitializeComponent () [inline]

InitializeComponent

Definition at line 58 of file [MemoryMap.g.i.cs](#).

```
00058                                     {
00059         if (_contentLoaded) {
00060             return;
00061         }
00062         _contentLoaded = true;
00063         System.Uri resourceLocator = new System.Uri("/Emulator;component/memorymap.xaml",
00064             System.UriKind.Relative);
00065         #line 1 "..\..\..\MemoryMap.xaml"
00066         System.Windows.Application.LoadComponent(this, resourceLocator);
00067         #line default
00068         #line hidden
00070     }
```

6.44.2.4 InitializeComponent() [2/2] `void Emulator.Window1.InitializeComponent () [inline]`

InitializeComponent

Definition at line 53 of file [Window1.g.i.cs](#).

```
00054     {
00055         if (_contentLoaded)
00056         {
00057             return;
00058         }
00059         _contentLoaded = true;
00060         System.Uri resourceLocator = new System.Uri("/Emulator;component/window1.xaml",
            System.UriKind.Relative);
00061
00062 #line 1 "..\..\..\Window1.xaml"
00063         System.Windows.Application.LoadComponent(this, resourceLocator);
00064
00065 #line default
00066 #line hidden
00067     }
```

6.44.3 Member Data Documentation

6.44.3.1 `_contentLoaded` `bool Emulator.Window1._contentLoaded [private]`

Definition at line 51 of file [MemoryMap.g.i.cs](#).

The documentation for this class was generated from the following files:

- [Emulator/obj/x86/Release/MemoryMap.g.i.cs](#)
- [Emulator/obj/x86/Release/Window1.g.i.cs](#)

7 File Documentation

7.1 Emulator/App.xaml.cs File Reference

Classes

- class [Emulator.App](#)
Interaction logic for App.xaml

Namespaces

- namespace [Emulator](#)

7.2 App.xaml.cs

[Go to the documentation of this file.](#)

```
00001 namespace Emulator
00002 {
00003     /// <summary>
00004     /// Interaction logic for App.xaml
00005     /// </summary>
00006     public partial class App
00007     {
00008     }
00009 }
```

7.3 Emulator/Classes/ExitCodes.cs File Reference

Classes

- class [Emulator.ExitCodes](#)

Namespaces

- namespace [Emulator](#)

7.4 ExitCodes.cs

[Go to the documentation of this file.](#)

```
00001 namespace Emulator
00002 {
00003     public class ExitCodes
00004     {
00005         public static readonly int NO_ERROR = 0x00;
00006
00007         public static readonly int USER_ERROR = 0x01;
00008
00009         public static readonly int NO_BIOS = 0x02;
00010         public static readonly int LOAD_BIOS_FILE_ERROR = 0x03;
00011         public static readonly int BIOS_LOADPROGRAM_ERROR = 0x04;
00012         public static readonly int LOAD_ROM_FILE_ERROR = 0x05;
00013         public static readonly int ROM_LOADPROGRAM_ERROR = 0x06;
00014         public static readonly int LOAD_STATE_ERROR = 0x07;
00015     }
00016 }
```

7.5 Emulator/Classes/FileLocations.cs File Reference

Namespaces

- namespace [Emulator](#)

7.6 FileLocations.cs

[Go to the documentation of this file.](#)

```
00001 namespace Emulator
00002 {
00003     internal class FileLocations
00004     {
00005         #region Fields
00006         public static string SettingsFile = "./Settings.xml";
00007         public static string ErrorFile = "./Errors.log";
00008         #if DEBUG
00009         public static string BiosFile = "../../../bios.bin";
00010         #else
00011         public static string BiosFile = "./bios.bin";
00012         #endif
00013     #endregion
00014     }
00015 }
```

7.7 Hardware/Classes/FileLocations.cs File Reference

Namespaces

- namespace [Hardware](#)

7.8 FileLocations.cs

[Go to the documentation of this file.](#)

```
00001 namespace Hardware
00002 {
00003     internal class FileLocations
00004     {
00005 #region Fields
00006         public static string ErrorFile = "./Hardware_Library_Errors.log";
00007         public static string MemoryDump = "./Hardware_Library_Memory_Dump.log";
00008 #endregion
00009     }
00010 }
```

7.9 Emulator/Classes/SettingsFile.cs File Reference

Classes

- class [Emulator.SettingsFile](#)

Namespaces

- namespace [Emulator](#)

7.10 SettingsFile.cs

[Go to the documentation of this file.](#)

```
00001 using Emulator.Model;
00002
00003 namespace Emulator
00004 {
00005     public static class SettingsFile
00006     {
00007         public static SettingsModel CreateNew()
00008         {
00009             // Create new settings file.
00010             SettingsModel _settings = new SettingsModel
00011             {
00012                 SettingsVersionMajor = Versioning.SettingsFile.Major,
00013                 SettingsVersionMinor = Versioning.SettingsFile.Minor,
00014                 SettingsVersionBuild = Versioning.SettingsFile.Build,
00015                 SettingsVersionRevision = Versioning.SettingsFile.Revision,
00016 #if DEBUG
00017                 ComPortName = "COM9",
00018 #else
00019                 ComPortName = "COM1",
00020 #endif
00021             };
00022             return _settings;
00023         }
00024     }
00025 }
```

7.11 Emulator/Classes/Versioning.cs File Reference

Classes

- class [Emulator.Versioning](#)
- class [Emulator.Versioning.Product](#)
- class [Emulator.Versioning.SettingsFile](#)

Namespaces

- namespace [Emulator](#)

7.12 Versioning.cs

[Go to the documentation of this file.](#)

```

00001 namespace Emulator
00002 {
00003     public static class Versioning
00004     {
00005         public class Product
00006         {
00007             public const int Major = 0;
00008             public const int Minor = 1;
00009             public const int Build = 3;
00010             public const int Revision = 1;
00011             public const string Title = Name;
00012             public const string Name = "WolfNet 65C02 WorkBench Computer Emulator";
00013             public const string Company = "WolfNet Computing";
00014             public const string Copyright = "Copyright 1' WolfNet Computing 2022";
00015             public const string VersionString = "0.2.4.1";
00016             public const string Description = "Emulator for the WolfNet 65C02 WorkBench Computer coded
in C# using the .NET Framework";
00017         }
00018         public class SettingsFile
00019         {
00020             public const byte Major = 1;
00021             public const byte Minor = 0;
00022             public const byte Build = 0;
00023             public const byte Revision = 0;
00024         }
00025     }
00026 }
00027 }
```

7.13 Hardware/Classes/Versioning.cs File Reference

Classes

- class [Hardware.Versioning.Product](#)

Namespaces

- namespace [Hardware](#)

7.14 Versioning.cs

[Go to the documentation of this file.](#)

```

00001 namespace Hardware
00002 {
00003     internal class Versioning
00004     {
00005         public class Product
00006         {
00007             public const string Title = Name;
00008             public const string Name = "WolfNet 65C02 Hardware Library";
00009             public const string Company = "WolfNet Computing";
00010             public const string Copyright = "Copyright 1' WolfNet Computing 2022";
00011             public const string Version = "1.3.0.0";
00012             public const string Description = "65C02 Hardware Library, coded in C# using the .NET
Framework";
00013         }
00014     }
00015 }
```


7.15 Emulator/Interfaces/IClosable.cs File Reference

Classes

- interface [Emulator.IClosable](#)

Namespaces

- namespace [Emulator](#)

7.16 IClosable.cs

[Go to the documentation of this file.](#)

```
00001 namespace Emulator
00002 {
00003     public interface IClosable
00004     {
00005         void Close();
00006     }
00007 }
```

7.17 Emulator/MainWindow.xaml.cs File Reference

Classes

- class [Emulator.MainWindow](#)
Interaction logic for MainWindow.xaml

Namespaces

- namespace [Emulator](#)

7.18 MainWindow.xaml.cs

[Go to the documentation of this file.](#)

```
00001 using Emulator.Model;
00002 using Emulator.ViewModel;
00003 using GalaSoft.MvvmLight.Messaging;
00004 using System;
00005 using System.Windows;
00006
00007 namespace Emulator
00008 {
00009     /// <summary>
00010     /// Interaction logic for MainWindow.xaml
00011     /// </summary>
00012     public partial class MainWindow : Window, IClosable
00013     {
00014         public MainWindow()
00015         {
00016             InitializeComponent();
00017             Messenger.Default.Register<NotificationMessage>(this, NotificationMessageReceived);
00018             Messenger.Default.Register<NotificationMessage<SettingsModel>>(this,
NotificationMessageReceived);
00019         }
00020
00021         private void ToClose(Object sender, EventArgs e)
00022         {
00023             Close();
00024         }
00025     }
00026 }
```

```

00025
00026     private void LoadFile(Object sender, EventArgs e)
00027     {
00028         Messenger.Default.Send(new NotificationMessage("LoadFile"));
00029     }
00030
00031     private void SaveFile(Object sender, EventArgs e)
00032     {
00033         Messenger.Default.Send(new NotificationMessage("SaveState"));
00034     }
00035
00036     private void CloseFile(Object sender, EventArgs e)
00037     {
00038         Messenger.Default.Send(new NotificationMessage("CloseFile"));
00039     }
00040
00041     private void NotificationMessageReceived(NotificationMessage notificationMessage)
00042     {
00043         if (notificationMessage.Notification == "CloseWindow")
00044         {
00045             Close();
00046         }
00047         else if (notificationMessage.Notification == "MemoryVisualWindow")
00048         {
00049             var memoryVisual = new MemoryVisual { DataContext = new MemoryVisualViewModel() };
00050             memoryVisual.Show();
00051         }
00052     }
00053
00054     private void NotificationMessageReceived(NotificationMessage<SettingsModel>
notificationMessage)
00055     {
00056         if (notificationMessage.Notification == "SettingsWindow")
00057         {
00058             var settingsFile = new Settings { DataContext = new
SettingsViewModel(notificationMessage.Content) };
00059             settingsFile.ShowDialog();
00060         }
00061     }
00062 }
00063 }

```

7.19 Emulator/MemoryVisual.xaml.cs File Reference

Classes

- class [Emulator.MemoryVisual](#)
Interaction logic for Window1.xaml

Namespaces

- namespace [Emulator](#)

7.20 MemoryVisual.xaml.cs

[Go to the documentation of this file.](#)

```

00001 using System.Windows;
00002
00003 namespace Emulator
00004 {
00005     /// <summary>
00006     /// Interaction logic for Window1.xaml
00007     /// </summary>
00008     public partial class MemoryVisual : Window
00009     {
00010         public MemoryVisual()
00011         {
00012             InitializeComponent();
00013         }
00014     }
00015 }

```

7.21 Emulator/Model/Breakpoint.cs File Reference

Classes

- class [Emulator.Model.Breakpoint](#)
A Representation of a [Breakpoint](#)

Namespaces

- namespace [Emulator](#)
- namespace [Emulator.Model](#)

7.22 Breakpoint.cs

[Go to the documentation of this file.](#)

```
00001 using System.Collections.Generic;
00002
00003 namespace Emulator.Model
00004 {
00005     /// <summary>
00006     /// A Representation of a Breakpoint
00007     /// </summary>
00008     public class Breakpoint
00009     {
00010         /// <summary>
00011         /// Is the Breakpoint enabled or disabled
00012         /// </summary>
00013         public bool IsEnabled { get; set; }
00014
00015         /// <summary>
00016         /// The Value of the Breakpoint
00017         /// </summary>
00018         public string Value { get; set; }
00019
00020         /// <summary>
00021         /// The Type of breakpoint being set
00022         /// </summary>
00023         public string Type { get; set; }
00024
00025         public List<string> AllTypes
00026         {
00027             get { return BreakpointType.AllTypes; }
00028         }
00029     }
00030 }
```

7.23 Emulator/Model/BreakpointType.cs File Reference

Classes

- class [Emulator.Model.BreakpointType](#)
The Type of [Breakpoint](#)

Namespaces

- namespace [Emulator](#)
- namespace [Emulator.Model](#)

7.24 BreakpointType.cs

[Go to the documentation of this file.](#)

```
00001 using System.Collections.Generic;
00002
00003 namespace Emulator.Model
00004 {
00005     /// <summary>
00006     /// The Type of Breakpoint
00007     /// </summary>
00008     public class BreakpointType
00009     {
00010         /// <summary>
00011         /// A Listing of all of the Current Types
00012         /// </summary>
00013         public static List<string> AllTypes = new List<string>
00014         {
00015             ProgramCounterType,
00016             NumberOfCycleType
00017         };
00018
00019         /// <summary>
00020         /// The ProgramCounter Breakpoint Type
00021         /// </summary>
00022         public const string ProgramCounterType = "Program Counter";
00023
00024         /// <summary>
00025         /// The CycleCount Breakpoint Type
00026         /// </summary>
00027         public const string NumberOfCycleType = "Number of Cycles";
00028     }
00029 }
00030 }
```

7.25 Emulator/Model/MemoryRowModel.cs File Reference

Classes

- class [Emulator.Model.MemoryRowModel](#)
A *Model* of a Single Page of memory

Namespaces

- namespace [Emulator](#)
- namespace [Emulator.Model](#)

7.26 MemoryRowModel.cs

[Go to the documentation of this file.](#)

```
00001 namespace Emulator.Model
00002 {
00003     /// <summary>
00004     /// A Model of a Single Page of memory
00005     /// </summary>
00006     public class MemoryRowModel
00007     {
00008         /// <summary>
00009         /// The offset of this row. Expressed in hex
00010         /// </summary>
00011         public string Offset { get; set; }
00012         /// <summary>
00013         /// The memory at the location offset + 00
00014         /// </summary>
00015         public string Location00 { get; set; }
00016         /// <summary>
00017         /// The memory at the location offset + 01
00018         /// </summary>
00019         public string Location01 { get; set; }
00020     }
00021 }
```

```
00021 /// The memory at the location offset + 02
00022 /// </summary>
00023     public string Location02 { get; set; }
00024 /// <summary>
00025 /// The memory at the location offset + 03
00026 /// </summary>
00027     public string Location03 { get; set; }
00028 /// <summary>
00029 /// The memory at the location offset + 04
00030 /// </summary>
00031     public string Location04 { get; set; }
00032 /// <summary>
00033 /// The memory at the location offset + 05
00034 /// </summary>
00035     public string Location05 { get; set; }
00036 /// <summary>
00037 /// The memory at the location offset + 06
00038 /// </summary>
00039     public string Location06 { get; set; }
00040 /// <summary>
00041 /// The memory at the location offset + 07
00042 /// </summary>
00043     public string Location07 { get; set; }
00044 /// <summary>
00045 /// The memory at the location offset + 08
00046 /// </summary>
00047     public string Location08 { get; set; }
00048 /// <summary>
00049 /// The memory at the location offset + 09
00050 /// </summary>
00051     public string Location09 { get; set; }
00052 /// <summary>
00053 /// The memory at the location offset + 0A
00054 /// </summary>
00055     public string Location0A { get; set; }
00056 /// <summary>
00057 /// The memory at the location offset + 0B
00058 /// </summary>
00059     public string Location0B { get; set; }
00060 /// <summary>
00061 /// The memory at the location offset + 0C
00062 /// </summary>
00063     public string Location0C { get; set; }
00064 /// <summary>
00065 /// The memory at the location offset + 0D
00066 /// </summary>
00067     public string Location0D { get; set; }
00068 /// <summary>
00069 /// The memory at the location offset + 0E
00070 /// </summary>
00071     public string Location0E { get; set; }
00072 /// <summary>
00073 /// The memory at the location offset + 0F
00074 /// </summary>
00075     public string Location0F { get; set; }
00076     }
00077 }
```

7.27 Emulator/Model/OutputLog.cs File Reference

Classes

- class [Emulator.Model.OutputLog](#)

The *OutputLog Model*. Used by the outputlog grid to show a history of operations performed by the CPU

Namespaces

- namespace [Emulator](#)
- namespace [Emulator.Model](#)

7.28 OutputLog.cs

Go to the documentation of this file.

```

00001 using Hardware;
00002 using System;
00003
00004 namespace Emulator.Model
00005 {
00006     /// <summary>
00007     /// The OutputLog Model. Used by the outputlog grid to show a history of operations performed by the
    CPU
00008     /// </summary>
00009     [Serializable]
00010     public class OutputLog : Disassembly
00011     {
00012         public OutputLog(Disassembly disassembly)
00013         {
00014             DisassemblyOutput = disassembly.DisassemblyOutput;
00015             HighAddress = disassembly.HighAddress;
00016             LowAddress = disassembly.LowAddress;
00017             OpCodeString = disassembly.OpCodeString;
00018         }
00019
00020     /// <summary>
00021     /// The Program Counter Value
00022     /// </summary>
00023     public string ProgramCounter { get; set; }
00024     /// <summary>
00025     /// The Current Ope Code
00026     /// </summary>
00027     public string CurrentOpCode { get; set; }
00028     /// <summary>
00029     /// The X Register
00030     /// </summary>
00031     public string XRegister { get; set; }
00032     /// <summary>
00033     /// The Y Register
00034     /// </summary>
00035     public string YRegister { get; set; }
00036     /// <summary>
00037     /// The Accumulator
00038     /// </summary>
00039     public string Accumulator { get; set; }
00040     /// <summary>
00041     /// The Stack Pointer
00042     /// </summary>
00043     public string StackPointer { get; set; }
00044     /// <summary>
00045     /// The number of cycles executed since the last load or reset
00046     /// </summary>
00047     public int NumberOfCycles { get; set; }
00048     }
00049 }

```

7.29 Emulator/Model/RomFileModel.cs File Reference

Classes

- class [Emulator.Model.RomFileModel](#)
The *Model* used when Loading a Program.

Namespaces

- namespace [Emulator](#)
- namespace [Emulator.Model](#)

7.30 RomFileModel.cs

[Go to the documentation of this file.](#)

```
00001 namespace Emulator.Model
00002 {
00003     /// <summary>
00004     /// The Model used when Loading a Program.
00005     /// </summary>
00006     public class RomFileModel
00007     {
00008         /// <summary>
00009         /// The Program Converted into Hex.
00010         /// </summary>
00011         public byte[][] Rom { get; set; }
00012
00013         /// <summary>
00014         /// The path of the Program that was loaded.
00015         /// </summary>
00016         public byte RomBanks { get; set; }
00017
00018         /// <summary>
00019         /// The name of the Program that was loaded.
00020         /// </summary>
00021         public int RomBankSize { get; set; }
00022
00023         /// <summary>
00024         /// The name of the Program that was loaded.
00025         /// </summary>
00026         public string RomFileName { get; set; }
00027
00028         /// <summary>
00029         /// The path of the Program that was loaded.
00030         /// </summary>
00031         public string RomFilePath { get; set; }
00032     }
00033 }
```

7.31 Emulator/Model/SettingsModel.cs File Reference

Classes

- class [Emulator.Model.SettingsModel](#)
Model that contains the required information needed to save the current settings to disk

Namespaces

- namespace [Emulator](#)
- namespace [Emulator.Model](#)

7.32 SettingsModel.cs

[Go to the documentation of this file.](#)

```
00001 using System;
00002 using System.Xml.Serialization;
00003
00004 namespace Emulator.Model
00005 {
00006     /// <summary>
00007     /// Model that contains the required information needed to save the current settings to disk
00008     /// </summary>
00009     [Serializable]
00010     [XmlRootAttribute("SettingsFileModel", Namespace = "Emulator.Model", IsNullable = false)]
00011     public class SettingsModel
00012     {
00013         /// <summary>
00014         /// The version of the file that is being saved
00015         /// </summary>
00016         public byte SettingsVersionMajor { get; set; }
00017     }
```

```

00018 /// <summary>
00019 /// The version of the file that is being saved
00020 /// </summary>
00021     public byte SettingsVersionMinor { get; set; }
00022
00023 /// <summary>
00024 /// The version of the file that is being saved
00025 /// </summary>
00026     public byte SettingsVersionBuild { get; set; }
00027
00028 /// <summary>
00029 /// The version of the file that is being saved
00030 /// </summary>
00031     public byte SettingsVersionRevision { get; set; }
00032
00033 /// <summary>
00034 /// The PC port that is being saved
00035 /// </summary>
00036     public string ComPortName { get; set; }
00037 }
00038 }

```

7.33 Emulator/Model/StateFileModel.cs File Reference

Classes

- class [Emulator.Model.StateFileModel](#)
Model that contains the required information needed to save the current state of the processor to disk

Namespaces

- namespace [Emulator](#)
- namespace [Emulator.Model](#)

7.34 StateFileModel.cs

[Go to the documentation of this file.](#)

```

00001 using System;
00002 using System.Collections.Generic;
00003
00004 namespace Emulator.Model
00005 {
00006     /// <summary>
00007     /// Model that contains the required information needed to save the current state of the processor to
00008     /// disk
00009     /// </summary>
00010     [Serializable]
00011     public class StateFileModel
00012     {
00013         /// <summary>
00014         /// The Number of Cycles the Program has Ran so Far
00015         /// </summary>
00016         public int NumberOfCycles { get; set; }
00017
00018         /// <summary>
00019         /// The output of the program
00020         /// </summary>
00021         public IList<OutputLog> OutputLog { get; set; }
00022
00023         /// <summary>
00024         /// The Processor Object that is being saved
00025         /// </summary>
00026         public Hardware.W65C02 W65C02 { get; set; }
00027
00028         /// <summary>
00029         /// The first VIA Object that is being saved
00030         /// </summary>
00031         public Hardware.W65C22 W65C22 { get; set; }
00032
00033         /// <summary>
00034         /// The second VIA Object that is being saved

```



```

00034 /// </summary>
00035     public Hardware.W65C22 MM65SIB { get; set; }
00036
00037 /// <summary>
00038 /// The ACIA Object that is being saved
00039 /// </summary>
00040     public Hardware.W65C51 W65C51 { get; set; }
00041
00042 /// <summary>
00043 /// The Shared ROM Object that is being saved
00044 /// </summary>
00045     public Hardware.AT28CXX AT28C010 { get; set; }
00046
00047 /// <summary>
00048 /// The Banked ROM Object that is being saved
00049 /// </summary>
00050     public Hardware.AT28CXX AT28C64 { get; set; }
00051     }
00052 }

```

7.35 Emulator/MultiThreadedCollection.cs File Reference

Classes

- class [Emulator.MultiThreadedObservableCollection< T >](#)

A MultiThreaedObservableCollection. This allows multiple threads to access the same observable collection in a safe manner.

Namespaces

- namespace [Emulator](#)

7.36 MultiThreadedCollection.cs

[Go to the documentation of this file.](#)

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Collections.ObjectModel;
00004 using System.Collections.Specialized;
00005 using System.Windows.Threading;
00006
00007 namespace Emulator
00008 {
00009     /// <summary>
00010     /// A MultiThreaedObservableCollection.
00011     /// This allows multiple threads to access the same observable collection in a safe manner.
00012     /// </summary>
00013     /// <typeparam name="T"></typeparam>
00014     public class MultiThreadedObservableCollection<T> : ObservableCollection<T>
00015     {
00016         /// <summary>
00017         /// Instantiates a new instance of the MultiThreadedObservableCollection
00018         /// </summary>
00019         public MultiThreadedObservableCollection()
00020         {
00021         }
00022     }
00023
00024     /// <summary>
00025     /// Instantiates a new instance of the MultiThreadedObservableCollection
00026     /// </summary>
00027     /// <param name="collection">The initial collection to be loaded</param>
00028     public MultiThreadedObservableCollection(IEnumerable<T> collection)
00029         : base(collection)
00030     {
00031     }
00032 }
00033
00034 /// <summary>
00035 /// Instantiates a new instance of the MultiThreadedObservableCollection
00036 /// </summary>

```

```

00037 /// <param name="list">The initial list to be loaded</param>
00038     public MultiThreadedObservableCollection(List<T> list)
00039     : base(list)
00040     {
00041     }
00042 }
00043
00044 /// <summary>
00045 /// The NotifyCollectionChangedEventHandler, Sends a notification anytime the collection has been
    modified.
00046 /// </summary>
00047     public override event NotifyCollectionChangedEventHandler CollectionChanged;
00048
00049
00050 /// <summary>
00051 /// The NotifyCollectionChangedEventHandler, Notifies the listeners in a thread safe manner
00052 /// </summary>
00053     protected override void OnCollectionChanged(NotifyCollectionChangedEventArgs e)
00054     {
00055         var collectionChanged = CollectionChanged;
00056         if (collectionChanged != null)
00057             foreach (NotifyCollectionChangedEventHandler nh in
    collectionChanged.GetInvocationList())
00058             {
00059                 var dispObj = nh.Target as DispatcherObject;
00060                 if (dispObj != null)
00061                 {
00062                     var dispatcher = dispObj.Dispatcher;
00063                     if (dispatcher != null && !dispatcher.CheckAccess())
00064                     {
00065                         var nh1 = nh;
00066                         dispatcher.BeginInvoke(
00067                             (Action)() => nh1.Invoke(this,
00068                                 new
    NotifyCollectionChangedEventArgs(NotifyCollectionChangedAction.Reset))),
00069                             DispatcherPriority.DataBind);
00070                         continue;
00071                     }
00072                 }
00073                 nh.Invoke(this, e);
00074             }
00075     }
00076 }
00077 }

```

7.37 Emulator/obj/x86/Debug/.NETFramework,Version=v4.8.AssemblyAttributes.cs File Reference

7.38 .NETFramework,Version=v4.8.AssemblyAttributes.cs

[Go to the documentation of this file.](#)

```

00001 // <autogenerated />
00002 using System;
00003 using System.Reflection;
00004 [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETFramework,Version=v4.8",
    FrameworkDisplayName = ".NET Framework 4.8")]

```

7.39 Emulator/obj/x86/Publish/.NETFramework,Version=v4.8.AssemblyAttributes.cs File Reference

7.40 .NETFramework,Version=v4.8.AssemblyAttributes.cs

[Go to the documentation of this file.](#)

```

00001 // <autogenerated />
00002 using System;
00003 using System.Reflection;
00004 [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETFramework,Version=v4.8",
    FrameworkDisplayName = ".NET Framework 4.8")]

```

7.41 Emulator/obj/x86/Release/.NETFramework,Version=v4.8.AssemblyAttributes.cs File Reference

7.42 .NETFramework,Version=v4.8.AssemblyAttributes.cs

[Go to the documentation of this file.](#)

```
00001 // <autogenerated />
00002 using System;
00003 using System.Reflection;
00004 [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETFramework,Version=v4.8",
    FrameworkDisplayName = ".NET Framework 4.8")]
```

7.43 Hardware/obj/Debug/.NETFramework,Version=v4.8.AssemblyAttributes.cs File Reference

7.44 .NETFramework,Version=v4.8.AssemblyAttributes.cs

[Go to the documentation of this file.](#)

```
00001 // <autogenerated />
00002 using System;
00003 using System.Reflection;
00004 [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETFramework,Version=v4.8",
    FrameworkDisplayName = ".NET Framework 4.8")]
```

7.45 Hardware/obj/Publish/.NETFramework,Version=v4.8.AssemblyAttributes.cs File Reference

7.46 .NETFramework,Version=v4.8.AssemblyAttributes.cs

[Go to the documentation of this file.](#)

```
00001 // <autogenerated />
00002 using System;
00003 using System.Reflection;
00004 [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETFramework,Version=v4.8",
    FrameworkDisplayName = ".NET Framework 4.8")]
```

7.47 Hardware/obj/Release/.NETFramework,Version=v4.8.AssemblyAttributes.cs File Reference

7.48 .NETFramework,Version=v4.8.AssemblyAttributes.cs

[Go to the documentation of this file.](#)

```
00001 // <autogenerated />
00002 using System;
00003 using System.Reflection;
00004 [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETFramework,Version=v4.8",
    FrameworkDisplayName = ".NET Framework 4.8")]
```

7.49 Emulator/obj/x86/Debug/App.g.cs File Reference

Classes

- class [XamlGeneratedNamespace.GeneratedApplication](#)
GeneratedApplication

Namespaces

- namespace [XamlGeneratedNamespace](#)

7.50 App.g.cs

Go to the documentation of this file.

```

00001 #pragma checksum "..\..\..\App.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
      "3C3B83350F313F767CDD9CA458D577D426BB4EF0F6F94CE9866749BCB08F1D0F"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using Emulator.ViewModel;
00013 using System;
00014 using System.Diagnostics;
00015 using System.Windows;
00016 using System.Windows.Automation;
00017 using System.Windows.Controls;
00018 using System.Windows.Controls.Primitives;
00019 using System.Windows.Data;
00020 using System.Windows.Documents;
00021 using System.Windows.Ink;
00022 using System.Windows.Input;
00023 using System.Windows.Markup;
00024 using System.Windows.Media;
00025 using System.Windows.Media.Animation;
00026 using System.Windows.Media.Effects;
00027 using System.Windows.Media.Imaging;
00028 using System.Windows.Media.Media3D;
00029 using System.Windows.Media.TextFormatting;
00030 using System.Windows.Navigation;
00031 using System.Windows.Shapes;
00032 using System.Windows.Shell;
00033
00034
00035 namespace XamlGeneratedNamespace {
00036
00037
00038     /// <summary>
00039     /// GeneratedApplication
00040     /// </summary>
00041     public partial class GeneratedApplication : System.Windows.Application {
00042
00043         private bool _contentLoaded;
00044
00045         /// <summary>
00046         /// InitializeComponent
00047         /// </summary>
00048         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00049         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00050         public void InitializeComponent() {
00051             if (_contentLoaded) {
00052                 return;
00053             }
00054             _contentLoaded = true;
00055
00056             #line 2 "..\..\..\App.xaml"
00057             this.StartupUri = new System.Uri("MainWindow.xaml", System.UriKind.Relative);
00058
00059             #line default
00060             #line hidden
00061             System.Uri resourceLocater = new System.Uri("/Emulator;component/app.xaml",
                System.UriKind.Relative);
00062
00063             #line 1 "..\..\..\App.xaml"
00064             System.Windows.Application.LoadComponent(this, resourceLocater);
00065
00066             #line default
00067             #line hidden
00068             }
00069
00070         /// <summary>
00071         /// Application Entry Point.
00072         /// </summary>

```

```

00073         [System.STAThreadAttribute()]
00074         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00075         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00076         public static void Main() {
00077             SplashScreen splashScreen = new SplashScreen("splashscreen.png");
00078             splashScreen.Show(true);
00079             XamlGeneratedNamespace.GeneratedApplication app = new
XamlGeneratedNamespace.GeneratedApplication();
00080             app.InitializeComponent();
00081             app.Run();
00082         }
00083     }
00084 }
00085

```

7.51 Emulator/obj/x86/Publish/App.g.cs File Reference

Classes

- class [XamlGeneratedNamespace.GeneratedApplication](#)
GeneratedApplication

Namespaces

- namespace [XamlGeneratedNamespace](#)

7.52 App.g.cs

Go to the documentation of this file.

```

00001 #pragma checksum "..\..\..\App.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
"3C3B83350F313F767CDD9CA458D577D426BB4EF0F6F94CE9866749BCB08F1D0F"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using Emulator.ViewModel;
00013 using System;
00014 using System.Diagnostics;
00015 using System.Windows;
00016 using System.Windows.Automation;
00017 using System.Windows.Controls;
00018 using System.Windows.Controls.Primitives;
00019 using System.Windows.Data;
00020 using System.Windows.Documents;
00021 using System.Windows.Ink;
00022 using System.Windows.Input;
00023 using System.Windows.Markup;
00024 using System.Windows.Media;
00025 using System.Windows.Media.Animation;
00026 using System.Windows.Media.Effects;
00027 using System.Windows.Media.Imaging;
00028 using System.Windows.Media.Media3D;
00029 using System.Windows.Media.TextFormatting;
00030 using System.Windows.Navigation;
00031 using System.Windows.Shapes;
00032 using System.Windows.Shell;
00033
00034
00035 namespace XamlGeneratedNamespace {
00036
00037     /// <summary>
00038     ///     GeneratedApplication
00039     /// </summary>
00040     public partial class GeneratedApplication : System.Windows.Application {
00041
00042

```

```

00043         private bool _contentLoaded;
00044
00045         /// <summary>
00046         /// InitializeComponent
00047         /// </summary>
00048         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00049         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00050         public void InitializeComponent() {
00051             if (_contentLoaded) {
00052                 return;
00053             }
00054             _contentLoaded = true;
00055
00056             #line 2 "..\..\..\App.xaml"
00057             this.StartupUri = new System.Uri("MainWindow.xaml", System.UriKind.Relative);
00058
00059             #line default
00060             #line hidden
00061             System.Uri resourceLocator = new System.Uri("/Emulator;component/app.xaml",
00062                 System.UriKind.Relative);
00063             #line 1 "..\..\..\App.xaml"
00064             System.Windows.Application.LoadComponent(this, resourceLocator);
00065
00066             #line default
00067             #line hidden
00068             }
00069
00070         /// <summary>
00071         /// Application Entry Point.
00072         /// </summary>
00073         [System.STAThreadAttribute()]
00074         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00075         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00076         public static void Main() {
00077             SplashScreen splashScreen = new SplashScreen("splashscreen.png");
00078             splashScreen.Show(true);
00079             XamlGeneratedNamespace.GeneratedApplication app = new
00080                 XamlGeneratedNamespace.GeneratedApplication();
00081             app.InitializeComponent();
00082             app.Run();
00083         }
00084     }
00085

```

7.53 Emulator/obj/x86/Release/App.g.cs File Reference

Classes

- class [XamlGeneratedNamespace.GeneratedApplication](#)
GeneratedApplication

Namespaces

- namespace [XamlGeneratedNamespace](#)

7.54 App.g.cs

Go to the documentation of this file.

```

00001 #pragma checksum "..\..\..\App.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00002 "3C3B83350F313F767CDD9CA458D577D426BB4EF0F6F94CE9866749BCB08F1D0F"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011

```

```

00012 using Emulator.ViewModel;
00013 using System;
00014 using System.Diagnostics;
00015 using System.Windows;
00016 using System.Windows.Automation;
00017 using System.Windows.Controls;
00018 using System.Windows.Controls.Primitives;
00019 using System.Windows.Data;
00020 using System.Windows.Documents;
00021 using System.Windows.Ink;
00022 using System.Windows.Input;
00023 using System.Windows.Markup;
00024 using System.Windows.Media;
00025 using System.Windows.Media.Animation;
00026 using System.Windows.Media.Effects;
00027 using System.Windows.Media.Imaging;
00028 using System.Windows.Media.Media3D;
00029 using System.Windows.Media.TextFormatting;
00030 using System.Windows.Navigation;
00031 using System.Windows.Shapes;
00032 using System.Windows.Shell;
00033
00034
00035 namespace XamlGeneratedNamespace {
00036
00037
00038     /// <summary>
00039     /// GeneratedApplication
00040     /// </summary>
00041     public partial class GeneratedApplication : System.Windows.Application {
00042
00043         private bool _contentLoaded;
00044
00045         /// <summary>
00046         /// InitializeComponent
00047         /// </summary>
00048         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00049         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00050         public void InitializeComponent() {
00051             if (_contentLoaded) {
00052                 return;
00053             }
00054             _contentLoaded = true;
00055
00056             #line 2 "..\..\..\App.xaml"
00057             this.StartupUri = new System.Uri("MainWindow.xaml", System.UriKind.Relative);
00058
00059             #line default
00060             #line hidden
00061             System.Uri resourceLocator = new System.Uri("/Emulator;component/app.xaml",
System.UriKind.Relative);
00062
00063             #line 1 "..\..\..\App.xaml"
00064             System.Windows.Application.LoadComponent(this, resourceLocator);
00065
00066             #line default
00067             #line hidden
00068             }
00069
00070         /// <summary>
00071         /// Application Entry Point.
00072         /// </summary>
00073         [System.STAThreadAttribute()]
00074         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00075         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00076         public static void Main() {
00077             SplashScreen splashScreen = new SplashScreen("splashscreen.png");
00078             splashScreen.Show(true);
00079             XamlGeneratedNamespace.GeneratedApplication app = new
XamlGeneratedNamespace.GeneratedApplication();
00080             app.InitializeComponent();
00081             app.Run();
00082         }
00083     }
00084 }
00085

```

7.55 Emulator/obj/x86/Debug/App.g.i.cs File Reference

Classes

- class [XamlGeneratedNamespace.GeneratedApplication](#)
GeneratedApplication

Namespaces

- namespace [XamlGeneratedNamespace](#)

7.56 App.g.i.cs

Go to the documentation of this file.

```

00001 #pragma checksum "..\..\..\App.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
      "3C3B83350F313F767CDD9CA458D577D426BB4EF0F6F94CE9866749BCB08F1D0F"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using Emulator.ViewModel;
00013 using System;
00014 using System.Diagnostics;
00015 using System.Windows;
00016 using System.Windows.Automation;
00017 using System.Windows.Controls;
00018 using System.Windows.Controls.Primitives;
00019 using System.Windows.Data;
00020 using System.Windows.Documents;
00021 using System.Windows.Ink;
00022 using System.Windows.Input;
00023 using System.Windows.Markup;
00024 using System.Windows.Media;
00025 using System.Windows.Media.Animation;
00026 using System.Windows.Media.Effects;
00027 using System.Windows.Media.Imaging;
00028 using System.Windows.Media.Media3D;
00029 using System.Windows.Media.TextFormatting;
00030 using System.Windows.Navigation;
00031 using System.Windows.Shapes;
00032 using System.Windows.Shell;
00033
00034
00035 namespace XamlGeneratedNamespace {
00036
00037
00038     /// <summary>
00039     /// GeneratedApplication
00040     /// </summary>
00041     public partial class GeneratedApplication : System.Windows.Application {
00042
00043         private bool _contentLoaded;
00044
00045         /// <summary>
00046         /// InitializeComponent
00047         /// </summary>
00048         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00049         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00050         public void InitializeComponent() {
00051             if (_contentLoaded) {
00052                 return;
00053             }
00054             _contentLoaded = true;
00055
00056             #line 2 "..\..\..\App.xaml"
00057             this.StartupUri = new System.Uri("MainWindow.xaml", System.UriKind.Relative);
00058
00059             #line default
00060             #line hidden
00061             System.Uri resourceLocater = new System.Uri("/Emulator;component/app.xaml",
                System.UriKind.Relative);
00062
00063             #line 1 "..\..\..\App.xaml"
00064             System.Windows.Application.LoadComponent(this, resourceLocater);
00065
00066             #line default
00067             #line hidden
00068             }
00069
00070         /// <summary>
00071         /// Application Entry Point.
00072         /// </summary>

```



```

00073         [System.STAThreadAttribute()]
00074         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00075         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00076         public static void Main() {
00077             SplashScreen splashScreen = new SplashScreen("splashscreen.png");
00078             splashScreen.Show(true);
00079             XamlGeneratedNamespace.GeneratedApplication app = new
XamlGeneratedNamespace.GeneratedApplication();
00080             app.InitializeComponent();
00081             app.Run();
00082         }
00083     }
00084 }
00085

```

7.57 Emulator/obj/x86/Publish/App.g.i.cs File Reference

Classes

- class [XamlGeneratedNamespace.GeneratedApplication](#)
GeneratedApplication

Namespaces

- namespace [XamlGeneratedNamespace](#)

7.58 App.g.i.cs

Go to the documentation of this file.

```

00001 #pragma checksum "..\..\..\App.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
"3C3B83350F313F767CDD9CA458D577D426BB4EF0F6F94CE9866749BCB08F1D0F"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using Emulator.ViewModel;
00013 using System;
00014 using System.Diagnostics;
00015 using System.Windows;
00016 using System.Windows.Automation;
00017 using System.Windows.Controls;
00018 using System.Windows.Controls.Primitives;
00019 using System.Windows.Data;
00020 using System.Windows.Documents;
00021 using System.Windows.Ink;
00022 using System.Windows.Input;
00023 using System.Windows.Markup;
00024 using System.Windows.Media;
00025 using System.Windows.Media.Animation;
00026 using System.Windows.Media.Effects;
00027 using System.Windows.Media.Imaging;
00028 using System.Windows.Media.Media3D;
00029 using System.Windows.Media.TextFormatting;
00030 using System.Windows.Navigation;
00031 using System.Windows.Shapes;
00032 using System.Windows.Shell;
00033
00034
00035 namespace XamlGeneratedNamespace {
00036
00037     /// <summary>
00038     ///     GeneratedApplication
00039     /// </summary>
00040     public partial class GeneratedApplication : System.Windows.Application {
00041
00042

```

```

00043         private bool _contentLoaded;
00044
00045         /// <summary>
00046         /// InitializeComponent
00047         /// </summary>
00048         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00049         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00050         public void InitializeComponent() {
00051             if (_contentLoaded) {
00052                 return;
00053             }
00054             _contentLoaded = true;
00055
00056             #line 2 "..\..\..\App.xaml"
00057             this.StartupUri = new System.Uri("MainWindow.xaml", System.UriKind.Relative);
00058
00059             #line default
00060             #line hidden
00061             System.Uri resourceLocator = new System.Uri("/Emulator;component/app.xaml",
00062                 System.UriKind.Relative);
00063             #line 1 "..\..\..\App.xaml"
00064             System.Windows.Application.LoadComponent(this, resourceLocator);
00065
00066             #line default
00067             #line hidden
00068             }
00069
00070         /// <summary>
00071         /// Application Entry Point.
00072         /// </summary>
00073         [System.STAThreadAttribute()]
00074         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00075         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00076         public static void Main() {
00077             SplashScreen splashScreen = new SplashScreen("splashscreen.png");
00078             splashScreen.Show(true);
00079             XamlGeneratedNamespace.GeneratedApplication app = new
00080                 XamlGeneratedNamespace.GeneratedApplication();
00081             app.InitializeComponent();
00082             app.Run();
00083         }
00084     }
00085

```

7.59 Emulator/obj/x86/Release/App.g.i.cs File Reference

Classes

- class [XamlGeneratedNamespace.GeneratedApplication](#)
GeneratedApplication

Namespaces

- namespace [XamlGeneratedNamespace](#)

7.60 App.g.i.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\App.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00002 "3C3B83350F313F767CDD9CA458D577D426BB4EF0F6F94CE9866749BCB08F1D0F"
00003 //-----
00004 // <auto-generated>
00005 //     This code was generated by a tool.
00006 //     Runtime Version:4.0.30319.42000
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011

```

```

00012 using Emulator.ViewModel;
00013 using System;
00014 using System.Diagnostics;
00015 using System.Windows;
00016 using System.Windows.Automation;
00017 using System.Windows.Controls;
00018 using System.Windows.Controls.Primitives;
00019 using System.Windows.Data;
00020 using System.Windows.Documents;
00021 using System.Windows.Ink;
00022 using System.Windows.Input;
00023 using System.Windows.Markup;
00024 using System.Windows.Media;
00025 using System.Windows.Media.Animation;
00026 using System.Windows.Media.Effects;
00027 using System.Windows.Media.Imaging;
00028 using System.Windows.Media.Media3D;
00029 using System.Windows.Media.TextFormatting;
00030 using System.Windows.Navigation;
00031 using System.Windows.Shapes;
00032 using System.Windows.Shell;
00033
00034
00035 namespace XamlGeneratedNamespace {
00036
00037
00038     /// <summary>
00039     /// GeneratedApplication
00040     /// </summary>
00041     public partial class GeneratedApplication : System.Windows.Application {
00042
00043         private bool _contentLoaded;
00044
00045         /// <summary>
00046         /// InitializeComponent
00047         /// </summary>
00048         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00049         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00050         public void InitializeComponent() {
00051             if (_contentLoaded) {
00052                 return;
00053             }
00054             _contentLoaded = true;
00055
00056             #line 2 "..\..\..\App.xaml"
00057             this.StartupUri = new System.Uri("MainWindow.xaml", System.UriKind.Relative);
00058
00059             #line default
00060             #line hidden
00061             System.Uri resourceLocater = new System.Uri("/Emulator;component/app.xaml",
System.UriKind.Relative);
00062
00063             #line 1 "..\..\..\App.xaml"
00064             System.Windows.Application.LoadComponent(this, resourceLocater);
00065
00066             #line default
00067             #line hidden
00068         }
00069
00070         /// <summary>
00071         /// Application Entry Point.
00072         /// </summary>
00073         [System.STAThreadAttribute()]
00074         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00075         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00076         public static void Main() {
00077             SplashScreen splashScreen = new SplashScreen("splashscreen.png");
00078             splashScreen.Show(true);
00079             XamlGeneratedNamespace.GeneratedApplication app = new
XamlGeneratedNamespace.GeneratedApplication();
00080             app.InitializeComponent();
00081             app.Run();
00082         }
00083     }
00084 }
00085

```

7.61 Emulator/obj/x86/Debug/Emulator_Content.g.cs File Reference

7.62 Emulator_Content.g.cs

[Go to the documentation of this file.](#)

```

00001 //-----
00002 // <auto-generated>
00003 //      This code was generated by a tool.
00004 //      Runtime Version:4.0.30319.42000
00005 //
00006 //      Changes to this file may cause incorrect behavior and will be lost if
00007 //      the code is regenerated.
00008 // </auto-generated>
00009 //-----
00010
00011 [assembly: System.Windows.Resources.AssemblyAssociatedContentFileAttribute("nlog.config")]
00012
00013

```

7.63 Emulator/obj/x86/Publish/Emulator_Content.g.cs File Reference

7.64 Emulator_Content.g.cs

[Go to the documentation of this file.](#)

```

00001 //-----
00002 // <auto-generated>
00003 //      This code was generated by a tool.
00004 //      Runtime Version:4.0.30319.42000
00005 //
00006 //      Changes to this file may cause incorrect behavior and will be lost if
00007 //      the code is regenerated.
00008 // </auto-generated>
00009 //-----
00010
00011 [assembly: System.Windows.Resources.AssemblyAssociatedContentFileAttribute("nlog.config")]
00012
00013

```

7.65 Emulator/obj/x86/Release/Emulator_Content.g.cs File Reference

7.66 Emulator_Content.g.cs

[Go to the documentation of this file.](#)

```

00001 //-----
00002 // <auto-generated>
00003 //      This code was generated by a tool.
00004 //      Runtime Version:4.0.30319.42000
00005 //
00006 //      Changes to this file may cause incorrect behavior and will be lost if
00007 //      the code is regenerated.
00008 // </auto-generated>
00009 //-----
00010
00011 [assembly: System.Windows.Resources.AssemblyAssociatedContentFileAttribute("nlog.config")]
00012
00013

```

7.67 Emulator/obj/x86/Debug/Emulator_Content.g.i.cs File Reference

7.68 Emulator_Content.g.i.cs

[Go to the documentation of this file.](#)

```

00001 //-----
00002 // <auto-generated>
00003 //      This code was generated by a tool.
00004 //      Runtime Version:4.0.30319.42000
00005 //
00006 //      Changes to this file may cause incorrect behavior and will be lost if
00007 //      the code is regenerated.
00008 // </auto-generated>
00009 //-----
00010
00011 [assembly: System.Windows.Resources.AssemblyAssociatedContentFileAttribute("nlog.config")]
00012
00013

```

7.69 Emulator/obj/x86/Publish/Emulator_Content.g.i.cs File Reference

7.70 Emulator_Content.g.i.cs

[Go to the documentation of this file.](#)

```
00001 //-----
00002 // <auto-generated>
00003 //     This code was generated by a tool.
00004 //     Runtime Version:4.0.30319.42000
00005 //
00006 //     Changes to this file may cause incorrect behavior and will be lost if
00007 //     the code is regenerated.
00008 // </auto-generated>
00009 //-----
00010
00011 [assembly: System.Windows.Resources.AssemblyAssociatedContentFileAttribute("nlog.config")]
00012
00013
```

7.71 Emulator/obj/x86/Release/Emulator_Content.g.i.cs File Reference

7.72 Emulator_Content.g.i.cs

[Go to the documentation of this file.](#)

```
00001 //-----
00002 // <auto-generated>
00003 //     This code was generated by a tool.
00004 //     Runtime Version:4.0.30319.42000
00005 //
00006 //     Changes to this file may cause incorrect behavior and will be lost if
00007 //     the code is regenerated.
00008 // </auto-generated>
00009 //-----
00010
00011 [assembly: System.Windows.Resources.AssemblyAssociatedContentFileAttribute("nlog.config")]
00012
00013
```

7.73 Emulator/obj/x86/Debug/GeneratedInternalTypeHelper.g.cs File Reference

7.74 GeneratedInternalTypeHelper.g.cs

[Go to the documentation of this file.](#)

```
00001
00002
```

7.75 Emulator/obj/x86/Publish/GeneratedInternalTypeHelper.g.cs File Reference

7.76 GeneratedInternalTypeHelper.g.cs

[Go to the documentation of this file.](#)

```
00001
00002
```

7.77 Emulator/obj/x86/Release/GeneratedInternalTypeHelper.g.cs File Reference

7.78 GeneratedInternalTypeHelper.g.cs

[Go to the documentation of this file.](#)

```
00001
00002
```

7.79 Emulator/obj/x86/Debug/GeneratedInternalTypeHelper.g.i.cs File Reference

Classes

- class [XamlGeneratedNamespace.GeneratedInternalTypeHelper](#)
GeneratedInternalTypeHelper

Namespaces

- namespace [XamlGeneratedNamespace](#)

7.80 GeneratedInternalTypeHelper.g.i.cs

[Go to the documentation of this file.](#)

```

00001 //-----
00002 // <auto-generated>
00003 //     This code was generated by a tool.
00004 //     Runtime Version:4.0.30319.42000
00005 //
00006 //     Changes to this file may cause incorrect behavior and will be lost if
00007 //     the code is regenerated.
00008 // </auto-generated>
00009 //-----
00010
00011 namespace XamlGeneratedNamespace {
00012
00013
00014     /// <summary>
00015     /// GeneratedInternalTypeHelper
00016     /// </summary>
00017     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00018     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00019     [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00020     public sealed class GeneratedInternalTypeHelper : System.Windows.Markup.InternalTypeHelper {
00021
00022     /// <summary>
00023     /// CreateInstance
00024     /// </summary>
00025     protected override object CreateInstance(System.Type type, System.Globalization.CultureInfo
culture) {
00026         return System.Activator.CreateInstance(type, ((System.Reflection.BindingFlags.Public |
System.Reflection.BindingFlags.NonPublic)
00027             | (System.Reflection.BindingFlags.Instance |
System.Reflection.BindingFlags.CreateInstance)), null, null, culture);
00028     }
00029
00030     /// <summary>
00031     /// GetPropertyValue
00032     /// </summary>
00033     protected override object GetPropertyValue(System.Reflection.PropertyInfo propertyInfo, object
target, System.Globalization.CultureInfo culture) {
00034         return propertyInfo.GetValue(target, System.Reflection.BindingFlags.Default, null, null,
culture);
00035     }
00036
00037     /// <summary>
00038     /// SetPropertyValue
00039     /// </summary>
00040     protected override void SetPropertyValue(System.Reflection.PropertyInfo propertyInfo, object
target, object value, System.Globalization.CultureInfo culture) {
00041         propertyInfo.SetValue(target, value, System.Reflection.BindingFlags.Default, null, null,
culture);
00042     }
00043
00044     /// <summary>
00045     /// CreateDelegate
00046     /// </summary>
00047     protected override System.Delegate CreateDelegate(System.Type delegateType, object target,
string handler) {
00048         return ((System.Delegate) (target.GetType().InvokeMember("_CreateDelegate",
(System.Reflection.BindingFlags.InvokeMethod
00049             | (System.Reflection.BindingFlags.NonPublic |
System.Reflection.BindingFlags.Instance)), null, target, new object[] {
00050             delegateType,
00051             handler}, null)));

```

```

00052         }
00053
00054     /// <summary>
00055     /// AddEventHandler
00056     /// </summary>
00057     protected override void AddEventHandler(System.Reflection.EventInfo eventInfo, object target,
    System.Delegate handler) {
00058         eventInfo.AddEventHandler(target, handler);
00059     }
00060 }
00061 }
00062

```

7.81 Emulator/obj/x86/Publish/GeneratedInternalTypeHelper.g.i.cs File Reference

Classes

- class [XamlGeneratedNamespace.GeneratedInternalTypeHelper](#)
GeneratedInternalTypeHelper

Namespaces

- namespace [XamlGeneratedNamespace](#)

7.82 GeneratedInternalTypeHelper.g.i.cs

[Go to the documentation of this file.](#)

```

00001 //-----
00002 // <auto-generated>
00003 //     This code was generated by a tool.
00004 //     Runtime Version:4.0.30319.42000
00005 //
00006 //     Changes to this file may cause incorrect behavior and will be lost if
00007 //     the code is regenerated.
00008 // </auto-generated>
00009 //-----
00010
00011 namespace XamlGeneratedNamespace {
00012
00013
00014     /// <summary>
00015     /// GeneratedInternalTypeHelper
00016     /// </summary>
00017     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00018     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00019     [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00020     public sealed class GeneratedInternalTypeHelper : System.Windows.Markup.InternalTypeHelper {
00021
00022     /// <summary>
00023     /// CreateInstance
00024     /// </summary>
00025     protected override object CreateInstance(System.Type type, System.Globalization.CultureInfo
    culture) {
00026         return System.Activator.CreateInstance(type, ((System.Reflection.BindingFlags.Public |
    System.Reflection.BindingFlags.NonPublic)
00027         | (System.Reflection.BindingFlags.Instance |
    System.Reflection.BindingFlags.CreateInstance)), null, null, culture);
00028     }
00029
00030     /// <summary>
00031     /// GetPropertyValue
00032     /// </summary>
00033     protected override object GetPropertyValue(System.Reflection.PropertyInfo propertyInfo, object
    target, System.Globalization.CultureInfo culture) {
00034         return propertyInfo.GetValue(target, System.Reflection.BindingFlags.Default, null, null,
    culture);
00035     }
00036
00037     /// <summary>
00038     /// SetPropertyValue
00039     /// </summary>

```

```

00040         protected override void SetPropertyValues(System.Reflection.PropertyInfo propertyInfo, object
target, object value, System.Globalization.CultureInfo culture) {
00041             propertyInfo.SetValue(target, value, System.Reflection.BindingFlags.Default, null, null,
culture);
00042         }
00043
00044         /// <summary>
00045         /// CreateDelegate
00046         /// </summary>
00047         protected override System.Delegate CreateDelegate(System.Type delegateType, object target,
string handler) {
00048             return ((System.Delegate) (target.GetType().InvokeMember("_CreateDelegate",
(System.Reflection.BindingFlags.InvokeMethod
| (System.Reflection.BindingFlags.NonPublic |
System.Reflection.BindingFlags.Instance)), null, target, new object[] {
00050                 delegateType,
00051                 handler}, null)));
00052         }
00053
00054         /// <summary>
00055         /// AddEventHandler
00056         /// </summary>
00057         protected override void AddEventHandler(System.Reflection.EventInfo eventInfo, object target,
System.Delegate handler) {
00058             eventInfo.AddEventHandler(target, handler);
00059         }
00060     }
00061 }
00062

```

7.83 Emulator/obj/x86/Release/GeneratedInternalTypeHelper.g.i.cs File Reference

Classes

- class [XamlGeneratedNamespace.GeneratedInternalTypeHelper](#)
GeneratedInternalTypeHelper

Namespaces

- namespace [XamlGeneratedNamespace](#)

7.84 GeneratedInternalTypeHelper.g.i.cs

[Go to the documentation of this file.](#)

```

00001 //-----
00002 // <auto-generated>
00003 //     This code was generated by a tool.
00004 //     Runtime Version:4.0.30319.42000
00005 //
00006 //     Changes to this file may cause incorrect behavior and will be lost if
00007 //     the code is regenerated.
00008 // </auto-generated>
00009 //-----
00010
00011 namespace XamlGeneratedNamespace {
00012
00013     /// <summary>
00014     /// GeneratedInternalTypeHelper
00015     /// </summary>
00016     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00017     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00018     [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00019     public sealed class GeneratedInternalTypeHelper : System.Windows.Markup.InternalTypeHelper {
00020
00021
00022     /// <summary>
00023     /// CreateInstance
00024     /// </summary>
00025     protected override object CreateInstance(System.Type type, System.Globalization.CultureInfo
culture) {
00026         return System.Activator.CreateInstance(type, ((System.Reflection.BindingFlags.Public |
System.Reflection.BindingFlags.NonPublic)

```



```

00027         | (System.Reflection.BindingFlags.Instance |
System.Reflection.BindingFlags.CreateInstance)), null, null, culture);
00028     }
00029
00030     /// <summary>
00031     /// GetPropertyValue
00032     /// </summary>
00033     protected override object GetPropertyValue(System.Reflection.PropertyInfo propertyInfo, object
target, System.Globalization.CultureInfo culture) {
00034         return propertyInfo.GetValue(target, System.Reflection.BindingFlags.Default, null, null,
culture);
00035     }
00036
00037     /// <summary>
00038     /// SetPropertyValue
00039     /// </summary>
00040     protected override void SetPropertyValue(System.Reflection.PropertyInfo propertyInfo, object
target, object value, System.Globalization.CultureInfo culture) {
00041         propertyInfo.SetValue(target, value, System.Reflection.BindingFlags.Default, null, null,
culture);
00042     }
00043
00044     /// <summary>
00045     /// CreateDelegate
00046     /// </summary>
00047     protected override System.Delegate CreateDelegate(System.Type delegateType, object target,
string handler) {
00048         return ((System.Delegate) (target.GetType().InvokeMember("_CreateDelegate",
(System.Reflection.BindingFlags.InvokeMethod
| (System.Reflection.BindingFlags.NonPublic |
System.Reflection.BindingFlags.Instance)), null, target, new object[] {
00050             delegateType,
00051             handler}, null)));
00052     }
00053
00054     /// <summary>
00055     /// AddEventHandler
00056     /// </summary>
00057     protected override void AddEventHandler(System.Reflection.EventInfo eventInfo, object target,
System.Delegate handler) {
00058         eventInfo.AddEventHandler(target, handler);
00059     }
00060 }
00061 }
00062

```

7.85 Emulator/obj/x86/Debug/MainWindow.g.cs File Reference

Classes

- class [Emulator.MainWindow](#)
Interaction logic for MainWindow.xaml

Namespaces

- namespace [Emulator](#)

7.86 MainWindow.g.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\MainWindows\MainWindow.g.cs" "{8829d00f-11b8-4213-878b-770e8597ac16}"
"1FB39AF98423D8DD6333B173E814398E2016BACFE25491D8BD824F3F8A79E0A5"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011

```

```
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037     /// <summary>
00038     /// MainWindow
00039     /// </summary>
00040     public partial class MainWindow : System.Windows.Window,
        System.Windows.Markup.IComponentConnector {
00041
00042
00043         #line 2 "..\..\..\MainWindow.xaml"
00044         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
        "CA1823:AvoidUnusedPrivateFields")]
00045         internal Emulator.MainWindow EmulatorWindow;
00046
00047         #line default
00048         #line hidden
00049
00050
00051         #line 92 "..\..\..\MainWindow.xaml"
00052         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
        "CA1823:AvoidUnusedPrivateFields")]
00053         internal System.Windows.Controls.DataGrid OutputLog;
00054
00055         #line default
00056         #line hidden
00057
00058
00059         #line 109 "..\..\..\MainWindow.xaml"
00060         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
        "CA1823:AvoidUnusedPrivateFields")]
00061         internal System.Windows.Controls.Button Run;
00062
00063         #line default
00064         #line hidden
00065
00066
00067         #line 110 "..\..\..\MainWindow.xaml"
00068         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
        "CA1823:AvoidUnusedPrivateFields")]
00069         internal System.Windows.Controls.Button Step;
00070
00071         #line default
00072         #line hidden
00073
00074
00075         #line 111 "..\..\..\MainWindow.xaml"
00076         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
        "CA1823:AvoidUnusedPrivateFields")]
00077         internal System.Windows.Controls.Button Reset;
00078
00079         #line default
00080         #line hidden
00081
00082
00083         #line 113 "..\..\..\MainWindow.xaml"
00084         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
        "CA1823:AvoidUnusedPrivateFields")]
00085         internal System.Windows.Controls.TextBlock RomFileNameText;
00086
00087         #line default
00088         #line hidden
00089
00090
00091         #line 114 "..\..\..\MainWindow.xaml"
```

```
00092         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00093         "CA1823:AvoidUnusedPrivateFields")]
00094         internal System.Windows.Controls.TextBlock ComPortNameText;
00095 #line default
00096 #line hidden
00097
00098
00099 #line 115 "..\..\..\MainWindow.xaml"
00100         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00101         "CA1823:AvoidUnusedPrivateFields")]
00102         internal System.Windows.Controls.DataGrid Breakpoints;
00103 #line default
00104 #line hidden
00105
00106
00107 #line 140 "..\..\..\MainWindow.xaml"
00108         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00109         "CA1823:AvoidUnusedPrivateFields")]
00110         internal System.Windows.Controls.TextBox YRegister;
00111 #line default
00112 #line hidden
00113
00114
00115 #line 141 "..\..\..\MainWindow.xaml"
00116         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00117         "CA1823:AvoidUnusedPrivateFields")]
00118         internal System.Windows.Controls.TextBox XRegister;
00119 #line default
00120 #line hidden
00121
00122
00123 #line 142 "..\..\..\MainWindow.xaml"
00124         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00125         "CA1823:AvoidUnusedPrivateFields")]
00126         internal System.Windows.Controls.TextBox Accumulator;
00127 #line default
00128 #line hidden
00129
00130
00131 #line 143 "..\..\..\MainWindow.xaml"
00132         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00133         "CA1823:AvoidUnusedPrivateFields")]
00134         internal System.Windows.Controls.TextBox StackPointer;
00135 #line default
00136 #line hidden
00137
00138
00139 #line 144 "..\..\..\MainWindow.xaml"
00140         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00141         "CA1823:AvoidUnusedPrivateFields")]
00142         internal System.Windows.Controls.TextBox ProgramCounter;
00143 #line default
00144 #line hidden
00145
00146
00147 #line 145 "..\..\..\MainWindow.xaml"
00148         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00149         "CA1823:AvoidUnusedPrivateFields")]
00150         internal System.Windows.Controls.TextBox Dissassembly;
00151 #line default
00152 #line hidden
00153
00154
00155 #line 146 "..\..\..\MainWindow.xaml"
00156         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00157         "CA1823:AvoidUnusedPrivateFields")]
00158         internal System.Windows.Controls.TextBox CycleCount;
00159 #line default
00160 #line hidden
00161
00162
00163 #line 147 "..\..\..\MainWindow.xaml"
00164         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00165         "CA1823:AvoidUnusedPrivateFields")]
00166         internal System.Windows.Controls.TextBlock XRegisterText;
00167 #line default
00168 #line hidden
```

```
00169
00170
00171 #line 148 "..\..\..\MainWindow.xaml"
00172 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00173     "CA1823:AvoidUnusedPrivateFields")]
00174     internal System.Windows.Controls.TextBlock YRegisterText;
00175 #line default
00176 #line hidden
00177
00178
00179 #line 149 "..\..\..\MainWindow.xaml"
00180 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00181     "CA1823:AvoidUnusedPrivateFields")]
00182     internal System.Windows.Controls.TextBlock StackPointerRegisterText;
00183 #line default
00184 #line hidden
00185
00186
00187 #line 150 "..\..\..\MainWindow.xaml"
00188 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00189     "CA1823:AvoidUnusedPrivateFields")]
00190     internal System.Windows.Controls.TextBlock AText;
00191 #line default
00192 #line hidden
00193
00194
00195 #line 151 "..\..\..\MainWindow.xaml"
00196 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00197     "CA1823:AvoidUnusedPrivateFields")]
00198     internal System.Windows.Controls.TextBlock CurrentInstructionText;
00199 #line default
00200 #line hidden
00201
00202
00203 #line 152 "..\..\..\MainWindow.xaml"
00204 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00205     "CA1823:AvoidUnusedPrivateFields")]
00206     internal System.Windows.Controls.TextBlock ProgramCounterText;
00207 #line default
00208 #line hidden
00209
00210
00211 #line 153 "..\..\..\MainWindow.xaml"
00212 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00213     "CA1823:AvoidUnusedPrivateFields")]
00214     internal System.Windows.Controls.TextBlock CycleCountText;
00215 #line default
00216 #line hidden
00217
00218
00219 #line 154 "..\..\..\MainWindow.xaml"
00220 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00221     "CA1823:AvoidUnusedPrivateFields")]
00222     internal System.Windows.Controls.CheckBox CarryFlag;
00223 #line default
00224 #line hidden
00225
00226
00227 #line 155 "..\..\..\MainWindow.xaml"
00228 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00229     "CA1823:AvoidUnusedPrivateFields")]
00230     internal System.Windows.Controls.TextBlock CarryFlagText;
00231 #line default
00232 #line hidden
00233
00234
00235 #line 156 "..\..\..\MainWindow.xaml"
00236 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00237     "CA1823:AvoidUnusedPrivateFields")]
00238     internal System.Windows.Controls.CheckBox ZeroFlag;
00239 #line default
00240 #line hidden
00241
00242
00243 #line 157 "..\..\..\MainWindow.xaml"
00244 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00245     "CA1823:AvoidUnusedPrivateFields")]
00246     internal System.Windows.Controls.TextBlock ZeroFlagText;
```

```
00246
00247 #line default
00248 #line hidden
00249
00250
00251 #line 158 "..\..\..\MainWindow.xaml"
00252 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00253 "CA1823:AvoidUnusedPrivateFields")]
00253     internal System.Windows.Controls.CheckBox InterrupFlag;
00254
00255 #line default
00256 #line hidden
00257
00258
00259 #line 159 "..\..\..\MainWindow.xaml"
00260 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00261 "CA1823:AvoidUnusedPrivateFields")]
00261     internal System.Windows.Controls.TextBlock InterruptFlagText;
00262
00263 #line default
00264 #line hidden
00265
00266
00267 #line 160 "..\..\..\MainWindow.xaml"
00268 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00269 "CA1823:AvoidUnusedPrivateFields")]
00269     internal System.Windows.Controls.CheckBox BcdFlag;
00270
00271 #line default
00272 #line hidden
00273
00274
00275 #line 161 "..\..\..\MainWindow.xaml"
00276 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00277 "CA1823:AvoidUnusedPrivateFields")]
00277     internal System.Windows.Controls.TextBlock BcdFlagText;
00278
00279 #line default
00280 #line hidden
00281
00282
00283 #line 162 "..\..\..\MainWindow.xaml"
00284 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00285 "CA1823:AvoidUnusedPrivateFields")]
00285     internal System.Windows.Controls.CheckBox BreakFlag;
00286
00287 #line default
00288 #line hidden
00289
00290
00291 #line 163 "..\..\..\MainWindow.xaml"
00292 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00293 "CA1823:AvoidUnusedPrivateFields")]
00293     internal System.Windows.Controls.TextBlock BreakFlagText;
00294
00295 #line default
00296 #line hidden
00297
00298
00299 #line 164 "..\..\..\MainWindow.xaml"
00300 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00301 "CA1823:AvoidUnusedPrivateFields")]
00301     internal System.Windows.Controls.CheckBox OverflowFlag;
00302
00303 #line default
00304 #line hidden
00305
00306
00307 #line 165 "..\..\..\MainWindow.xaml"
00308 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00309 "CA1823:AvoidUnusedPrivateFields")]
00309     internal System.Windows.Controls.TextBlock OverflowFlagText;
00310
00311 #line default
00312 #line hidden
00313
00314
00315 #line 166 "..\..\..\MainWindow.xaml"
00316 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00317 "CA1823:AvoidUnusedPrivateFields")]
00317     internal System.Windows.Controls.CheckBox NegativeFlag;
00318
00319 #line default
00320 #line hidden
00321
00322
00323 #line 167 "..\..\..\MainWindow.xaml"
```

```

00324         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00325         internal System.Windows.Controls.TextBlock NegativeFlagText;
00326
00327 #line default
00328 #line hidden
00329
00330
00331 #line 168 "..\..\..\MainWindows\MainWindow.xaml"
00332         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00333         internal System.Windows.Controls.Slider CpuSpeed;
00334
00335 #line default
00336 #line hidden
00337
00338
00339 #line 169 "..\..\..\MainWindows\MainWindow.xaml"
00340         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00341         internal System.Windows.Controls.TextBlock SpeedText;
00342
00343 #line default
00344 #line hidden
00345
00346         private bool _contentLoaded;
00347
00348 /// <summary>
00349 /// InitializeComponent
00350 /// </summary>
00351         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00352         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00353         public void InitializeComponent() {
00354             if (_contentLoaded) {
00355                 return;
00356             }
00357             _contentLoaded = true;
00358             System.Uri resourceLocater = new System.Uri("/Emulator;component/mainwindow.xaml",
System.UriKind.Relative);
00359
00360 #line 1 "..\..\..\MainWindows\MainWindow.xaml"
00361             System.Windows.Application.LoadComponent(this, resourceLocater);
00362
00363 #line default
00364 #line hidden
00365         }
00366
00367         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00368         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00369
00370         [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00371         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
"CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00372         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
"CA1502:AvoidExcessiveComplexity")]
00373         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1800:DoNotCastUnnecessarily")]
00374         void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00375             switch (connectionId)
00376             {
00377                 case 1:
00378                     this.EmulatorWindow = ((Emulator.MainWindow)(target));
00379                     return;
00380                 case 2:
00381
00382 #line 72 "..\..\..\MainWindows\MainWindow.xaml"
00383                     ((System.Windows.Controls.MenuItem)(target)).Click += new
System.Windows.RoutedEventHandler(this.LoadFile);
00384
00385 #line default
00386 #line hidden
00387                     return;
00388                 case 3:
00389
00390 #line 73 "..\..\..\MainWindows\MainWindow.xaml"
00391                     ((System.Windows.Controls.MenuItem)(target)).Click += new
System.Windows.RoutedEventHandler(this.SaveFile);
00392
00393 #line default
00394 #line hidden
00395                     return;
00396                 case 4:
00397
00398 #line 74 "..\..\..\MainWindows\MainWindow.xaml"
00399                     ((System.Windows.Controls.MenuItem)(target)).Click += new
System.Windows.RoutedEventHandler(this.CloseFile);
00399

```

```
00400 #line default
00401 #line hidden
00402         return;
00403     case 5:
00404
00405 #line 76 "..\..\MainWindow.xaml"
00406         ((System.Windows.Controls.MenuItem) (target)).Click += new
            System.Windows.RoutedEventHandler(this.ToClose);
00407
00408 #line default
00409 #line hidden
00410         return;
00411     case 6:
00412         this.OutputLog = ((System.Windows.Controls.DataGrid) (target));
00413         return;
00414     case 7:
00415         this.Run = ((System.Windows.Controls.Button) (target));
00416         return;
00417     case 8:
00418         this.Step = ((System.Windows.Controls.Button) (target));
00419         return;
00420     case 9:
00421         this.Reset = ((System.Windows.Controls.Button) (target));
00422         return;
00423     case 10:
00424         this.RomFileNameText = ((System.Windows.Controls.TextBlock) (target));
00425         return;
00426     case 11:
00427         this.ComPortNameText = ((System.Windows.Controls.TextBlock) (target));
00428         return;
00429     case 12:
00430         this.Breakpoints = ((System.Windows.Controls.DataGrid) (target));
00431         return;
00432     case 13:
00433         this.YRegister = ((System.Windows.Controls.TextBox) (target));
00434         return;
00435     case 14:
00436         this.XRegister = ((System.Windows.Controls.TextBox) (target));
00437         return;
00438     case 15:
00439         this.Accumulator = ((System.Windows.Controls.TextBox) (target));
00440         return;
00441     case 16:
00442         this.StackPointer = ((System.Windows.Controls.TextBox) (target));
00443         return;
00444     case 17:
00445         this.ProgramCounter = ((System.Windows.Controls.TextBox) (target));
00446         return;
00447     case 18:
00448         this.Dissambly = ((System.Windows.Controls.TextBox) (target));
00449         return;
00450     case 19:
00451         this.CycleCount = ((System.Windows.Controls.TextBox) (target));
00452         return;
00453     case 20:
00454         this.XRegisterText = ((System.Windows.Controls.TextBlock) (target));
00455         return;
00456     case 21:
00457         this.YRegisterText = ((System.Windows.Controls.TextBlock) (target));
00458         return;
00459     case 22:
00460         this.StackPointerRegisterText = ((System.Windows.Controls.TextBlock) (target));
00461         return;
00462     case 23:
00463         this.AText = ((System.Windows.Controls.TextBlock) (target));
00464         return;
00465     case 24:
00466         this.CurrentInstructionText = ((System.Windows.Controls.TextBlock) (target));
00467         return;
00468     case 25:
00469         this.ProgramCounterText = ((System.Windows.Controls.TextBlock) (target));
00470         return;
00471     case 26:
00472         this.CycleCountText = ((System.Windows.Controls.TextBlock) (target));
00473         return;
00474     case 27:
00475         this.CarryFlag = ((System.Windows.Controls.CheckBox) (target));
00476         return;
00477     case 28:
00478         this.CarryFlagText = ((System.Windows.Controls.TextBlock) (target));
00479         return;
00480     case 29:
00481         this.ZeroFlag = ((System.Windows.Controls.CheckBox) (target));
00482         return;
00483     case 30:
00484         this.ZeroFlagText = ((System.Windows.Controls.TextBlock) (target));
00485         return;
```

```

00486         case 31:
00487             this.InterruptFlag = ((System.Windows.Controls.CheckBox) (target));
00488             return;
00489         case 32:
00490             this.InterruptFlagText = ((System.Windows.Controls.TextBlock) (target));
00491             return;
00492         case 33:
00493             this.BcdFlag = ((System.Windows.Controls.CheckBox) (target));
00494             return;
00495         case 34:
00496             this.BcdFlagText = ((System.Windows.Controls.TextBlock) (target));
00497             return;
00498         case 35:
00499             this.BreakFlag = ((System.Windows.Controls.CheckBox) (target));
00500             return;
00501         case 36:
00502             this.BreakFlagText = ((System.Windows.Controls.TextBlock) (target));
00503             return;
00504         case 37:
00505             this.OverflowFlag = ((System.Windows.Controls.CheckBox) (target));
00506             return;
00507         case 38:
00508             this.OverflowFlagText = ((System.Windows.Controls.TextBlock) (target));
00509             return;
00510         case 39:
00511             this.NegativeFlag = ((System.Windows.Controls.CheckBox) (target));
00512             return;
00513         case 40:
00514             this.NegativeFlagText = ((System.Windows.Controls.TextBlock) (target));
00515             return;
00516         case 41:
00517             this.CpuSpeed = ((System.Windows.Controls.Slider) (target));
00518             return;
00519         case 42:
00520             this.SpeedText = ((System.Windows.Controls.TextBlock) (target));
00521             return;
00522     }
00523     this._contentLoaded = true;
00524 }
00525 }
00526 }
00527

```

7.87 Emulator/obj/x86/Publish/MainWindow.g.cs File Reference

Classes

- class [Emulator.MainWindow](#)
Interaction logic for MainWindow.xaml

Namespaces

- namespace [Emulator](#)

7.88 MainWindow.g.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\Main Window.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00002 "B80FD745A4A855A2770EA0E1513AC1103AE202406DDCEC01FD8CA1DB0293F06C"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;

```



```

00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037     /// <summary>
00038     /// MainWindow
00039     /// </summary>
00040     public partial class MainWindow : System.Windows.Window,
        System.Windows.Markup.IComponentConnector {
00041
00042
00043         #line 2 "..\..\..\Mainwindow.xaml"
00044         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
        "CA1823:AvoidUnusedPrivateFields")]
00045         internal Emulator.MainWindow EmulatorWindow;
00046
00047         #line default
00048         #line hidden
00049
00050
00051         #line 89 "..\..\..\Mainwindow.xaml"
00052         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
        "CA1823:AvoidUnusedPrivateFields")]
00053         internal System.Windows.Controls.DataGrid OutputLog;
00054
00055         #line default
00056         #line hidden
00057
00058
00059         #line 106 "..\..\..\Mainwindow.xaml"
00060         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
        "CA1823:AvoidUnusedPrivateFields")]
00061         internal System.Windows.Controls.Button Run;
00062
00063         #line default
00064         #line hidden
00065
00066
00067         #line 107 "..\..\..\Mainwindow.xaml"
00068         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
        "CA1823:AvoidUnusedPrivateFields")]
00069         internal System.Windows.Controls.Button Step;
00070
00071         #line default
00072         #line hidden
00073
00074
00075         #line 108 "..\..\..\Mainwindow.xaml"
00076         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
        "CA1823:AvoidUnusedPrivateFields")]
00077         internal System.Windows.Controls.Button Reset;
00078
00079         #line default
00080         #line hidden
00081
00082
00083         #line 110 "..\..\..\Mainwindow.xaml"
00084         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
        "CA1823:AvoidUnusedPrivateFields")]
00085         internal System.Windows.Controls.TextBlock RomFileNameText;
00086
00087         #line default
00088         #line hidden
00089
00090
00091         #line 111 "..\..\..\Mainwindow.xaml"
00092         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
        "CA1823:AvoidUnusedPrivateFields")]
00093         internal System.Windows.Controls.TextBlock ComPortNameText;

```

```
00094
00095 #line default
00096 #line hidden
00097
00098
00099 #line 112 "..\..\..\MainWindow.xaml"
00100 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00101 "CA1823:AvoidUnusedPrivateFields")]
00102     internal System.Windows.Controls.DataGrid Breakpoints;
00103
00104 #line default
00105 #line hidden
00106
00107 #line 137 "..\..\..\MainWindow.xaml"
00108 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00109 "CA1823:AvoidUnusedPrivateFields")]
00110     internal System.Windows.Controls.TextBox YRegister;
00111
00112 #line default
00113 #line hidden
00114
00115 #line 138 "..\..\..\MainWindow.xaml"
00116 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00117 "CA1823:AvoidUnusedPrivateFields")]
00118     internal System.Windows.Controls.TextBox XRegister;
00119
00120 #line default
00121 #line hidden
00122
00123 #line 139 "..\..\..\MainWindow.xaml"
00124 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00125 "CA1823:AvoidUnusedPrivateFields")]
00126     internal System.Windows.Controls.TextBox Accumulator;
00127
00128 #line default
00129 #line hidden
00130
00131 #line 140 "..\..\..\MainWindow.xaml"
00132 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00133 "CA1823:AvoidUnusedPrivateFields")]
00134     internal System.Windows.Controls.TextBox StackPointer;
00135
00136 #line default
00137 #line hidden
00138
00139 #line 141 "..\..\..\MainWindow.xaml"
00140 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00141 "CA1823:AvoidUnusedPrivateFields")]
00142     internal System.Windows.Controls.TextBox ProgramCounter;
00143
00144 #line default
00145 #line hidden
00146
00147 #line 142 "..\..\..\MainWindow.xaml"
00148 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00149 "CA1823:AvoidUnusedPrivateFields")]
00150     internal System.Windows.Controls.TextBox Dissambly;
00151
00152 #line default
00153 #line hidden
00154
00155 #line 143 "..\..\..\MainWindow.xaml"
00156 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00157 "CA1823:AvoidUnusedPrivateFields")]
00158     internal System.Windows.Controls.TextBox CycleCount;
00159
00160 #line default
00161 #line hidden
00162
00163 #line 144 "..\..\..\MainWindow.xaml"
00164 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00165 "CA1823:AvoidUnusedPrivateFields")]
00166     internal System.Windows.Controls.TextBlock XRegisterText;
00167
00168 #line default
00169 #line hidden
00170
00171 #line 145 "..\..\..\MainWindow.xaml"
```

```
00172         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00173         "CA1823:AvoidUnusedPrivateFields")]
00174         internal System.Windows.Controls.TextBlock YRegisterText;
00175 #line default
00176 #line hidden
00177
00178
00179 #line 146 "..\..\..\MainWindow.xaml"
00180         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00181         "CA1823:AvoidUnusedPrivateFields")]
00182         internal System.Windows.Controls.TextBlock StackPointerRegisterText;
00183 #line default
00184 #line hidden
00185
00186
00187 #line 147 "..\..\..\MainWindow.xaml"
00188         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00189         "CA1823:AvoidUnusedPrivateFields")]
00190         internal System.Windows.Controls.TextBlock AText;
00191 #line default
00192 #line hidden
00193
00194
00195 #line 148 "..\..\..\MainWindow.xaml"
00196         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00197         "CA1823:AvoidUnusedPrivateFields")]
00198         internal System.Windows.Controls.TextBlock CurrentInstructionText;
00199 #line default
00200 #line hidden
00201
00202
00203 #line 149 "..\..\..\MainWindow.xaml"
00204         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00205         "CA1823:AvoidUnusedPrivateFields")]
00206         internal System.Windows.Controls.TextBlock ProgramCounterText;
00207 #line default
00208 #line hidden
00209
00210
00211 #line 150 "..\..\..\MainWindow.xaml"
00212         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00213         "CA1823:AvoidUnusedPrivateFields")]
00214         internal System.Windows.Controls.TextBlock CycleCountText;
00215 #line default
00216 #line hidden
00217
00218
00219 #line 151 "..\..\..\MainWindow.xaml"
00220         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00221         "CA1823:AvoidUnusedPrivateFields")]
00222         internal System.Windows.Controls.CheckBox CarryFlag;
00223 #line default
00224 #line hidden
00225
00226
00227 #line 152 "..\..\..\MainWindow.xaml"
00228         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00229         "CA1823:AvoidUnusedPrivateFields")]
00230         internal System.Windows.Controls.TextBlock CarryFlagText;
00231 #line default
00232 #line hidden
00233
00234
00235 #line 153 "..\..\..\MainWindow.xaml"
00236         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00237         "CA1823:AvoidUnusedPrivateFields")]
00238         internal System.Windows.Controls.CheckBox ZeroFlag;
00239 #line default
00240 #line hidden
00241
00242
00243 #line 154 "..\..\..\MainWindow.xaml"
00244         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00245         "CA1823:AvoidUnusedPrivateFields")]
00246         internal System.Windows.Controls.TextBlock ZeroFlagText;
00247 #line default
00248 #line hidden
```

```

00249
00250
00251 #line 155 "..\..\..\MainWindow.xaml"
00252     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00253         "CA1823:AvoidUnusedPrivateFields")]
00253         internal System.Windows.Controls.CheckBox InterrupFlag;
00254
00255 #line default
00256 #line hidden
00257
00258
00259 #line 156 "..\..\..\MainWindow.xaml"
00260     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00261         "CA1823:AvoidUnusedPrivateFields")]
00261         internal System.Windows.Controls.TextBlock InterruptFlagText;
00262
00263 #line default
00264 #line hidden
00265
00266
00267 #line 157 "..\..\..\MainWindow.xaml"
00268     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00269         "CA1823:AvoidUnusedPrivateFields")]
00269         internal System.Windows.Controls.CheckBox BcdFlag;
00270
00271 #line default
00272 #line hidden
00273
00274
00275 #line 158 "..\..\..\MainWindow.xaml"
00276     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00277         "CA1823:AvoidUnusedPrivateFields")]
00277         internal System.Windows.Controls.TextBlock BcdFlagText;
00278
00279 #line default
00280 #line hidden
00281
00282
00283 #line 159 "..\..\..\MainWindow.xaml"
00284     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00285         "CA1823:AvoidUnusedPrivateFields")]
00285         internal System.Windows.Controls.CheckBox BreakFlag;
00286
00287 #line default
00288 #line hidden
00289
00290
00291 #line 160 "..\..\..\MainWindow.xaml"
00292     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00293         "CA1823:AvoidUnusedPrivateFields")]
00293         internal System.Windows.Controls.TextBlock BreakFlagText;
00294
00295 #line default
00296 #line hidden
00297
00298
00299 #line 161 "..\..\..\MainWindow.xaml"
00300     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00301         "CA1823:AvoidUnusedPrivateFields")]
00301         internal System.Windows.Controls.CheckBox OverflowFlag;
00302
00303 #line default
00304 #line hidden
00305
00306
00307 #line 162 "..\..\..\MainWindow.xaml"
00308     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00309         "CA1823:AvoidUnusedPrivateFields")]
00309         internal System.Windows.Controls.TextBlock OverflowFlagText;
00310
00311 #line default
00312 #line hidden
00313
00314
00315 #line 163 "..\..\..\MainWindow.xaml"
00316     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00317         "CA1823:AvoidUnusedPrivateFields")]
00317         internal System.Windows.Controls.CheckBox NegativeFlag;
00318
00319 #line default
00320 #line hidden
00321
00322
00323 #line 164 "..\..\..\MainWindow.xaml"
00324     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00325         "CA1823:AvoidUnusedPrivateFields")]
00325         internal System.Windows.Controls.TextBlock NegativeFlagText;

```

```

00326
00327 #line default
00328 #line hidden
00329
00330
00331 #line 165 "..\..\..\MainWindow.xaml"
00332 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00333     internal System.Windows.Controls.Slider CpuSpeed;
00334
00335 #line default
00336 #line hidden
00337
00338
00339 #line 166 "..\..\..\MainWindow.xaml"
00340 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00341     internal System.Windows.Controls.TextBlock SpeedText;
00342
00343 #line default
00344 #line hidden
00345
00346     private bool _contentLoaded;
00347
00348 /// <summary>
00349 /// InitializeComponent
00350 /// </summary>
00351 [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00352 [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00353 public void InitializeComponent() {
00354     if (_contentLoaded) {
00355         return;
00356     }
00357     _contentLoaded = true;
00358     System.Uri resourceLocater = new System.Uri("/Emulator;component/mainwindow.xaml",
System.UriKind.Relative);
00359
00360 #line 1 "..\..\..\MainWindow.xaml"
00361     System.Windows.Application.LoadComponent(this, resourceLocater);
00362
00363 #line default
00364 #line hidden
00365 }
00366
00367 [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00368 [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00369
00370 [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00371 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
"CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00372 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
"CA1502:AvoidExcessiveComplexity")]
00373 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1800:DoNotCastUnnecessarily")]
00374 void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00375     {
00376         case 1:
00377             this.EmulatorWindow = ((Emulator.MainWindow)(target));
00378             return;
00379         case 2:
00380
00381 #line 72 "..\..\..\MainWindow.xaml"
00382             ((System.Windows.Controls.MenuItem)(target)).Click += new
System.Windows.RoutedEventHandler(this.LoadFile);
00383
00384 #line default
00385 #line hidden
00386             return;
00387         case 3:
00388
00389 #line 73 "..\..\..\MainWindow.xaml"
00390             ((System.Windows.Controls.MenuItem)(target)).Click += new
System.Windows.RoutedEventHandler(this.SaveFile);
00391
00392 #line default
00393 #line hidden
00394             return;
00395         case 4:
00396
00397 #line 74 "..\..\..\MainWindow.xaml"
00398             ((System.Windows.Controls.MenuItem)(target)).Click += new
System.Windows.RoutedEventHandler(this.CloseFile);
00399
00400 #line default
00401 #line hidden
00402             return;

```

```
00403         case 5:
00404
00405 #line 76 "..\..\Main\nMainWindow.xaml"
00406         ((System.Windows.Controls.MenuItem)(target)).Click += new
            System.Windows.RoutedEventHandler(this.ToClose);
00407
00408 #line default
00409 #line hidden
00410         return;
00411         case 6:
00412             this.OutputLog = ((System.Windows.Controls.DataGrid)(target));
00413             return;
00414         case 7:
00415             this.Run = ((System.Windows.Controls.Button)(target));
00416             return;
00417         case 8:
00418             this.Step = ((System.Windows.Controls.Button)(target));
00419             return;
00420         case 9:
00421             this.Reset = ((System.Windows.Controls.Button)(target));
00422             return;
00423         case 10:
00424             this.RomFileNameText = ((System.Windows.Controls.TextBlock)(target));
00425             return;
00426         case 11:
00427             this.ComPortNameText = ((System.Windows.Controls.TextBlock)(target));
00428             return;
00429         case 12:
00430             this.Breakpoints = ((System.Windows.Controls.DataGrid)(target));
00431             return;
00432         case 13:
00433             this.YRegister = ((System.Windows.Controls.TextBox)(target));
00434             return;
00435         case 14:
00436             this.XRegister = ((System.Windows.Controls.TextBox)(target));
00437             return;
00438         case 15:
00439             this.Accumulator = ((System.Windows.Controls.TextBox)(target));
00440             return;
00441         case 16:
00442             this.StackPointer = ((System.Windows.Controls.TextBox)(target));
00443             return;
00444         case 17:
00445             this.ProgramCounter = ((System.Windows.Controls.TextBox)(target));
00446             return;
00447         case 18:
00448             this.Dissassembly = ((System.Windows.Controls.TextBox)(target));
00449             return;
00450         case 19:
00451             this.CycleCount = ((System.Windows.Controls.TextBox)(target));
00452             return;
00453         case 20:
00454             this.XRegisterText = ((System.Windows.Controls.TextBlock)(target));
00455             return;
00456         case 21:
00457             this.YRegisterText = ((System.Windows.Controls.TextBlock)(target));
00458             return;
00459         case 22:
00460             this.StackPointerRegisterText = ((System.Windows.Controls.TextBlock)(target));
00461             return;
00462         case 23:
00463             this.AText = ((System.Windows.Controls.TextBlock)(target));
00464             return;
00465         case 24:
00466             this.CurrentInstructionText = ((System.Windows.Controls.TextBlock)(target));
00467             return;
00468         case 25:
00469             this.ProgramCounterText = ((System.Windows.Controls.TextBlock)(target));
00470             return;
00471         case 26:
00472             this.CycleCountText = ((System.Windows.Controls.TextBlock)(target));
00473             return;
00474         case 27:
00475             this.CarryFlag = ((System.Windows.Controls.CheckBox)(target));
00476             return;
00477         case 28:
00478             this.CarryFlagText = ((System.Windows.Controls.TextBlock)(target));
00479             return;
00480         case 29:
00481             this.ZeroFlag = ((System.Windows.Controls.CheckBox)(target));
00482             return;
00483         case 30:
00484             this.ZeroFlagText = ((System.Windows.Controls.TextBlock)(target));
00485             return;
00486         case 31:
00487             this.InterruptFlag = ((System.Windows.Controls.CheckBox)(target));
00488             return;
```

```

00489         case 32:
00490             this.InterruptFlagText = ((System.Windows.Controls.TextBlock) (target));
00491             return;
00492         case 33:
00493             this.BcdFlag = ((System.Windows.Controls.CheckBox) (target));
00494             return;
00495         case 34:
00496             this.BcdFlagText = ((System.Windows.Controls.TextBlock) (target));
00497             return;
00498         case 35:
00499             this.BreakFlag = ((System.Windows.Controls.CheckBox) (target));
00500             return;
00501         case 36:
00502             this.BreakFlagText = ((System.Windows.Controls.TextBlock) (target));
00503             return;
00504         case 37:
00505             this.OverflowFlag = ((System.Windows.Controls.CheckBox) (target));
00506             return;
00507         case 38:
00508             this.OverflowFlagText = ((System.Windows.Controls.TextBlock) (target));
00509             return;
00510         case 39:
00511             this.NegativeFlag = ((System.Windows.Controls.CheckBox) (target));
00512             return;
00513         case 40:
00514             this.NegativeFlagText = ((System.Windows.Controls.TextBlock) (target));
00515             return;
00516         case 41:
00517             this.CpuSpeed = ((System.Windows.Controls.Slider) (target));
00518             return;
00519         case 42:
00520             this.SpeedText = ((System.Windows.Controls.TextBlock) (target));
00521             return;
00522     }
00523     this._contentLoaded = true;
00524 }
00525 }
00526 }
00527

```

7.89 Emulator/obj/x86/Release/MainWindow.g.cs File Reference

Classes

- class [Emulator.MainWindow](#)
Interaction logic for MainWindow.xaml

Namespaces

- namespace [Emulator](#)

7.90 MainWindow.g.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\MainWindows\MainWindow.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00002 "1FB39AF98423D8DD6333B173E814398E2016BACFE25491D8BD824F3F8A79E0A5"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;

```

```

00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037 /// <summary>
00038 /// MainWindow
00039 /// </summary>
00040 public partial class MainWindow : System.Windows.Window,
    System.Windows.Markup.IComponentConnector {
00041
00042
00043 #line 2 "..\..\..\Main\MainWindow.xaml"
00044     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00045         "CA1823:AvoidUnusedPrivateFields")]
00046     internal Emulator.MainWindow EmulatorWindow;
00047
00048 #line default
00049 #line hidden
00050
00051 #line 92 "..\..\..\Main\MainWindow.xaml"
00052     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00053         "CA1823:AvoidUnusedPrivateFields")]
00054     internal System.Windows.Controls.DataGrid OutputLog;
00055
00056 #line default
00057 #line hidden
00058
00059 #line 109 "..\..\..\Main\MainWindow.xaml"
00060     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00061         "CA1823:AvoidUnusedPrivateFields")]
00062     internal System.Windows.Controls.Button Run;
00063
00064 #line default
00065 #line hidden
00066
00067 #line 110 "..\..\..\Main\MainWindow.xaml"
00068     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00069         "CA1823:AvoidUnusedPrivateFields")]
00070     internal System.Windows.Controls.Button Step;
00071
00072 #line default
00073 #line hidden
00074
00075 #line 111 "..\..\..\Main\MainWindow.xaml"
00076     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00077         "CA1823:AvoidUnusedPrivateFields")]
00078     internal System.Windows.Controls.Button Reset;
00079
00080 #line default
00081 #line hidden
00082
00083 #line 113 "..\..\..\Main\MainWindow.xaml"
00084     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00085         "CA1823:AvoidUnusedPrivateFields")]
00086     internal System.Windows.Controls.TextBlock RomFileNameText;
00087
00088 #line default
00089 #line hidden
00090
00091 #line 114 "..\..\..\Main\MainWindow.xaml"
00092     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00093         "CA1823:AvoidUnusedPrivateFields")]
00094     internal System.Windows.Controls.TextBlock ComPortNameText;
00095
00096 #line default
00097 #line hidden

```



```
00097
00098
00099 #line 115 "..\..\..\MainWindow.xaml"
00100 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00101     "CA1823:AvoidUnusedPrivateFields")]
00102     internal System.Windows.Controls.DataGrid Breakpoints;
00103
00104 #line default
00105 #line hidden
00106
00107 #line 140 "..\..\..\MainWindow.xaml"
00108 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00109     "CA1823:AvoidUnusedPrivateFields")]
00110     internal System.Windows.Controls.TextBox YRegister;
00111
00112 #line default
00113 #line hidden
00114
00115 #line 141 "..\..\..\MainWindow.xaml"
00116 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00117     "CA1823:AvoidUnusedPrivateFields")]
00118     internal System.Windows.Controls.TextBox XRegister;
00119
00120 #line default
00121 #line hidden
00122
00123 #line 142 "..\..\..\MainWindow.xaml"
00124 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00125     "CA1823:AvoidUnusedPrivateFields")]
00126     internal System.Windows.Controls.TextBox Accumulator;
00127
00128 #line default
00129 #line hidden
00130
00131 #line 143 "..\..\..\MainWindow.xaml"
00132 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00133     "CA1823:AvoidUnusedPrivateFields")]
00134     internal System.Windows.Controls.TextBox StackPointer;
00135
00136 #line default
00137 #line hidden
00138
00139 #line 144 "..\..\..\MainWindow.xaml"
00140 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00141     "CA1823:AvoidUnusedPrivateFields")]
00142     internal System.Windows.Controls.TextBox ProgramCounter;
00143
00144 #line default
00145 #line hidden
00146
00147 #line 145 "..\..\..\MainWindow.xaml"
00148 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00149     "CA1823:AvoidUnusedPrivateFields")]
00150     internal System.Windows.Controls.TextBox Dissambly;
00151
00152 #line default
00153 #line hidden
00154
00155 #line 146 "..\..\..\MainWindow.xaml"
00156 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00157     "CA1823:AvoidUnusedPrivateFields")]
00158     internal System.Windows.Controls.TextBox CycleCount;
00159
00160 #line default
00161 #line hidden
00162
00163 #line 147 "..\..\..\MainWindow.xaml"
00164 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00165     "CA1823:AvoidUnusedPrivateFields")]
00166     internal System.Windows.Controls.TextBlock XRegisterText;
00167
00168 #line default
00169 #line hidden
00170
00171 #line 148 "..\..\..\MainWindow.xaml"
00172 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00173     "CA1823:AvoidUnusedPrivateFields")]
00174     internal System.Windows.Controls.TextBlock YRegisterText;
```

```
00174
00175 #line default
00176 #line hidden
00177
00178
00179 #line 149 "..\..\..\MainWindow.xaml"
00180     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00181         "CA1823:AvoidUnusedPrivateFields")]
00182     internal System.Windows.Controls.TextBlock StackPointerRegisterText;
00183
00184 #line default
00185 #line hidden
00186
00187 #line 150 "..\..\..\MainWindow.xaml"
00188     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00189         "CA1823:AvoidUnusedPrivateFields")]
00190     internal System.Windows.Controls.TextBlock AText;
00191
00192 #line default
00193 #line hidden
00194
00195 #line 151 "..\..\..\MainWindow.xaml"
00196     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00197         "CA1823:AvoidUnusedPrivateFields")]
00198     internal System.Windows.Controls.TextBlock CurrentInstructionText;
00199
00200 #line default
00201 #line hidden
00202
00203 #line 152 "..\..\..\MainWindow.xaml"
00204     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00205         "CA1823:AvoidUnusedPrivateFields")]
00206     internal System.Windows.Controls.TextBlock ProgramCounterText;
00207
00208 #line default
00209 #line hidden
00210
00211 #line 153 "..\..\..\MainWindow.xaml"
00212     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00213         "CA1823:AvoidUnusedPrivateFields")]
00214     internal System.Windows.Controls.TextBlock CycleCountText;
00215
00216 #line default
00217 #line hidden
00218
00219 #line 154 "..\..\..\MainWindow.xaml"
00220     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00221         "CA1823:AvoidUnusedPrivateFields")]
00222     internal System.Windows.Controls.CheckBox CarryFlag;
00223
00224 #line default
00225 #line hidden
00226
00227 #line 155 "..\..\..\MainWindow.xaml"
00228     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00229         "CA1823:AvoidUnusedPrivateFields")]
00230     internal System.Windows.Controls.TextBlock CarryFlagText;
00231
00232 #line default
00233 #line hidden
00234
00235 #line 156 "..\..\..\MainWindow.xaml"
00236     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00237         "CA1823:AvoidUnusedPrivateFields")]
00238     internal System.Windows.Controls.CheckBox ZeroFlag;
00239
00240 #line default
00241 #line hidden
00242
00243 #line 157 "..\..\..\MainWindow.xaml"
00244     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00245         "CA1823:AvoidUnusedPrivateFields")]
00246     internal System.Windows.Controls.TextBlock ZeroFlagText;
00247
00248 #line default
00249 #line hidden
00250
00251 #line 158 "..\..\..\MainWindow.xaml"
```

```
00252         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00253         "CA1823:AvoidUnusedPrivateFields")]
00253         internal System.Windows.Controls.CheckBox InterrupFlag;
00254
00255 #line default
00256 #line hidden
00257
00258
00259 #line 159 "..\..\..\MainWindow.xaml"
00260         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00261         "CA1823:AvoidUnusedPrivateFields")]
00261         internal System.Windows.Controls.TextBlock InterruptFlagText;
00262
00263 #line default
00264 #line hidden
00265
00266
00267 #line 160 "..\..\..\MainWindow.xaml"
00268         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00269         "CA1823:AvoidUnusedPrivateFields")]
00269         internal System.Windows.Controls.CheckBox BcdFlag;
00270
00271 #line default
00272 #line hidden
00273
00274
00275 #line 161 "..\..\..\MainWindow.xaml"
00276         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00277         "CA1823:AvoidUnusedPrivateFields")]
00277         internal System.Windows.Controls.TextBlock BcdFlagText;
00278
00279 #line default
00280 #line hidden
00281
00282
00283 #line 162 "..\..\..\MainWindow.xaml"
00284         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00285         "CA1823:AvoidUnusedPrivateFields")]
00285         internal System.Windows.Controls.CheckBox BreakFlag;
00286
00287 #line default
00288 #line hidden
00289
00290
00291 #line 163 "..\..\..\MainWindow.xaml"
00292         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00293         "CA1823:AvoidUnusedPrivateFields")]
00293         internal System.Windows.Controls.TextBlock BreakFlagText;
00294
00295 #line default
00296 #line hidden
00297
00298
00299 #line 164 "..\..\..\MainWindow.xaml"
00300         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00301         "CA1823:AvoidUnusedPrivateFields")]
00301         internal System.Windows.Controls.CheckBox OverflowFlag;
00302
00303 #line default
00304 #line hidden
00305
00306
00307 #line 165 "..\..\..\MainWindow.xaml"
00308         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00309         "CA1823:AvoidUnusedPrivateFields")]
00309         internal System.Windows.Controls.TextBlock OverflowFlagText;
00310
00311 #line default
00312 #line hidden
00313
00314
00315 #line 166 "..\..\..\MainWindow.xaml"
00316         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00317         "CA1823:AvoidUnusedPrivateFields")]
00317         internal System.Windows.Controls.CheckBox NegativeFlag;
00318
00319 #line default
00320 #line hidden
00321
00322
00323 #line 167 "..\..\..\MainWindow.xaml"
00324         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00325         "CA1823:AvoidUnusedPrivateFields")]
00325         internal System.Windows.Controls.TextBlock NegativeFlagText;
00326
00327 #line default
00328 #line hidden
```

```

00329
00330
00331 #line 168 "..\..\..\MainWindow.xaml"
00332 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00333     "CA1823:AvoidUnusedPrivateFields")]
00334     internal System.Windows.Controls.Slider CpuSpeed;
00335
00336 #line default
00337 #line hidden
00338
00339 #line 169 "..\..\..\MainWindow.xaml"
00340 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00341     "CA1823:AvoidUnusedPrivateFields")]
00342     internal System.Windows.Controls.TextBlock SpeedText;
00343
00344 #line default
00345 #line hidden
00346
00347     private bool _contentLoaded;
00348
00349 /// <summary>
00350 /// InitializeComponent
00351 /// </summary>
00352 [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00353 [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00354 public void InitializeComponent() {
00355     if (_contentLoaded) {
00356         return;
00357     }
00358     _contentLoaded = true;
00359     System.Uri resourceLocater = new System.Uri("/Emulator;component/mainwindow.xaml",
00360         System.UriKind.Relative);
00361
00362 #line 1 "..\..\..\MainWindow.xaml"
00363     System.Windows.Application.LoadComponent(this, resourceLocater);
00364
00365 #line default
00366 #line hidden
00367     }
00368
00369 [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00370 [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00371 [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00372 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
00373     "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00374 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
00375     "CA1502:AvoidExcessiveComplexity")]
00376 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00377     "CA1800:DoNotCastUnnecessarily")]
00378 void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00379     switch (connectionId)
00380     {
00381     {
00382         case 1:
00383             this.EmulatorWindow = ((Emulator.MainWindow)(target));
00384             return;
00385         case 2:
00386
00387 #line 72 "..\..\..\MainWindow.xaml"
00388         ((System.Windows.Controls.MenuItem)(target)).Click += new
00389             System.Windows.RoutedEventHandler(this.LoadFile);
00390
00391 #line default
00392 #line hidden
00393         return;
00394         case 3:
00395
00396 #line 73 "..\..\..\MainWindow.xaml"
00397         ((System.Windows.Controls.MenuItem)(target)).Click += new
00398             System.Windows.RoutedEventHandler(this.SaveFile);
00399
00400 #line default
00401 #line hidden
00402         return;
00403         case 4:
00404
00405 #line 74 "..\..\..\MainWindow.xaml"
00406         ((System.Windows.Controls.MenuItem)(target)).Click += new
00407             System.Windows.RoutedEventHandler(this.CloseFile);
00408
00409 #line default
00410 #line hidden
00411         return;
00412         case 5:
00413
00414 #line 76 "..\..\..\MainWindow.xaml"

```

```
00406         ((System.Windows.Controls.MenuItem)(target)).Click += new
System.Windows.RoutedEventHandler(this.ToClose);
00407
00408 #line default
00409 #line hidden
00410         return;
00411         case 6:
00412             this.OutputLog = ((System.Windows.Controls.DataGrid)(target));
00413             return;
00414         case 7:
00415             this.Run = ((System.Windows.Controls.Button)(target));
00416             return;
00417         case 8:
00418             this.Step = ((System.Windows.Controls.Button)(target));
00419             return;
00420         case 9:
00421             this.Reset = ((System.Windows.Controls.Button)(target));
00422             return;
00423         case 10:
00424             this.RomFileNameText = ((System.Windows.Controls.TextBlock)(target));
00425             return;
00426         case 11:
00427             this.ComPortNameText = ((System.Windows.Controls.TextBlock)(target));
00428             return;
00429         case 12:
00430             this.Breakpoints = ((System.Windows.Controls.DataGrid)(target));
00431             return;
00432         case 13:
00433             this.YRegister = ((System.Windows.Controls.TextBox)(target));
00434             return;
00435         case 14:
00436             this.XRegister = ((System.Windows.Controls.TextBox)(target));
00437             return;
00438         case 15:
00439             this.Accumulator = ((System.Windows.Controls.TextBox)(target));
00440             return;
00441         case 16:
00442             this.StackPointer = ((System.Windows.Controls.TextBox)(target));
00443             return;
00444         case 17:
00445             this.ProgramCounter = ((System.Windows.Controls.TextBox)(target));
00446             return;
00447         case 18:
00448             this.Dissassembly = ((System.Windows.Controls.TextBox)(target));
00449             return;
00450         case 19:
00451             this.CycleCount = ((System.Windows.Controls.TextBox)(target));
00452             return;
00453         case 20:
00454             this.XRegisterText = ((System.Windows.Controls.TextBlock)(target));
00455             return;
00456         case 21:
00457             this.YRegisterText = ((System.Windows.Controls.TextBlock)(target));
00458             return;
00459         case 22:
00460             this.StackPointerRegisterText = ((System.Windows.Controls.TextBlock)(target));
00461             return;
00462         case 23:
00463             this.AText = ((System.Windows.Controls.TextBlock)(target));
00464             return;
00465         case 24:
00466             this.CurrentInstructionText = ((System.Windows.Controls.TextBlock)(target));
00467             return;
00468         case 25:
00469             this.ProgramCounterText = ((System.Windows.Controls.TextBlock)(target));
00470             return;
00471         case 26:
00472             this.CycleCountText = ((System.Windows.Controls.TextBlock)(target));
00473             return;
00474         case 27:
00475             this.CarryFlag = ((System.Windows.Controls.CheckBox)(target));
00476             return;
00477         case 28:
00478             this.CarryFlagText = ((System.Windows.Controls.TextBlock)(target));
00479             return;
00480         case 29:
00481             this.ZeroFlag = ((System.Windows.Controls.CheckBox)(target));
00482             return;
00483         case 30:
00484             this.ZeroFlagText = ((System.Windows.Controls.TextBlock)(target));
00485             return;
00486         case 31:
00487             this.InterruptFlag = ((System.Windows.Controls.CheckBox)(target));
00488             return;
00489         case 32:
00490             this.InterruptFlagText = ((System.Windows.Controls.TextBlock)(target));
00491             return;
```

```

00492         case 33:
00493             this.BcdFlag = ((System.Windows.Controls.CheckBox) (target));
00494             return;
00495         case 34:
00496             this.BcdFlagText = ((System.Windows.Controls.TextBlock) (target));
00497             return;
00498         case 35:
00499             this.BreakFlag = ((System.Windows.Controls.CheckBox) (target));
00500             return;
00501         case 36:
00502             this.BreakFlagText = ((System.Windows.Controls.TextBlock) (target));
00503             return;
00504         case 37:
00505             this.OverflowFlag = ((System.Windows.Controls.CheckBox) (target));
00506             return;
00507         case 38:
00508             this.OverflowFlagText = ((System.Windows.Controls.TextBlock) (target));
00509             return;
00510         case 39:
00511             this.NegativeFlag = ((System.Windows.Controls.CheckBox) (target));
00512             return;
00513         case 40:
00514             this.NegativeFlagText = ((System.Windows.Controls.TextBlock) (target));
00515             return;
00516         case 41:
00517             this.CpuSpeed = ((System.Windows.Controls.Slider) (target));
00518             return;
00519         case 42:
00520             this.SpeedText = ((System.Windows.Controls.TextBlock) (target));
00521             return;
00522     }
00523     this._contentLoaded = true;
00524 }
00525 }
00526 }
00527

```

7.91 Emulator/obj/x86/Debug/MainWindow.g.i.cs File Reference

Classes

- class [Emulator.MainWindow](#)
Interaction logic for MainWindow.xaml

Namespaces

- namespace [Emulator](#)

7.92 MainWindow.g.i.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\MainWindows\MainWindow.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00002 "1FB39AF98423D8DD6333B173E814398E2016BACFE25491D8BD824F3F8A79E0A5"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;

```

```

00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037 /// <summary>
00038 /// MainWindow
00039 /// </summary>
00040 public partial class MainWindow : System.Windows.Window,
System.Windows.Markup.IComponentConnector {
00041
00042
00043 #line 2 "..\..\..\Mainwindow.xaml"
00044 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00045     internal Emulator.MainWindow EmulatorWindow;
00046
00047 #line default
00048 #line hidden
00049
00050
00051 #line 92 "..\..\..\Mainwindow.xaml"
00052 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00053     internal System.Windows.Controls.DataGrid OutputLog;
00054
00055 #line default
00056 #line hidden
00057
00058
00059 #line 109 "..\..\..\Mainwindow.xaml"
00060 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00061     internal System.Windows.Controls.Button Run;
00062
00063 #line default
00064 #line hidden
00065
00066
00067 #line 110 "..\..\..\Mainwindow.xaml"
00068 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00069     internal System.Windows.Controls.Button Step;
00070
00071 #line default
00072 #line hidden
00073
00074
00075 #line 111 "..\..\..\Mainwindow.xaml"
00076 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00077     internal System.Windows.Controls.Button Reset;
00078
00079 #line default
00080 #line hidden
00081
00082
00083 #line 113 "..\..\..\Mainwindow.xaml"
00084 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00085     internal System.Windows.Controls.TextBlock RomFileNameText;
00086
00087 #line default
00088 #line hidden
00089
00090
00091 #line 114 "..\..\..\Mainwindow.xaml"
00092 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00093     internal System.Windows.Controls.TextBlock ComPortNameText;
00094
00095 #line default
00096 #line hidden
00097
00098
00099 #line 115 "..\..\..\Mainwindow.xaml"

```

```
00100         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00101         "CA1823:AvoidUnusedPrivateFields")]
00102         internal System.Windows.Controls.DataGrid Breakpoints;
00103 #line default
00104 #line hidden
00105
00106
00107 #line 140 "..\..\..\MainWindow.xaml"
00108         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00109         "CA1823:AvoidUnusedPrivateFields")]
00110         internal System.Windows.Controls.TextBox YRegister;
00111 #line default
00112 #line hidden
00113
00114
00115 #line 141 "..\..\..\MainWindow.xaml"
00116         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00117         "CA1823:AvoidUnusedPrivateFields")]
00118         internal System.Windows.Controls.TextBox XRegister;
00119 #line default
00120 #line hidden
00121
00122
00123 #line 142 "..\..\..\MainWindow.xaml"
00124         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00125         "CA1823:AvoidUnusedPrivateFields")]
00126         internal System.Windows.Controls.TextBox Accumulator;
00127 #line default
00128 #line hidden
00129
00130
00131 #line 143 "..\..\..\MainWindow.xaml"
00132         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00133         "CA1823:AvoidUnusedPrivateFields")]
00134         internal System.Windows.Controls.TextBox StackPointer;
00135 #line default
00136 #line hidden
00137
00138
00139 #line 144 "..\..\..\MainWindow.xaml"
00140         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00141         "CA1823:AvoidUnusedPrivateFields")]
00142         internal System.Windows.Controls.TextBox ProgramCounter;
00143 #line default
00144 #line hidden
00145
00146
00147 #line 145 "..\..\..\MainWindow.xaml"
00148         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00149         "CA1823:AvoidUnusedPrivateFields")]
00150         internal System.Windows.Controls.TextBox Dissambly;
00151 #line default
00152 #line hidden
00153
00154
00155 #line 146 "..\..\..\MainWindow.xaml"
00156         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00157         "CA1823:AvoidUnusedPrivateFields")]
00158         internal System.Windows.Controls.TextBox CycleCount;
00159 #line default
00160 #line hidden
00161
00162
00163 #line 147 "..\..\..\MainWindow.xaml"
00164         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00165         "CA1823:AvoidUnusedPrivateFields")]
00166         internal System.Windows.Controls.TextBlock XRegisterText;
00167 #line default
00168 #line hidden
00169
00170
00171 #line 148 "..\..\..\MainWindow.xaml"
00172         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00173         "CA1823:AvoidUnusedPrivateFields")]
00174         internal System.Windows.Controls.TextBlock YRegisterText;
00175 #line default
00176 #line hidden
```



```
00177
00178
00179 #line 149 "..\..\..\MainWindow.xaml"
00180 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00181     "CA1823:AvoidUnusedPrivateFields")]
00182     internal System.Windows.Controls.TextBlock StackPointerRegisterText;
00183 #line default
00184 #line hidden
00185
00186
00187 #line 150 "..\..\..\MainWindow.xaml"
00188 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00189     "CA1823:AvoidUnusedPrivateFields")]
00190     internal System.Windows.Controls.TextBlock AText;
00191 #line default
00192 #line hidden
00193
00194
00195 #line 151 "..\..\..\MainWindow.xaml"
00196 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00197     "CA1823:AvoidUnusedPrivateFields")]
00198     internal System.Windows.Controls.TextBlock CurrentInstructionText;
00199 #line default
00200 #line hidden
00201
00202
00203 #line 152 "..\..\..\MainWindow.xaml"
00204 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00205     "CA1823:AvoidUnusedPrivateFields")]
00206     internal System.Windows.Controls.TextBlock ProgramCounterText;
00207 #line default
00208 #line hidden
00209
00210
00211 #line 153 "..\..\..\MainWindow.xaml"
00212 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00213     "CA1823:AvoidUnusedPrivateFields")]
00214     internal System.Windows.Controls.TextBlock CycleCountText;
00215 #line default
00216 #line hidden
00217
00218
00219 #line 154 "..\..\..\MainWindow.xaml"
00220 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00221     "CA1823:AvoidUnusedPrivateFields")]
00222     internal System.Windows.Controls.CheckBox CarryFlag;
00223 #line default
00224 #line hidden
00225
00226
00227 #line 155 "..\..\..\MainWindow.xaml"
00228 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00229     "CA1823:AvoidUnusedPrivateFields")]
00230     internal System.Windows.Controls.TextBlock CarryFlagText;
00231 #line default
00232 #line hidden
00233
00234
00235 #line 156 "..\..\..\MainWindow.xaml"
00236 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00237     "CA1823:AvoidUnusedPrivateFields")]
00238     internal System.Windows.Controls.CheckBox ZeroFlag;
00239 #line default
00240 #line hidden
00241
00242
00243 #line 157 "..\..\..\MainWindow.xaml"
00244 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00245     "CA1823:AvoidUnusedPrivateFields")]
00246     internal System.Windows.Controls.TextBlock ZeroFlagText;
00247 #line default
00248 #line hidden
00249
00250
00251 #line 158 "..\..\..\MainWindow.xaml"
00252 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00253     "CA1823:AvoidUnusedPrivateFields")]
00254     internal System.Windows.Controls.CheckBox InterrupFlag;
```

```
00254
00255 #line default
00256 #line hidden
00257
00258
00259 #line 159 "..\..\..\MainWindow.xaml"
00260 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00261 "CA1823:AvoidUnusedPrivateFields")]
00262     internal System.Windows.Controls.TextBlock InterruptFlagText;
00263
00264 #line default
00265 #line hidden
00266
00267 #line 160 "..\..\..\MainWindow.xaml"
00268 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00269 "CA1823:AvoidUnusedPrivateFields")]
00270     internal System.Windows.Controls.CheckBox BcdFlag;
00271
00272 #line default
00273 #line hidden
00274
00275 #line 161 "..\..\..\MainWindow.xaml"
00276 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00277 "CA1823:AvoidUnusedPrivateFields")]
00278     internal System.Windows.Controls.TextBlock BcdFlagText;
00279
00280 #line default
00281 #line hidden
00282
00283 #line 162 "..\..\..\MainWindow.xaml"
00284 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00285 "CA1823:AvoidUnusedPrivateFields")]
00286     internal System.Windows.Controls.CheckBox BreakFlag;
00287
00288 #line default
00289 #line hidden
00290
00291 #line 163 "..\..\..\MainWindow.xaml"
00292 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00293 "CA1823:AvoidUnusedPrivateFields")]
00294     internal System.Windows.Controls.TextBlock BreakFlagText;
00295
00296 #line default
00297 #line hidden
00298
00299 #line 164 "..\..\..\MainWindow.xaml"
00300 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00301 "CA1823:AvoidUnusedPrivateFields")]
00302     internal System.Windows.Controls.CheckBox OverflowFlag;
00303
00304 #line default
00305 #line hidden
00306
00307 #line 165 "..\..\..\MainWindow.xaml"
00308 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00309 "CA1823:AvoidUnusedPrivateFields")]
00310     internal System.Windows.Controls.TextBlock OverflowFlagText;
00311
00312 #line default
00313 #line hidden
00314
00315 #line 166 "..\..\..\MainWindow.xaml"
00316 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00317 "CA1823:AvoidUnusedPrivateFields")]
00318     internal System.Windows.Controls.CheckBox NegativeFlag;
00319
00320 #line default
00321 #line hidden
00322
00323 #line 167 "..\..\..\MainWindow.xaml"
00324 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00325 "CA1823:AvoidUnusedPrivateFields")]
00326     internal System.Windows.Controls.TextBlock NegativeFlagText;
00327
00328 #line default
00329 #line hidden
00330
00331 #line 168 "..\..\..\MainWindow.xaml"
```

```

00332         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00333         "CA1823:AvoidUnusedPrivateFields")]
00333         internal System.Windows.Controls.Slider CpuSpeed;
00334
00335 #line default
00336 #line hidden
00337
00338
00339 #line 169 "..\..\..\MainWindow.xaml"
00340         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00341         "CA1823:AvoidUnusedPrivateFields")]
00341         internal System.Windows.Controls.TextBlock SpeedText;
00342
00343 #line default
00344 #line hidden
00345
00346         private bool _contentLoaded;
00347
00348 /// <summary>
00349 /// InitializeComponent
00350 /// </summary>
00351         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00352         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00353         public void InitializeComponent() {
00354             if (_contentLoaded) {
00355                 return;
00356             }
00357             _contentLoaded = true;
00358             System.Uri resourceLocater = new System.Uri("/Emulator;component/mainwindow.xaml",
00359             System.UriKind.Relative);
00360 #line 1 "..\..\..\MainWindow.xaml"
00361             System.Windows.Application.LoadComponent(this, resourceLocater);
00362
00363 #line default
00364 #line hidden
00365         }
00366
00367         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00368         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00369
00370         [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00370         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
00371         "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00371         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
00372         "CA1502:AvoidExcessiveComplexity")]
00372         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00373         "CA1800:DoNotCastUnnecessarily")]
00373         void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00374             switch (connectionId)
00375             {
00376                 case 1:
00377                     this.EmulatorWindow = ((Emulator.MainWindow)(target));
00378                     return;
00379                 case 2:
00380
00381 #line 72 "..\..\..\MainWindow.xaml"
00382                     ((System.Windows.Controls.MenuItem)(target)).Click += new
00383                     System.Windows.RoutedEventHandler(this.LoadFile);
00384 #line default
00385 #line hidden
00386                     return;
00387                 case 3:
00388
00389 #line 73 "..\..\..\MainWindow.xaml"
00390                     ((System.Windows.Controls.MenuItem)(target)).Click += new
00391                     System.Windows.RoutedEventHandler(this.SaveFile);
00392 #line default
00393 #line hidden
00394                     return;
00395                 case 4:
00396
00397 #line 74 "..\..\..\MainWindow.xaml"
00398                     ((System.Windows.Controls.MenuItem)(target)).Click += new
00399                     System.Windows.RoutedEventHandler(this.CloseFile);
00400 #line default
00401 #line hidden
00402                     return;
00403                 case 5:
00404
00405 #line 76 "..\..\..\MainWindow.xaml"
00406                     ((System.Windows.Controls.MenuItem)(target)).Click += new
00407                     System.Windows.RoutedEventHandler(this.ToClose);

```

```
00408 #line default
00409 #line hidden
00410         return;
00411         case 6:
00412             this.OutputLog = ((System.Windows.Controls.DataGrid) (target));
00413             return;
00414         case 7:
00415             this.Run = ((System.Windows.Controls.Button) (target));
00416             return;
00417         case 8:
00418             this.Step = ((System.Windows.Controls.Button) (target));
00419             return;
00420         case 9:
00421             this.Reset = ((System.Windows.Controls.Button) (target));
00422             return;
00423         case 10:
00424             this.RomFileNameText = ((System.Windows.Controls.TextBlock) (target));
00425             return;
00426         case 11:
00427             this.ComPortNameText = ((System.Windows.Controls.TextBlock) (target));
00428             return;
00429         case 12:
00430             this.Breakpoints = ((System.Windows.Controls.DataGrid) (target));
00431             return;
00432         case 13:
00433             this.YRegister = ((System.Windows.Controls.TextBox) (target));
00434             return;
00435         case 14:
00436             this.XRegister = ((System.Windows.Controls.TextBox) (target));
00437             return;
00438         case 15:
00439             this.Accumulator = ((System.Windows.Controls.TextBox) (target));
00440             return;
00441         case 16:
00442             this.StackPointer = ((System.Windows.Controls.TextBox) (target));
00443             return;
00444         case 17:
00445             this.ProgramCounter = ((System.Windows.Controls.TextBox) (target));
00446             return;
00447         case 18:
00448             this.Dissambly = ((System.Windows.Controls.TextBox) (target));
00449             return;
00450         case 19:
00451             this.CycleCount = ((System.Windows.Controls.TextBox) (target));
00452             return;
00453         case 20:
00454             this.XRegisterText = ((System.Windows.Controls.TextBlock) (target));
00455             return;
00456         case 21:
00457             this.YRegisterText = ((System.Windows.Controls.TextBlock) (target));
00458             return;
00459         case 22:
00460             this.StackPointerRegisterText = ((System.Windows.Controls.TextBlock) (target));
00461             return;
00462         case 23:
00463             this.AText = ((System.Windows.Controls.TextBlock) (target));
00464             return;
00465         case 24:
00466             this.CurrentInstructionText = ((System.Windows.Controls.TextBlock) (target));
00467             return;
00468         case 25:
00469             this.ProgramCounterText = ((System.Windows.Controls.TextBlock) (target));
00470             return;
00471         case 26:
00472             this.CycleCountText = ((System.Windows.Controls.TextBlock) (target));
00473             return;
00474         case 27:
00475             this.CarryFlag = ((System.Windows.Controls.CheckBox) (target));
00476             return;
00477         case 28:
00478             this.CarryFlagText = ((System.Windows.Controls.TextBlock) (target));
00479             return;
00480         case 29:
00481             this.ZeroFlag = ((System.Windows.Controls.CheckBox) (target));
00482             return;
00483         case 30:
00484             this.ZeroFlagText = ((System.Windows.Controls.TextBlock) (target));
00485             return;
00486         case 31:
00487             this.InterruptFlag = ((System.Windows.Controls.CheckBox) (target));
00488             return;
00489         case 32:
00490             this.InterruptFlagText = ((System.Windows.Controls.TextBlock) (target));
00491             return;
00492         case 33:
00493             this.BcdFlag = ((System.Windows.Controls.CheckBox) (target));
00494             return;
```

```

00495         case 34:
00496             this.BcdFlagText = ((System.Windows.Controls.TextBlock)(target));
00497             return;
00498         case 35:
00499             this.BreakFlag = ((System.Windows.Controls.CheckBox)(target));
00500             return;
00501         case 36:
00502             this.BreakFlagText = ((System.Windows.Controls.TextBlock)(target));
00503             return;
00504         case 37:
00505             this.OverflowFlag = ((System.Windows.Controls.CheckBox)(target));
00506             return;
00507         case 38:
00508             this.OverflowFlagText = ((System.Windows.Controls.TextBlock)(target));
00509             return;
00510         case 39:
00511             this.NegativeFlag = ((System.Windows.Controls.CheckBox)(target));
00512             return;
00513         case 40:
00514             this.NegativeFlagText = ((System.Windows.Controls.TextBlock)(target));
00515             return;
00516         case 41:
00517             this.CpuSpeed = ((System.Windows.Controls.Slider)(target));
00518             return;
00519         case 42:
00520             this.SpeedText = ((System.Windows.Controls.TextBlock)(target));
00521             return;
00522     }
00523     this._contentLoaded = true;
00524 }
00525 }
00526 }
00527

```

7.93 Emulator/obj/x86/Publish/MainWindow.g.i.cs File Reference

Classes

- class [Emulator.MainWindow](#)
Interaction logic for MainWindow.xaml

Namespaces

- namespace [Emulator](#)

7.94 MainWindow.g.i.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\MainWindow.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00002 "B80FD745A4A855A2770EA0E1513AC1103AE202406DDCEC01FD8CA1DB0293F06C"
00003 //-----
00004 // <auto-generated>
00005 //     This code was generated by a tool.
00006 //     Runtime Version:4.0.30319.42000
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;

```

```

00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037 /// <summary>
00038 /// MainWindow
00039 /// </summary>
00040 public partial class MainWindow : System.Windows.Window,
System.Windows.Markup.IComponentConnector {
00041
00042
00043 #line 2 "..\..\..\Main\MainWindow.xaml"
00044 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00045     internal Emulator.MainWindow EmulatorWindow;
00046
00047 #line default
00048 #line hidden
00049
00050
00051 #line 89 "..\..\..\Main\MainWindow.xaml"
00052 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00053     internal System.Windows.Controls.DataGrid OutputLog;
00054
00055 #line default
00056 #line hidden
00057
00058
00059 #line 106 "..\..\..\Main\MainWindow.xaml"
00060 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00061     internal System.Windows.Controls.Button Run;
00062
00063 #line default
00064 #line hidden
00065
00066
00067 #line 107 "..\..\..\Main\MainWindow.xaml"
00068 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00069     internal System.Windows.Controls.Button Step;
00070
00071 #line default
00072 #line hidden
00073
00074
00075 #line 108 "..\..\..\Main\MainWindow.xaml"
00076 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00077     internal System.Windows.Controls.Button Reset;
00078
00079 #line default
00080 #line hidden
00081
00082
00083 #line 110 "..\..\..\Main\MainWindow.xaml"
00084 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00085     internal System.Windows.Controls.TextBlock RomFileNameText;
00086
00087 #line default
00088 #line hidden
00089
00090
00091 #line 111 "..\..\..\Main\MainWindow.xaml"
00092 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00093     internal System.Windows.Controls.TextBlock ComPortNameText;
00094
00095 #line default
00096 #line hidden
00097
00098
00099 #line 112 "..\..\..\Main\MainWindow.xaml"
00100 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00101     internal System.Windows.Controls.DataGrid Breakpoints;

```

```

00102
00103 #line default
00104 #line hidden
00105
00106
00107 #line 137 "..\..\..\MainWindow.xaml"
00108 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00109     internal System.Windows.Controls.TextBox YRegister;
00110
00111 #line default
00112 #line hidden
00113
00114
00115 #line 138 "..\..\..\MainWindow.xaml"
00116 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00117     internal System.Windows.Controls.TextBox XRegister;
00118
00119 #line default
00120 #line hidden
00121
00122
00123 #line 139 "..\..\..\MainWindow.xaml"
00124 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00125     internal System.Windows.Controls.TextBox Accumulator;
00126
00127 #line default
00128 #line hidden
00129
00130
00131 #line 140 "..\..\..\MainWindow.xaml"
00132 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00133     internal System.Windows.Controls.TextBox StackPointer;
00134
00135 #line default
00136 #line hidden
00137
00138
00139 #line 141 "..\..\..\MainWindow.xaml"
00140 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00141     internal System.Windows.Controls.TextBox ProgramCounter;
00142
00143 #line default
00144 #line hidden
00145
00146
00147 #line 142 "..\..\..\MainWindow.xaml"
00148 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00149     internal System.Windows.Controls.TextBox Dissambly;
00150
00151 #line default
00152 #line hidden
00153
00154
00155 #line 143 "..\..\..\MainWindow.xaml"
00156 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00157     internal System.Windows.Controls.TextBox CycleCount;
00158
00159 #line default
00160 #line hidden
00161
00162
00163 #line 144 "..\..\..\MainWindow.xaml"
00164 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00165     internal System.Windows.Controls.TextBlock XRegisterText;
00166
00167 #line default
00168 #line hidden
00169
00170
00171 #line 145 "..\..\..\MainWindow.xaml"
00172 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00173     internal System.Windows.Controls.TextBlock YRegisterText;
00174
00175 #line default
00176 #line hidden
00177
00178
00179 #line 146 "..\..\..\MainWindow.xaml"

```

```
00180         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00181         "CA1823:AvoidUnusedPrivateFields")]
00182         internal System.Windows.Controls.TextBlock StackPointerRegisterText;
00183 #line default
00184 #line hidden
00185
00186
00187 #line 147 "..\..\..\MainWindows\MainWindow.xaml"
00188         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00189         "CA1823:AvoidUnusedPrivateFields")]
00190         internal System.Windows.Controls.TextBlock AText;
00191 #line default
00192 #line hidden
00193
00194
00195 #line 148 "..\..\..\MainWindows\MainWindow.xaml"
00196         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00197         "CA1823:AvoidUnusedPrivateFields")]
00198         internal System.Windows.Controls.TextBlock CurrentInstructionText;
00199 #line default
00200 #line hidden
00201
00202
00203 #line 149 "..\..\..\MainWindows\MainWindow.xaml"
00204         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00205         "CA1823:AvoidUnusedPrivateFields")]
00206         internal System.Windows.Controls.TextBlock ProgramCounterText;
00207 #line default
00208 #line hidden
00209
00210
00211 #line 150 "..\..\..\MainWindows\MainWindow.xaml"
00212         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00213         "CA1823:AvoidUnusedPrivateFields")]
00214         internal System.Windows.Controls.TextBlock CycleCountText;
00215 #line default
00216 #line hidden
00217
00218
00219 #line 151 "..\..\..\MainWindows\MainWindow.xaml"
00220         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00221         "CA1823:AvoidUnusedPrivateFields")]
00222         internal System.Windows.Controls.CheckBox CarryFlag;
00223 #line default
00224 #line hidden
00225
00226
00227 #line 152 "..\..\..\MainWindows\MainWindow.xaml"
00228         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00229         "CA1823:AvoidUnusedPrivateFields")]
00230         internal System.Windows.Controls.TextBlock CarryFlagText;
00231 #line default
00232 #line hidden
00233
00234
00235 #line 153 "..\..\..\MainWindows\MainWindow.xaml"
00236         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00237         "CA1823:AvoidUnusedPrivateFields")]
00238         internal System.Windows.Controls.CheckBox ZeroFlag;
00239 #line default
00240 #line hidden
00241
00242
00243 #line 154 "..\..\..\MainWindows\MainWindow.xaml"
00244         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00245         "CA1823:AvoidUnusedPrivateFields")]
00246         internal System.Windows.Controls.TextBlock ZeroFlagText;
00247 #line default
00248 #line hidden
00249
00250
00251 #line 155 "..\..\..\MainWindows\MainWindow.xaml"
00252         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00253         "CA1823:AvoidUnusedPrivateFields")]
00254         internal System.Windows.Controls.CheckBox InterrupFlag;
00255 #line default
00256 #line hidden
```



```

00257
00258
00259 #line 156 "..\..\..\MainWindow.xaml"
00260     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00261         "CA1823:AvoidUnusedPrivateFields")]
00262         internal System.Windows.Controls.TextBlock InterruptFlagText;
00263 #line default
00264 #line hidden
00265
00266
00267 #line 157 "..\..\..\MainWindow.xaml"
00268     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00269         "CA1823:AvoidUnusedPrivateFields")]
00270         internal System.Windows.Controls.CheckBox BcdFlag;
00271 #line default
00272 #line hidden
00273
00274
00275 #line 158 "..\..\..\MainWindow.xaml"
00276     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00277         "CA1823:AvoidUnusedPrivateFields")]
00278         internal System.Windows.Controls.TextBlock BcdFlagText;
00279 #line default
00280 #line hidden
00281
00282
00283 #line 159 "..\..\..\MainWindow.xaml"
00284     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00285         "CA1823:AvoidUnusedPrivateFields")]
00286         internal System.Windows.Controls.CheckBox BreakFlag;
00287 #line default
00288 #line hidden
00289
00290
00291 #line 160 "..\..\..\MainWindow.xaml"
00292     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00293         "CA1823:AvoidUnusedPrivateFields")]
00294         internal System.Windows.Controls.TextBlock BreakFlagText;
00295 #line default
00296 #line hidden
00297
00298
00299 #line 161 "..\..\..\MainWindow.xaml"
00300     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00301         "CA1823:AvoidUnusedPrivateFields")]
00302         internal System.Windows.Controls.CheckBox OverflowFlag;
00303 #line default
00304 #line hidden
00305
00306
00307 #line 162 "..\..\..\MainWindow.xaml"
00308     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00309         "CA1823:AvoidUnusedPrivateFields")]
00310         internal System.Windows.Controls.TextBlock OverflowFlagText;
00311 #line default
00312 #line hidden
00313
00314
00315 #line 163 "..\..\..\MainWindow.xaml"
00316     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00317         "CA1823:AvoidUnusedPrivateFields")]
00318         internal System.Windows.Controls.CheckBox NegativeFlag;
00319 #line default
00320 #line hidden
00321
00322
00323 #line 164 "..\..\..\MainWindow.xaml"
00324     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00325         "CA1823:AvoidUnusedPrivateFields")]
00326         internal System.Windows.Controls.TextBlock NegativeFlagText;
00327 #line default
00328 #line hidden
00329
00330
00331 #line 165 "..\..\..\MainWindow.xaml"
00332     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00333         "CA1823:AvoidUnusedPrivateFields")]
00334         internal System.Windows.Controls.Slider CpuSpeed;

```

```

00334
00335 #line default
00336 #line hidden
00337
00338
00339 #line 166 "..\..\..\MainWindows.Xaml"
00340 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
    "CA1823:AvoidUnusedPrivateFields")]
00341     internal System.Windows.Controls.TextBlock SpeedText;
00342
00343 #line default
00344 #line hidden
00345
00346     private bool _contentLoaded;
00347
00348 /// <summary>
00349 /// InitializeComponent
00350 /// </summary>
00351 [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00352 [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00353 public void InitializeComponent() {
00354     if (_contentLoaded) {
00355         return;
00356     }
00357     _contentLoaded = true;
00358     System.Uri resourceLocater = new System.Uri("/Emulator;component/mainwindow.xaml",
        System.UriKind.Relative);
00359
00360 #line 1 "..\..\..\MainWindows.Xaml"
00361     System.Windows.Application.LoadComponent(this, resourceLocater);
00362
00363 #line default
00364 #line hidden
00365 }
00366
00367 [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00368 [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00369
00370 [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00371 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
    "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00372 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
    "CA1502:AvoidExcessiveComplexity")]
00373 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
    "CA1800:DoNotCastUnnecessarily")]
00374 void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00375     switch (connectionId)
00376     {
00377         case 1:
00378             this.EmulatorWindow = ((Emulator.MainWindow)(target));
00379             return;
00380         case 2:
00381 #line 72 "..\..\..\MainWindows.Xaml"
00382             ((System.Windows.Controls.MenuItem)(target)).Click += new
        System.Windows.RoutedEventHandler(this.LoadFile);
00383
00384 #line default
00385 #line hidden
00386             return;
00387         case 3:
00388 #line 73 "..\..\..\MainWindows.Xaml"
00389             ((System.Windows.Controls.MenuItem)(target)).Click += new
        System.Windows.RoutedEventHandler(this.SaveFile);
00390
00391 #line default
00392 #line hidden
00393             return;
00394         case 4:
00395 #line 74 "..\..\..\MainWindows.Xaml"
00396             ((System.Windows.Controls.MenuItem)(target)).Click += new
        System.Windows.RoutedEventHandler(this.CloseFile);
00397
00398 #line default
00399 #line hidden
00400             return;
00401         case 5:
00402 #line 76 "..\..\..\MainWindows.Xaml"
00403             ((System.Windows.Controls.MenuItem)(target)).Click += new
        System.Windows.RoutedEventHandler(this.ToClose);
00404
00405 #line default
00406 #line hidden
00407             return;
00408 #line default
00409 #line hidden
00410     }
00411 }

```

```
00411         case 6:
00412             this.OutputLog = ((System.Windows.Controls.DataGrid) (target));
00413             return;
00414         case 7:
00415             this.Run = ((System.Windows.Controls.Button) (target));
00416             return;
00417         case 8:
00418             this.Step = ((System.Windows.Controls.Button) (target));
00419             return;
00420         case 9:
00421             this.Reset = ((System.Windows.Controls.Button) (target));
00422             return;
00423         case 10:
00424             this.RomFileNameText = ((System.Windows.Controls.TextBlock) (target));
00425             return;
00426         case 11:
00427             this.ComPortNameText = ((System.Windows.Controls.TextBlock) (target));
00428             return;
00429         case 12:
00430             this.Breakpoints = ((System.Windows.Controls.DataGrid) (target));
00431             return;
00432         case 13:
00433             this.YRegister = ((System.Windows.Controls.TextBox) (target));
00434             return;
00435         case 14:
00436             this.XRegister = ((System.Windows.Controls.TextBox) (target));
00437             return;
00438         case 15:
00439             this.Accumulator = ((System.Windows.Controls.TextBox) (target));
00440             return;
00441         case 16:
00442             this.StackPointer = ((System.Windows.Controls.TextBox) (target));
00443             return;
00444         case 17:
00445             this.ProgramCounter = ((System.Windows.Controls.TextBox) (target));
00446             return;
00447         case 18:
00448             this.Dissassembly = ((System.Windows.Controls.TextBox) (target));
00449             return;
00450         case 19:
00451             this.CycleCount = ((System.Windows.Controls.TextBox) (target));
00452             return;
00453         case 20:
00454             this.XRegisterText = ((System.Windows.Controls.TextBlock) (target));
00455             return;
00456         case 21:
00457             this.YRegisterText = ((System.Windows.Controls.TextBlock) (target));
00458             return;
00459         case 22:
00460             this.StackPointerRegisterText = ((System.Windows.Controls.TextBlock) (target));
00461             return;
00462         case 23:
00463             this.AText = ((System.Windows.Controls.TextBlock) (target));
00464             return;
00465         case 24:
00466             this.CurrentInstructionText = ((System.Windows.Controls.TextBlock) (target));
00467             return;
00468         case 25:
00469             this.ProgramCounterText = ((System.Windows.Controls.TextBlock) (target));
00470             return;
00471         case 26:
00472             this.CycleCountText = ((System.Windows.Controls.TextBlock) (target));
00473             return;
00474         case 27:
00475             this.CarryFlag = ((System.Windows.Controls.CheckBox) (target));
00476             return;
00477         case 28:
00478             this.CarryFlagText = ((System.Windows.Controls.TextBlock) (target));
00479             return;
00480         case 29:
00481             this.ZeroFlag = ((System.Windows.Controls.CheckBox) (target));
00482             return;
00483         case 30:
00484             this.ZeroFlagText = ((System.Windows.Controls.TextBlock) (target));
00485             return;
00486         case 31:
00487             this.InterruptFlag = ((System.Windows.Controls.CheckBox) (target));
00488             return;
00489         case 32:
00490             this.InterruptFlagText = ((System.Windows.Controls.TextBlock) (target));
00491             return;
00492         case 33:
00493             this.BcdFlag = ((System.Windows.Controls.CheckBox) (target));
00494             return;
00495         case 34:
00496             this.BcdFlagText = ((System.Windows.Controls.TextBlock) (target));
00497             return;
```

```

00498         case 35:
00499             this.BreakFlag = ((System.Windows.Controls.CheckBox) (target));
00500             return;
00501         case 36:
00502             this.BreakFlagText = ((System.Windows.Controls.TextBlock) (target));
00503             return;
00504         case 37:
00505             this.OverflowFlag = ((System.Windows.Controls.CheckBox) (target));
00506             return;
00507         case 38:
00508             this.OverflowFlagText = ((System.Windows.Controls.TextBlock) (target));
00509             return;
00510         case 39:
00511             this.NegativeFlag = ((System.Windows.Controls.CheckBox) (target));
00512             return;
00513         case 40:
00514             this.NegativeFlagText = ((System.Windows.Controls.TextBlock) (target));
00515             return;
00516         case 41:
00517             this.CpuSpeed = ((System.Windows.Controls.Slider) (target));
00518             return;
00519         case 42:
00520             this.SpeedText = ((System.Windows.Controls.TextBlock) (target));
00521             return;
00522     }
00523     this._contentLoaded = true;
00524 }
00525 }
00526 }
00527

```

7.95 Emulator/obj/x86/Release/MainWindow.g.i.cs File Reference

Classes

- class [Emulator.MainWindow](#)
Interaction logic for MainWindow.xaml

Namespaces

- namespace [Emulator](#)

7.96 MainWindow.g.i.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\MainWindow.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00002 "1FB39AF98423D8DD6333B173E814398E2016BACFE25491D8BD824F3F8A79E0A5"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;

```

```

00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037 /// <summary>
00038 /// MainWindow
00039 /// </summary>
00040 public partial class MainWindow : System.Windows.Window,
System.Windows.Markup.IComponentConnector {
00041
00042
00043 #line 2 "..\..\..\MainWindows.xml"
00044 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00045 internal Emulator.MainWindow EmulatorWindow;
00046
00047 #line default
00048 #line hidden
00049
00050
00051 #line 92 "..\..\..\MainWindows.xml"
00052 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00053 internal System.Windows.Controls.DataGrid OutputLog;
00054
00055 #line default
00056 #line hidden
00057
00058
00059 #line 109 "..\..\..\MainWindows.xml"
00060 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00061 internal System.Windows.Controls.Button Run;
00062
00063 #line default
00064 #line hidden
00065
00066
00067 #line 110 "..\..\..\MainWindows.xml"
00068 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00069 internal System.Windows.Controls.Button Step;
00070
00071 #line default
00072 #line hidden
00073
00074
00075 #line 111 "..\..\..\MainWindows.xml"
00076 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00077 internal System.Windows.Controls.Button Reset;
00078
00079 #line default
00080 #line hidden
00081
00082
00083 #line 113 "..\..\..\MainWindows.xml"
00084 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00085 internal System.Windows.Controls.TextBlock RomFileNameText;
00086
00087 #line default
00088 #line hidden
00089
00090
00091 #line 114 "..\..\..\MainWindows.xml"
00092 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00093 internal System.Windows.Controls.TextBlock ComPortNameText;
00094
00095 #line default
00096 #line hidden
00097
00098
00099 #line 115 "..\..\..\MainWindows.xml"
00100 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00101 internal System.Windows.Controls.DataGrid Breakpoints;
00102
00103 #line default
00104 #line hidden

```

```
00105
00106
00107 #line 140 "..\..\..\MainWindow.xaml"
00108 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00109 "CA1823:AvoidUnusedPrivateFields")]
00109     internal System.Windows.Controls.TextBox YRegister;
00110
00111 #line default
00112 #line hidden
00113
00114
00115 #line 141 "..\..\..\MainWindow.xaml"
00116 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00117 "CA1823:AvoidUnusedPrivateFields")]
00117     internal System.Windows.Controls.TextBox XRegister;
00118
00119 #line default
00120 #line hidden
00121
00122
00123 #line 142 "..\..\..\MainWindow.xaml"
00124 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00125 "CA1823:AvoidUnusedPrivateFields")]
00125     internal System.Windows.Controls.TextBox Accumulator;
00126
00127 #line default
00128 #line hidden
00129
00130
00131 #line 143 "..\..\..\MainWindow.xaml"
00132 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00133 "CA1823:AvoidUnusedPrivateFields")]
00133     internal System.Windows.Controls.TextBox StackPointer;
00134
00135 #line default
00136 #line hidden
00137
00138
00139 #line 144 "..\..\..\MainWindow.xaml"
00140 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00141 "CA1823:AvoidUnusedPrivateFields")]
00141     internal System.Windows.Controls.TextBox ProgramCounter;
00142
00143 #line default
00144 #line hidden
00145
00146
00147 #line 145 "..\..\..\MainWindow.xaml"
00148 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00149 "CA1823:AvoidUnusedPrivateFields")]
00149     internal System.Windows.Controls.TextBox Dissambly;
00150
00151 #line default
00152 #line hidden
00153
00154
00155 #line 146 "..\..\..\MainWindow.xaml"
00156 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00157 "CA1823:AvoidUnusedPrivateFields")]
00157     internal System.Windows.Controls.TextBox CycleCount;
00158
00159 #line default
00160 #line hidden
00161
00162
00163 #line 147 "..\..\..\MainWindow.xaml"
00164 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00165 "CA1823:AvoidUnusedPrivateFields")]
00165     internal System.Windows.Controls.TextBlock XRegisterText;
00166
00167 #line default
00168 #line hidden
00169
00170
00171 #line 148 "..\..\..\MainWindow.xaml"
00172 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00173 "CA1823:AvoidUnusedPrivateFields")]
00173     internal System.Windows.Controls.TextBlock YRegisterText;
00174
00175 #line default
00176 #line hidden
00177
00178
00179 #line 149 "..\..\..\MainWindow.xaml"
00180 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00181 "CA1823:AvoidUnusedPrivateFields")]
00181     internal System.Windows.Controls.TextBlock StackPointerRegisterText;
```

```
00182
00183 #line default
00184 #line hidden
00185
00186
00187 #line 150 "..\..\..\MainWindow.xaml"
00188 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00189 "CA1823:AvoidUnusedPrivateFields")]
00189     internal System.Windows.Controls.TextBlock AText;
00190
00191 #line default
00192 #line hidden
00193
00194
00195 #line 151 "..\..\..\MainWindow.xaml"
00196 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00197 "CA1823:AvoidUnusedPrivateFields")]
00197     internal System.Windows.Controls.TextBlock CurrentInstructionText;
00198
00199 #line default
00200 #line hidden
00201
00202
00203 #line 152 "..\..\..\MainWindow.xaml"
00204 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00205 "CA1823:AvoidUnusedPrivateFields")]
00205     internal System.Windows.Controls.TextBlock ProgramCounterText;
00206
00207 #line default
00208 #line hidden
00209
00210
00211 #line 153 "..\..\..\MainWindow.xaml"
00212 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00213 "CA1823:AvoidUnusedPrivateFields")]
00213     internal System.Windows.Controls.TextBlock CycleCountText;
00214
00215 #line default
00216 #line hidden
00217
00218
00219 #line 154 "..\..\..\MainWindow.xaml"
00220 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00221 "CA1823:AvoidUnusedPrivateFields")]
00221     internal System.Windows.Controls.CheckBox CarryFlag;
00222
00223 #line default
00224 #line hidden
00225
00226
00227 #line 155 "..\..\..\MainWindow.xaml"
00228 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00229 "CA1823:AvoidUnusedPrivateFields")]
00229     internal System.Windows.Controls.TextBlock CarryFlagText;
00230
00231 #line default
00232 #line hidden
00233
00234
00235 #line 156 "..\..\..\MainWindow.xaml"
00236 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00237 "CA1823:AvoidUnusedPrivateFields")]
00237     internal System.Windows.Controls.CheckBox ZeroFlag;
00238
00239 #line default
00240 #line hidden
00241
00242
00243 #line 157 "..\..\..\MainWindow.xaml"
00244 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00245 "CA1823:AvoidUnusedPrivateFields")]
00245     internal System.Windows.Controls.TextBlock ZeroFlagText;
00246
00247 #line default
00248 #line hidden
00249
00250
00251 #line 158 "..\..\..\MainWindow.xaml"
00252 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00253 "CA1823:AvoidUnusedPrivateFields")]
00253     internal System.Windows.Controls.CheckBox InterrupFlag;
00254
00255 #line default
00256 #line hidden
00257
00258
00259 #line 159 "..\..\..\MainWindow.xaml"
```

```
00260         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00261         "CA1823:AvoidUnusedPrivateFields")]
00262         internal System.Windows.Controls.TextBlock InterruptFlagText;
00263 #line default
00264 #line hidden
00265
00266
00267 #line 160 "..\..\..\MainWindows\MainWindow.xaml"
00268         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00269         "CA1823:AvoidUnusedPrivateFields")]
00270         internal System.Windows.Controls.CheckBox BcdFlag;
00271 #line default
00272 #line hidden
00273
00274
00275 #line 161 "..\..\..\MainWindows\MainWindow.xaml"
00276         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00277         "CA1823:AvoidUnusedPrivateFields")]
00278         internal System.Windows.Controls.TextBlock BcdFlagText;
00279 #line default
00280 #line hidden
00281
00282
00283 #line 162 "..\..\..\MainWindows\MainWindow.xaml"
00284         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00285         "CA1823:AvoidUnusedPrivateFields")]
00286         internal System.Windows.Controls.CheckBox BreakFlag;
00287 #line default
00288 #line hidden
00289
00290
00291 #line 163 "..\..\..\MainWindows\MainWindow.xaml"
00292         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00293         "CA1823:AvoidUnusedPrivateFields")]
00294         internal System.Windows.Controls.TextBlock BreakFlagText;
00295 #line default
00296 #line hidden
00297
00298
00299 #line 164 "..\..\..\MainWindows\MainWindow.xaml"
00300         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00301         "CA1823:AvoidUnusedPrivateFields")]
00302         internal System.Windows.Controls.CheckBox OverflowFlag;
00303 #line default
00304 #line hidden
00305
00306
00307 #line 165 "..\..\..\MainWindows\MainWindow.xaml"
00308         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00309         "CA1823:AvoidUnusedPrivateFields")]
00310         internal System.Windows.Controls.TextBlock OverflowFlagText;
00311 #line default
00312 #line hidden
00313
00314
00315 #line 166 "..\..\..\MainWindows\MainWindow.xaml"
00316         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00317         "CA1823:AvoidUnusedPrivateFields")]
00318         internal System.Windows.Controls.CheckBox NegativeFlag;
00319 #line default
00320 #line hidden
00321
00322
00323 #line 167 "..\..\..\MainWindows\MainWindow.xaml"
00324         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00325         "CA1823:AvoidUnusedPrivateFields")]
00326         internal System.Windows.Controls.TextBlock NegativeFlagText;
00327 #line default
00328 #line hidden
00329
00330
00331 #line 168 "..\..\..\MainWindows\MainWindow.xaml"
00332         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00333         "CA1823:AvoidUnusedPrivateFields")]
00334         internal System.Windows.Controls.Slider CpuSpeed;
00335 #line default
00336 #line hidden
```



```

00337
00338
00339 #line 169 "..\..\..\MainWindow.xaml"
00340 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
    "CA1823:AvoidUnusedPrivateFields")]
00341     internal System.Windows.Controls.TextBlock SpeedText;
00342
00343 #line default
00344 #line hidden
00345
00346     private bool _contentLoaded;
00347
00348 /// <summary>
00349 /// InitializeComponent
00350 /// </summary>
00351 [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00352 [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00353 public void InitializeComponent() {
00354     if (_contentLoaded) {
00355         return;
00356     }
00357     _contentLoaded = true;
00358     System.Uri resourceLocater = new System.Uri("/Emulator;component/mainwindow.xaml",
        System.UriKind.Relative);
00359
00360 #line 1 "..\..\..\MainWindow.xaml"
00361     System.Windows.Application.LoadComponent(this, resourceLocater);
00362
00363 #line default
00364 #line hidden
00365 }
00366
00367 [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00368 [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00369 [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00370 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
    "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00371 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
    "CA1502:AvoidExcessiveComplexity")]
00372 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
    "CA1800:DoNotCastUnnecessarily")]
00373 void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00374     switch (connectionId)
00375     {
00376     case 1:
00377         this.EmulatorWindow = ((Emulator.MainWindow)(target));
00378         return;
00379     case 2:
00380
00381 #line 72 "..\..\..\MainWindow.xaml"
00382         ((System.Windows.Controls.MenuItem)(target)).Click += new
            System.Windows.RoutedEventHandler(this.LoadFile);
00383
00384 #line default
00385 #line hidden
00386         return;
00387     case 3:
00388
00389 #line 73 "..\..\..\MainWindow.xaml"
00390         ((System.Windows.Controls.MenuItem)(target)).Click += new
            System.Windows.RoutedEventHandler(this.SaveFile);
00391
00392 #line default
00393 #line hidden
00394         return;
00395     case 4:
00396
00397 #line 74 "..\..\..\MainWindow.xaml"
00398         ((System.Windows.Controls.MenuItem)(target)).Click += new
            System.Windows.RoutedEventHandler(this.CloseFile);
00399
00400 #line default
00401 #line hidden
00402         return;
00403     case 5:
00404
00405 #line 76 "..\..\..\MainWindow.xaml"
00406         ((System.Windows.Controls.MenuItem)(target)).Click += new
            System.Windows.RoutedEventHandler(this.ToClose);
00407
00408 #line default
00409 #line hidden
00410         return;
00411     case 6:
00412         this.OutputLog = ((System.Windows.Controls.DataGrid)(target));
00413         return;

```

```
00414         case 7:
00415             this.Run = ((System.Windows.Controls.Button) (target));
00416             return;
00417         case 8:
00418             this.Step = ((System.Windows.Controls.Button) (target));
00419             return;
00420         case 9:
00421             this.Reset = ((System.Windows.Controls.Button) (target));
00422             return;
00423         case 10:
00424             this.RomFileNameText = ((System.Windows.Controls.TextBlock) (target));
00425             return;
00426         case 11:
00427             this.ComPortNameText = ((System.Windows.Controls.TextBlock) (target));
00428             return;
00429         case 12:
00430             this.Breakpoints = ((System.Windows.Controls.DataGrid) (target));
00431             return;
00432         case 13:
00433             this.YRegister = ((System.Windows.Controls.TextBox) (target));
00434             return;
00435         case 14:
00436             this.XRegister = ((System.Windows.Controls.TextBox) (target));
00437             return;
00438         case 15:
00439             this.Accumulator = ((System.Windows.Controls.TextBox) (target));
00440             return;
00441         case 16:
00442             this.StackPointer = ((System.Windows.Controls.TextBox) (target));
00443             return;
00444         case 17:
00445             this.ProgramCounter = ((System.Windows.Controls.TextBox) (target));
00446             return;
00447         case 18:
00448             this.Dissambly = ((System.Windows.Controls.TextBox) (target));
00449             return;
00450         case 19:
00451             this.CycleCount = ((System.Windows.Controls.TextBox) (target));
00452             return;
00453         case 20:
00454             this.XRegisterText = ((System.Windows.Controls.TextBlock) (target));
00455             return;
00456         case 21:
00457             this.YRegisterText = ((System.Windows.Controls.TextBlock) (target));
00458             return;
00459         case 22:
00460             this.StackPointerRegisterText = ((System.Windows.Controls.TextBlock) (target));
00461             return;
00462         case 23:
00463             this.AText = ((System.Windows.Controls.TextBlock) (target));
00464             return;
00465         case 24:
00466             this.CurrentInstructionText = ((System.Windows.Controls.TextBlock) (target));
00467             return;
00468         case 25:
00469             this.ProgramCounterText = ((System.Windows.Controls.TextBlock) (target));
00470             return;
00471         case 26:
00472             this.CycleCountText = ((System.Windows.Controls.TextBlock) (target));
00473             return;
00474         case 27:
00475             this.CarryFlag = ((System.Windows.Controls.CheckBox) (target));
00476             return;
00477         case 28:
00478             this.CarryFlagText = ((System.Windows.Controls.TextBlock) (target));
00479             return;
00480         case 29:
00481             this.ZeroFlag = ((System.Windows.Controls.CheckBox) (target));
00482             return;
00483         case 30:
00484             this.ZeroFlagText = ((System.Windows.Controls.TextBlock) (target));
00485             return;
00486         case 31:
00487             this.InterruptFlag = ((System.Windows.Controls.CheckBox) (target));
00488             return;
00489         case 32:
00490             this.InterruptFlagText = ((System.Windows.Controls.TextBlock) (target));
00491             return;
00492         case 33:
00493             this.BcdFlag = ((System.Windows.Controls.CheckBox) (target));
00494             return;
00495         case 34:
00496             this.BcdFlagText = ((System.Windows.Controls.TextBlock) (target));
00497             return;
00498         case 35:
00499             this.BreakFlag = ((System.Windows.Controls.CheckBox) (target));
00500             return;
```

```

00501         case 36:
00502             this.BreakFlagText = ((System.Windows.Controls.TextBlock)(target));
00503             return;
00504         case 37:
00505             this.OverflowFlag = ((System.Windows.Controls.CheckBox)(target));
00506             return;
00507         case 38:
00508             this.OverflowFlagText = ((System.Windows.Controls.TextBlock)(target));
00509             return;
00510         case 39:
00511             this.NegativeFlag = ((System.Windows.Controls.CheckBox)(target));
00512             return;
00513         case 40:
00514             this.NegativeFlagText = ((System.Windows.Controls.TextBlock)(target));
00515             return;
00516         case 41:
00517             this.CpuSpeed = ((System.Windows.Controls.Slider)(target));
00518             return;
00519         case 42:
00520             this.SpeedText = ((System.Windows.Controls.TextBlock)(target));
00521             return;
00522         }
00523         this._contentLoaded = true;
00524     }
00525 }
00526 }
00527

```

7.97 Emulator/obj/x86/Debug/MemoryVisual.g.cs File Reference

Classes

- class [Emulator.MemoryVisual](#)
Interaction logic for Window1.xaml

Namespaces

- namespace [Emulator](#)

7.98 MemoryVisual.g.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\MemoryVisual.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00002 "7B2B3A560666411FAE151D1ACBF76FF18A3D6578E63FF227C3D020B54E19F5CB"
00003 //-----
00004 // <auto-generated>
00005 //     This code was generated by a tool.
00006 //     Runtime Version:4.0.30319.42000
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;

```

```

00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037 /// <summary>
00038 /// MemoryVisual
00039 /// </summary>
00040 public partial class MemoryVisual : System.Windows.Window,
    System.Windows.Markup.IComponentConnector {
00041
00042
00043 #line 46 "..\..\..\MemoryVisual.xaml"
00044 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
    "CA1823:AvoidUnusedPrivateFields")]
00045     internal System.Windows.Controls.DataGrid MemoryMap;
00046
00047 #line default
00048 #line hidden
00049
00050
00051 #line 69 "..\..\..\MemoryVisual.xaml"
00052 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
    "CA1823:AvoidUnusedPrivateFields")]
00053     internal System.Windows.Controls.TextBox CurrentPage;
00054
00055 #line default
00056 #line hidden
00057
00058
00059 #line 74 "..\..\..\MemoryVisual.xaml"
00060 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
    "CA1823:AvoidUnusedPrivateFields")]
00061     internal System.Windows.Controls.TextBlock CurrentPageText;
00062
00063 #line default
00064 #line hidden
00065
00066     private bool _contentLoaded;
00067
00068 /// <summary>
00069 /// InitializeComponent
00070 /// </summary>
00071 [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00072 [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00073     public void InitializeComponent() {
00074         if (_contentLoaded) {
00075             return;
00076         }
00077         _contentLoaded = true;
00078         System.Uri resourceLocater = new System.Uri("/Emulator;component/memoryvisual.xaml",
    System.UriKind.Relative);
00079
00080 #line 1 "..\..\..\MemoryVisual.xaml"
00081         System.Windows.Application.LoadComponent(this, resourceLocater);
00082
00083 #line default
00084 #line hidden
00085     }
00086
00087 [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00088 [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00089
    [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00090 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
    "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00091 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
    "CA1502:AvoidExcessiveComplexity")]
00092 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
    "CA1800:DoNotCastUnnecessarily")]
00093     void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00094         switch (connectionId)
00095         {
00096             case 1:
00097                 this.MemoryMap = ((System.Windows.Controls.DataGrid) (target));
00098                 return;
00099             case 2:
00100                 this.CurrentPage = ((System.Windows.Controls.TextBox) (target));
00101                 return;
00102             case 3:
00103                 this.CurrentPageText = ((System.Windows.Controls.TextBlock) (target));
00104                 return;
00105         }
00106         this._contentLoaded = true;
00107     }

```

```

00108     }
00109 }
00110

```

7.99 Emulator/obj/x86/Release/MemoryVisual.g.cs File Reference

Classes

- class [Emulator.MemoryVisual](#)
Interaction logic for Window1.xaml

Namespaces

- namespace [Emulator](#)

7.100 MemoryVisual.g.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\MemoryVisual.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00002     "7B2B3A560666411FAE151D1ACBF76FF18A3D6578E63FF227C3D020B54E19F5CB"
00003 //-----
00004 // <auto-generated>
00005 //     This code was generated by a tool.
00006 //     Runtime Version:4.0.30319.42000
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037     /// <summary>
00038     ///     MemoryVisual
00039     /// </summary>
00040     public partial class MemoryVisual : System.Windows.Window,
00041         System.Windows.Markup.IComponentConnector {
00042
00043         #line 46 "..\..\..\MemoryVisual.xaml"
00044         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00045             "CA1823:AvoidUnusedPrivateFields")]
00046         internal System.Windows.Controls.DataGrid MemoryMap;
00047
00048         #line default
00049         #line hidden
00050
00051         #line 69 "..\..\..\MemoryVisual.xaml"

```

```

00052         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00053         internal System.Windows.Controls.TextBox currentPage;
00054
00055 #line default
00056 #line hidden
00057
00058
00059 #line 74 "..\..\..\MemoryVisual.xaml"
00060         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00061         internal System.Windows.Controls.TextBlock currentPageText;
00062
00063 #line default
00064 #line hidden
00065
00066         private bool _contentLoaded;
00067
00068 /// <summary>
00069 /// InitializeComponent
00070 /// </summary>
00071         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00072         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00073         public void InitializeComponent() {
00074             if (_contentLoaded) {
00075                 return;
00076             }
00077             _contentLoaded = true;
00078             System.Uri resourceLocator = new System.Uri("/Emulator;component/memoryvisual.xaml",
System.UriKind.Relative);
00079
00080 #line 1 "..\..\..\MemoryVisual.xaml"
00081             System.Windows.Application.LoadComponent(this, resourceLocator);
00082
00083 #line default
00084 #line hidden
00085         }
00086
00087         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00088         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00089
00090         [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00091         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
"CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00092         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
"CA1502:AvoidExcessiveComplexity")]
00093         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1800:DoNotCastUnnecessarily")]
00094         void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00095             switch (connectionId)
00096             {
00097                 case 1:
00098                     this.MemoryMap = ((System.Windows.Controls.DataGrid) (target));
00099                     return;
00100                 case 2:
00101                     this.CurrentPage = ((System.Windows.Controls.TextBox) (target));
00102                     return;
00103                 case 3:
00104                     this.CurrentPageText = ((System.Windows.Controls.TextBlock) (target));
00105                     return;
00106             }
00107             this._contentLoaded = true;
00108         }
00109     }
00110

```

7.101 Emulator/obj/x86/Debug/MemoryVisual.g.i.cs File Reference

Classes

- class [Emulator.MemoryVisual](#)
Interaction logic for Window1.xaml

Namespaces

- namespace [Emulator](#)

7.102 MemoryVisual.g.i.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\MemoryVisual.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00002 "7B2B3A560666411FAE151D1ACBF76FF18A3D6578E63FF227C3D020B54E19F5CB"
00003 //-----
00004 // <auto-generated>
00005 //     This code was generated by a tool.
00006 //     Runtime Version:4.0.30319.42000
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037     /// <summary>
00038     /// MemoryVisual
00039     /// </summary>
00040     public partial class MemoryVisual : System.Windows.Window,
00041         System.Windows.Markup.IComponentConnector {
00042
00043         #line 46 "..\..\..\MemoryVisual.xaml"
00044         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00045             "CA1823:AvoidUnusedPrivateFields")]
00046         internal System.Windows.Controls.DataGrid MemoryMap;
00047
00048         #line default
00049         #line hidden
00050
00051         #line 69 "..\..\..\MemoryVisual.xaml"
00052         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00053             "CA1823:AvoidUnusedPrivateFields")]
00054         internal System.Windows.Controls.TextBox CurrentPage;
00055
00056         #line default
00057         #line hidden
00058
00059         #line 74 "..\..\..\MemoryVisual.xaml"
00060         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00061             "CA1823:AvoidUnusedPrivateFields")]
00062         internal System.Windows.Controls.TextBlock CurrentPageText;
00063
00064         #line default
00065         #line hidden
00066         private bool _contentLoaded;
00067
00068         /// <summary>
00069         /// InitializeComponent
00070         /// </summary>
00071         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00072         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00073         public void InitializeComponent() {
00074             if (_contentLoaded) {
00075                 return;
00076             }
00077             _contentLoaded = true;

```

```

00078         System.Uri resourceLocator = new System.Uri("/Emulator;component/memoryvisual.xaml",
00079             System.UriKind.Relative);
00080 #line 1 "..\\..\\..\\MemoryVisual.xaml"
00081         System.Windows.Application.LoadComponent(this, resourceLocator);
00082
00083 #line default
00084 #line hidden
00085     }
00086
00087     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00088     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00089
00090     [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00091     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
00092         "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00093     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
00094         "CA1502:AvoidExcessiveComplexity")]
00095     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00096         "CA1800:DoNotCastUnnecessarily")]
00097     void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00098         switch (connectionId)
00099         {
00100             case 1:
00101                 this.MemoryMap = ((System.Windows.Controls.DataGrid) (target));
00102                 return;
00103             case 2:
00104                 this.CurrentPage = ((System.Windows.Controls.TextBox) (target));
00105                 return;
00106             case 3:
00107                 this.CurrentPageText = ((System.Windows.Controls.TextBlock) (target));
00108                 return;
00109             }
00110         this._contentLoaded = true;
00111     }
00112 }
00113
00114 }
00115
00116 }
00117
00118 }
00119
00120 }
00121
00122 }
00123
00124 }
00125
00126 }
00127
00128 }
00129
00130 }
00131
00132 }
00133
00134 }
00135
00136 }
00137
00138 }
00139
00140 }
00141
00142 }
00143
00144 }
00145
00146 }
00147
00148 }
00149
00150 }
00151
00152 }
00153
00154 }
00155
00156 }
00157
00158 }
00159
00160 }
00161
00162 }
00163
00164 }
00165
00166 }
00167
00168 }
00169
00170 }
00171
00172 }
00173
00174 }
00175
00176 }
00177
00178 }
00179
00180 }
00181
00182 }
00183
00184 }
00185
00186 }
00187
00188 }
00189
00190 }
00191
00192 }
00193
00194 }
00195
00196 }
00197
00198 }
00199
00200 }
00201
00202 }
00203
00204 }
00205
00206 }
00207
00208 }
00209
00210 }
00211
00212 }
00213
00214 }
00215
00216 }
00217
00218 }
00219
00220 }
00221
00222 }
00223
00224 }
00225
00226 }
00227
00228 }
00229
00230 }
00231
00232 }
00233
00234 }
00235
00236 }
00237
00238 }
00239
00240 }
00241
00242 }
00243
00244 }
00245
00246 }
00247
00248 }
00249
00250 }
00251
00252 }
00253
00254 }
00255
00256 }
00257
00258 }
00259
00260 }
00261
00262 }
00263
00264 }
00265
00266 }
00267
00268 }
00269
00270 }
00271
00272 }
00273
00274 }
00275
00276 }
00277
00278 }
00279
00280 }
00281
00282 }
00283
00284 }
00285
00286 }
00287
00288 }
00289
00290 }
00291
00292 }
00293
00294 }
00295
00296 }
00297
00298 }
00299
00300 }
00301
00302 }
00303
00304 }
00305
00306 }
00307
00308 }
00309
00310 }
00311
00312 }
00313
00314 }
00315
00316 }
00317
00318 }
00319
00320 }
00321
00322 }
00323
00324 }
00325
00326 }
00327
00328 }
00329
00330 }
00331
00332 }
00333
00334 }
00335
00336 }
00337
00338 }
00339
00340 }
00341
00342 }
00343
00344 }
00345
00346 }
00347
00348 }
00349
00350 }
00351
00352 }
00353
00354 }
00355
00356 }
00357
00358 }
00359
00360 }
00361
00362 }
00363
00364 }
00365
00366 }
00367
00368 }
00369
00370 }
00371
00372 }
00373
00374 }
00375
00376 }
00377
00378 }
00379
00380 }
00381
00382 }
00383
00384 }
00385
00386 }
00387
00388 }
00389
00390 }
00391
00392 }
00393
00394 }
00395
00396 }
00397
00398 }
00399
00400 }
00401
00402 }
00403
00404 }
00405
00406 }
00407
00408 }
00409
00410 }
00411
00412 }
00413
00414 }
00415
00416 }
00417
00418 }
00419
00420 }
00421
00422 }
00423
00424 }
00425
00426 }
00427
00428 }
00429
00430 }
00431
00432 }
00433
00434 }
00435
00436 }
00437
00438 }
00439
00440 }
00441
00442 }
00443
00444 }
00445
00446 }
00447
00448 }
00449
00450 }
00451
00452 }
00453
00454 }
00455
00456 }
00457
00458 }
00459
00460 }
00461
00462 }
00463
00464 }
00465
00466 }
00467
00468 }
00469
00470 }
00471
00472 }
00473
00474 }
00475
00476 }
00477
00478 }
00479
00480 }
00481
00482 }
00483
00484 }
00485
00486 }
00487
00488 }
00489
00490 }
00491
00492 }
00493
00494 }
00495
00496 }
00497
00498 }
00499
00500 }
00501
00502 }
00503
00504 }
00505
00506 }
00507
00508 }
00509
00510 }
00511
00512 }
00513
00514 }
00515
00516 }
00517
00518 }
00519
00520 }
00521
00522 }
00523
00524 }
00525
00526 }
00527
00528 }
00529
00530 }
00531
00532 }
00533
00534 }
00535
00536 }
00537
00538 }
00539
00540 }
00541
00542 }
00543
00544 }
00545
00546 }
00547
00548 }
00549
00550 }
00551
00552 }
00553
00554 }
00555
00556 }
00557
00558 }
00559
00560 }
00561
00562 }
00563
00564 }
00565
00566 }
00567
00568 }
00569
00570 }
00571
00572 }
00573
00574 }
00575
00576 }
00577
00578 }
00579
00580 }
00581
00582 }
00583
00584 }
00585
00586 }
00587
00588 }
00589
00590 }
00591
00592 }
00593
00594 }
00595
00596 }
00597
00598 }
00599
00600 }
00601
00602 }
00603
00604 }
00605
00606 }
00607
00608 }
00609
00610 }
00611
00612 }
00613
00614 }
00615
00616 }
00617
00618 }
00619
00620 }
00621
00622 }
00623
00624 }
00625
00626 }
00627
00628 }
00629
00630 }
00631
00632 }
00633
00634 }
00635
00636 }
00637
00638 }
00639
00640 }
00641
00642 }
00643
00644 }
00645
00646 }
00647
00648 }
00649
00650 }
00651
00652 }
00653
00654 }
00655
00656 }
00657
00658 }
00659
00660 }
00661
00662 }
00663
00664 }
00665
00666 }
00667
00668 }
00669
00670 }
00671
00672 }
00673
00674 }
00675
00676 }
00677
00678 }
00679
00680 }
00681
00682 }
00683
00684 }
00685
00686 }
00687
00688 }
00689
00690 }
00691
00692 }
00693
00694 }
00695
00696 }
00697
00698 }
00699
00700 }
00701
00702 }
00703
00704 }
00705
00706 }
00707
00708 }
00709
00710 }
00711
00712 }
00713
00714 }
00715
00716 }
00717
00718 }
00719
00720 }
00721
00722 }
00723
00724 }
00725
00726 }
00727
00728 }
00729
00730 }
00731
00732 }
00733
00734 }
00735
00736 }
00737
00738 }
00739
00740 }
00741
00742 }
00743
00744 }
00745
00746 }
00747
00748 }
00749
00750 }
00751
00752 }
00753
00754 }
00755
00756 }
00757
00758 }
00759
00760 }
00761
00762 }
00763
00764 }
00765
00766 }
00767
00768 }
00769
00770 }
00771
00772 }
00773
00774 }
00775
00776 }
00777
00778 }
00779
00780 }
00781
00782 }
00783
00784 }
00785
00786 }
00787
00788 }
00789
00790 }
00791
00792 }
00793
00794 }
00795
00796 }
00797
00798 }
00799
00800 }
00801
00802 }
00803
00804 }
00805
00806 }
00807
00808 }
00809
00810 }
00811
00812 }
00813
00814 }
00815
00816 }
00817
00818 }
00819
00820 }
00821
00822 }
00823
00824 }
00825
00826 }
00827
00828 }
00829
00830 }
00831
00832 }
00833
00834 }
00835
00836 }
00837
00838 }
00839
00840 }
00841
00842 }
00843
00844 }
00845
00846 }
00847
00848 }
00849
00850 }
00851
00852 }
00853
00854 }
00855
00856 }
00857
00858 }
00859
00860 }
00861
00862 }
00863
00864 }
00865
00866 }
00867
00868 }
00869
00870 }
00871
00872 }
00873
00874 }
00875
00876 }
00877
00878 }
00879
00880 }
00881
00882 }
00883
00884 }
00885
00886 }
00887
00888 }
00889
00890 }
00891
00892 }
00893
00894 }
00895
00896 }
00897
00898 }
00899
00900 }
00901
00902 }
00903
00904 }
00905
00906 }
00907
00908 }
00909
00910 }
00911
00912 }
00913
00914 }
00915
00916 }
00917
00918 }
00919
00920 }
00921
00922 }
00923
00924 }
00925
00926 }
00927
00928 }
00929
00930 }
00931
00932 }
00933
00934 }
00935
00936 }
00937
00938 }
00939
00940 }
00941
00942 }
00943
00944 }
00945
00946 }
00947
00948 }
00949
00950 }
00951
00952 }
00953
00954 }
00955
00956 }
00957
00958 }
00959
00960 }
00961
00962 }
00963
00964 }
00965
00966 }
00967
00968 }
00969
00970 }
00971
00972 }
00973
00974 }
00975
00976 }
00977
00978 }
00979
00980 }
00981
00982 }
00983
00984 }
00985
00986 }
00987
00988 }
00989
00990 }
00991
00992 }
00993
00994 }
00995
00996 }
00997
00998 }
00999
01000 }
01001
01002 }
01003
01004 }
01005
01006 }
01007
01008 }
01009
01010 }
01011
01012 }
01013
01014 }
01015
01016 }
01017
01018 }
01019
01020 }
01021
01022 }
01023
01024 }
01025
01026 }
01027
01028 }
01029
01030 }
01031
01032 }
01033
01034 }
01035
01036 }
01037
01038 }
01039
01040 }
01041
01042 }
01043
01044 }
01045
01046 }
01047
01048 }
01049
01050 }
01051
01052 }
01053
01054 }
01055
01056 }
01057
01058 }
01059
01060 }
01061
01062 }
01063
01064 }
01065
01066 }
01067
01068 }
01069
01070 }
01071
01072 }
01073
01074 }
01075
01076 }
01077
01078 }
01079
01080 }
01081
01082 }
01083
01084 }
01085
01086 }
01087
01088 }
01089
01090 }
01091
01092 }
01093
01094 }
01095
01096 }
01097
01098 }
01099
01100 }
01101
01102 }
01103
01104 }
01105
01106 }
01107
01108 }
01109
01110 }
01111
01112 }
01113
01114 }
01115
01116 }
01117
01118 }
01119
01120 }
01121
01122 }
01123
01124 }
01125
01126 }
01127
01128 }
01129
01130 }
01131
01132 }
01133
01134 }
01135
01136 }
01137
01138 }
01139
01140 }
01141
01142 }
01143
01144 }
01145
01146 }
01147
01148 }
01149
01150 }
01151
01152 }
01153
01154 }
01155
01156 }
01157
01158 }
01159
01160 }
01161
01162 }
01163
01164 }
01165
01166 }
01167
01168 }
01169
01170 }
01171
01172 }
01173
01174 }
01175
01176 }
01177
01178 }
01179
01180 }
01181
01182 }
01183
01184 }
01185
01186 }
01187
01188 }
01189
01190 }
01191
01192 }
01193
01194 }
01195
01196 }
01197
01198 }
01199
01200 }
01201
01202 }
01203
01204 }
01205
01206 }
01207
01208 }
01209
01210 }
01211
01212 }
01213
01214 }
01215
01216 }
01217
01218 }
01219
01220 }
01221
01222 }
01223
01224 }
01225
01226 }
01227
01228 }
01229
01230 }
01231
01232 }
01233
01234 }
01235
01236 }
01237
01238 }
01239
01240 }
01241
01242 }
01243
01244 }
01245
01246 }
01247
01248 }
01249
01250 }
01251
01252 }
01253
01254 }
01255
01256 }
01257
01258 }
01259
01260 }
01261
01262 }
01263
01264 }
01265
01266 }
01267
01268 }
01269
01270 }
01271
01272 }
01273
01274 }
01275
01276 }
01277
01278 }
01279
01280 }
01281
01282 }
01283
01284 }
01285
01286 }
01287
01288 }
01289
01290 }
01291
01292 }
01293
01294 }
01295
01296 }
01297
01298 }
01299
01300 }
01301
01302 }
01303
01304 }
01305
01306 }
01307
01308 }
01309
01310 }
01311
01312 }
01313
01314 }
01315
01316 }
01317
01318 }
01319
01320 }
01321
01322 }
01323
01324 }
01325
01326 }
01327
01328 }
01329
01330 }
01331
01332 }
01333
01334 }
01335
01336 }
01337
01338 }
01339
01340 }
01341
01342 }
01343
01344 }
01345
01346 }
01347
01348 }
01349
01350 }
01351
01352 }
01353
01354 }
01355
01356 }
01357
01358 }
01359
01360 }
01361
01362 }
01363
01364 }
01365
01366 }
01367
01368 }
01369
01370 }
01371
01372 }
01373
01374 }
01375
01376 }
01377
01378 }
01379
01380 }
01381
01382 }
01383
01384 }
01385
01386 }
01387
01388 }
01389
01390 }
01391
01392 }
01393
01394 }
01395
01396 }
01397
01398 }
01399
01400 }
01401
01402 }
01403
01404 }
01405
01406 }
01407
01408 }
01409
01410 }
01411
01412 }
01413
01414 }
01415
01416 }
01417
01418 }
01419
01420 }
01421
01422 }
01423
01424 }
01425
01426 }
01427
01428 }
01429
01430 }
01431
01432 }
01433
01434 }
01435
01436 }
01437
01438 }
01439
01440 }
01441
01442 }
01443
01444 }
01445
01446 }
01447
01448 }
01449
01450 }
01451
01452 }
01453
01454 }
01455
01456 }
01457
01458 }
01459
01460 }
01461
01462 }
01463
01464 }
01465
01466 }
01467
01468 }
01469
01470 }
01471
01472 }
01473
01474 }
01475
01476 }
01477
01478 }
01479
01480 }
01481
01482 }
01483
01484 }
01485
01486 }
01487
01488 }
01489
01490 }
01491
01492 }
01493
01494 }
01495
01496 }
01497
01498 }
01499
01500 }
01501
01502 }
01503
01504 }
01505
01506 }
01507
01508 }
01509
01510 }
01511
01512 }
01513
01514 }
01515
01516 }
01517
01518 }
01519
01520 }
01521
01522 }
01523
01524 }
01525
01526 }
01527
01528 }
01529
01530 }
01531
01532 }
01533
01534 }
01535
01536 }
01537
01538 }
01539
01540 }
01541
01542 }
01543
01544 }
01545
01546 }
01547
01548 }
01549
01550 }
01551
01552 }
01553
01554 }
01555
01556 }
01557
01558 }
01559
01560 }
01561
01562 }
01563
01564 }
01565
01566 }
01567
01568 }
01569
01570 }
01571
01572 }
01573
01574 }
01575
01576 }
01577
01578 }
01579
01580 }
01581
01582 }
01583
01584 }
01585
01586 }
01587
01588 }
01589
01590 }
01591
01592 }
01593
01594 }
01595
01596 }
01597
01598 }
01599
01600 }
01601
01602 }
01603
01604 }
01605
01606 }
01607
01608 }
01609
01610 }
01611
01612 }
01613
01614 }
01615
01616 }
01617
01618 }
01619
01620 }
01621
01622 }
01623
01624 }
01625
01626 }
01627
01628 }
01629
01630 }
01631
01632 }
01633
01634 }
01635
01636 }
01637
01638 }
01639
01640 }
01641
01642 }
01643
01644 }
01645
01646 }
01647
01648 }
01649
01650 }
01651
01652 }
01653
01654 }
01655
01656 }
01657
01658 }
01659
01660 }
01661
01662 }
01663
01664 }
01665
01666 }
01667
01668 }
01669
01670 }
01671
01672 }
01673
01674 }
01675
01676 }
01677
01678 }
01679
01680 }
01681
01682 }
01683
01684 }
01685
01686 }
01687
01688 }
01689
01690 }
01691
01692 }
01693
01694 }
01695
01696 }
01697
01698 }
01699
01700 }
01701
01702 }
01703
01704 }
01705
01706 }
01707
01708 }
01709
01710 }
01711
01712 }
01713
01714 }
01715
01716 }
01717
01718 }
01719
01720 }
01721
01722 }
01723
01724 }
01725
01726 }
01727
01728 }
01729
01730 }
01731
01732 }
01733
01734 }
01735
01736 }
01737
01738 }
01739
01740 }
01741
01742 }
01743
01744 }
01745
01746 }
01747
01748 }
01749
01750 }
01751
01752 }
01753
01754 }
01755
01756 }
01757
01758 }
01759
01760 }
01761
01762 }
01763
01764 }
01765
01766 }
01767
01768 }
01769
01770 }
01771
01772 }
01773
01774 }
01775
01776 }
01777
01778 }
01779
01780 }
01781
01782 }
01783
01784 }
01785
01786 }
01787
01788 }
01789
01790 }
01791
01792 }
01793
01794 }
01795
01796 }
01797
01798 }
01799
01800 }
01801
01802 }
01803
01804 }
01805
01806 }
01807
01808 }
01809
01810 }
01811
01812 }
01813
01814 }
01815
01816 }
01817
01818 }
01819
01820 }
01821
01822 }
01823
01824 }
01825
01826 }
01827
01828 }
01829
01830 }
01831
01832 }
01833
01834 }
01835
01836 }
01837
01838 }
01839
01840 }
01841
01842 }
01843
01844 }
01845
01846 }
01847
01848 }
01849
01850 }
01851
01852 }
01853
01854 }
01855
01856 }
01857
01858 }
01859
01860 }
01861
01862 }
01863
01864 }
01865
01866 }
01867
01868 }
01869
01870 }
01871
01872 }
01873
01874 }
01875
01876 }
01877
01878 }
01879
01880 }
01881
01882 }
01883
01884 }
01885
01886 }
01887
01888 }
01889
01890 }
01891
01892 }
01893
01894 }
01895
01896 }
01897
01898 }
01899
01900 }
01901
01902 }
01903
01904 }
01905
01906 }
01907
01908 }
01909
01910 }
01911
01912 }
01913
01914 }
01915
01916 }
01917
01918 }
01919
01920 }
01921
01922
```



```

00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037 /// <summary>
00038 /// MemoryVisual
00039 /// </summary>
00040 public partial class MemoryVisual : System.Windows.Window,
    System.Windows.Markup.IComponentConnector {
00041
00042
00043 #line 46 "..\..\..\MemoryVisual.xaml"
00044     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00045         "CA1823:AvoidUnusedPrivateFields")]
00046     internal System.Windows.Controls.DataGrid MemoryMap;
00047
00048 #line default
00049 #line hidden
00050
00051 #line 69 "..\..\..\MemoryVisual.xaml"
00052     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00053         "CA1823:AvoidUnusedPrivateFields")]
00054     internal System.Windows.Controls.TextBox CurrentPage;
00055
00056 #line default
00057 #line hidden
00058
00059 #line 74 "..\..\..\MemoryVisual.xaml"
00060     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00061         "CA1823:AvoidUnusedPrivateFields")]
00062     internal System.Windows.Controls.TextBlock CurrentPageText;
00063
00064 #line default
00065 #line hidden
00066     private bool _contentLoaded;
00067
00068 /// <summary>
00069 /// InitializeComponent
00070 /// </summary>
00071     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00072     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00073     public void InitializeComponent() {
00074         if (_contentLoaded) {
00075             return;
00076         }
00077         _contentLoaded = true;
00078         System.Uri resourceLocator = new System.Uri("/Emulator;component/memoryvisual.xaml",
00079             System.UriKind.Relative);
00080
00081 #line 1 "..\..\..\MemoryVisual.xaml"
00082         System.Windows.Application.LoadComponent(this, resourceLocator);
00083
00084 #line default
00085 #line hidden
00086     }
00087
00088     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00089     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00090     [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00091     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
00092         "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00093     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
00094         "CA1502:AvoidExcessiveComplexity")]
00095     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00096         "CA1800:DoNotCastUnnecessarily")]
00097     void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00098         switch (connectionId)
00099         {
00100             case 1:

```

```

00097         this.MemoryMap = ((System.Windows.Controls.DataGrid) (target));
00098         return;
00099         case 2:
00100             this.CurrentPage = ((System.Windows.Controls.TextBox) (target));
00101             return;
00102             case 3:
00103                 this.CurrentPageText = ((System.Windows.Controls.TextBlock) (target));
00104                 return;
00105             }
00106             this._contentLoaded = true;
00107         }
00108     }
00109 }
00110

```

7.105 Emulator/obj/x86/Debug/SaveFile.g.cs File Reference

Classes

- class [Emulator.SaveFile](#)
SaveFile

Namespaces

- namespace [Emulator](#)

7.106 SaveFile.g.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\SaveFile.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00002 "34689CE75633CB3BE5E4FDF3C6E7ECDD6274F88E3F05662C41A2D31C677175A9"
00002 //-----
00003 // <auto-generated>
00004 //      This code was generated by a tool.
00005 //      Runtime Version:4.0.30319.42000
00006 //
00007 //      Changes to this file may cause incorrect behavior and will be lost if
00008 //      the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037     /// <summary>
00038     /// SaveFile
00039     /// </summary>
00040     public partial class SaveFile : System.Windows.Window, System.Windows.Markup.IComponentConnector
00041     {
00042

```

```

00042
00043 #line 7 "..\..\..\SaveFile.xaml"
00044     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00045     internal System.Windows.Controls.Button SelectFile;
00046
00047 #line default
00048 #line hidden
00049
00050
00051 #line 8 "..\..\..\SaveFile.xaml"
00052     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00053     internal System.Windows.Controls.TextBox FilePath;
00054
00055 #line default
00056 #line hidden
00057
00058
00059 #line 9 "..\..\..\SaveFile.xaml"
00060     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00061     internal System.Windows.Controls.TextBlock PathText;
00062
00063 #line default
00064 #line hidden
00065
00066
00067 #line 10 "..\..\..\SaveFile.xaml"
00068     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00069     internal System.Windows.Controls.Button CancelButton;
00070
00071 #line default
00072 #line hidden
00073
00074
00075 #line 11 "..\..\..\SaveFile.xaml"
00076     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00077     internal System.Windows.Controls.Button LoadButton;
00078
00079 #line default
00080 #line hidden
00081
00082     private bool _contentLoaded;
00083
00084     /// <summary>
00085     /// InitializeComponent
00086     /// </summary>
00087     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00088     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00089     public void InitializeComponent() {
00090         if (_contentLoaded) {
00091             return;
00092         }
00093         _contentLoaded = true;
00094         System.Uri resourceLocator = new System.Uri("/Emulator;component/savefile.xaml",
System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097         System.Windows.Application.LoadComponent(this, resourceLocator);
00098
00099 #line default
00100 #line hidden
00101     }
00102
00103     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00104     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00105
00106     [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00107     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
"CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00108     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
"CA1502:AvoidExcessiveComplexity")]
00109     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1800:DoNotCastUnnecessarily")]
00110     void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00111         switch (connectionId)
00112         {
00113             case 1:
00114                 this.SelectFile = ((System.Windows.Controls.Button) (target));
00115                 return;
00116             case 2:
00117                 this.FilePath = ((System.Windows.Controls.TextBox) (target));
00118                 return;
00119             case 3:

```

```

00119         this.PathText = ((System.Windows.Controls.TextBlock) (target));
00120         return;
00121     case 4:
00122         this.CancelButton = ((System.Windows.Controls.Button) (target));
00123         return;
00124     case 5:
00125         this.LoadButton = ((System.Windows.Controls.Button) (target));
00126         return;
00127     }
00128     this._contentLoaded = true;
00129 }
00130 }
00131 }
00132

```

7.107 Emulator/obj/x86/Publish/SaveFile.g.cs File Reference

Classes

- class [Emulator.SaveFile](#)
SaveFile

Namespaces

- namespace [Emulator](#)

7.108 SaveFile.g.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\SaveFile.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00002 "34689CE75633CB3BE5E4FDF3C6E7ECDD6274F88E3F05662C41A2D31C677175A9"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037     /// <summary>
00038     /// SaveFile
00039     /// </summary>
00040     public partial class SaveFile : System.Windows.Window, System.Windows.Markup.IComponentConnector
00041     {

```

```

00042
00043 #line 7 "..\..\..\SaveFile.xaml"
00044     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00045         "CA1823:AvoidUnusedPrivateFields")]
00046     internal System.Windows.Controls.Button SelectFile;
00047 #line default
00048 #line hidden
00049
00050
00051 #line 8 "..\..\..\SaveFile.xaml"
00052     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00053         "CA1823:AvoidUnusedPrivateFields")]
00054     internal System.Windows.Controls.TextBox FilePath;
00055 #line default
00056 #line hidden
00057
00058
00059 #line 9 "..\..\..\SaveFile.xaml"
00060     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00061         "CA1823:AvoidUnusedPrivateFields")]
00062     internal System.Windows.Controls.TextBlock PathText;
00063 #line default
00064 #line hidden
00065
00066
00067 #line 10 "..\..\..\SaveFile.xaml"
00068     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00069         "CA1823:AvoidUnusedPrivateFields")]
00070     internal System.Windows.Controls.Button CancelButton;
00071 #line default
00072 #line hidden
00073
00074
00075 #line 11 "..\..\..\SaveFile.xaml"
00076     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00077         "CA1823:AvoidUnusedPrivateFields")]
00078     internal System.Windows.Controls.Button LoadButton;
00079 #line default
00080 #line hidden
00081
00082     private bool _contentLoaded;
00083
00084     /// <summary>
00085     /// InitializeComponent
00086     /// </summary>
00087     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00088     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00089     public void InitializeComponent() {
00090         if (_contentLoaded) {
00091             return;
00092         }
00093         _contentLoaded = true;
00094         System.Uri resourceLocator = new System.Uri("/Emulator;component/savefile.xaml",
00095             System.UriKind.Relative);
00096 #line 1 "..\..\..\SaveFile.xaml"
00097         System.Windows.Application.LoadComponent(this, resourceLocator);
00098 #line default
00099 #line hidden
00100     }
00101
00102
00103     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00104     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00105
00106     [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00107     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
00108         "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00109     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
00110         "CA1502:AvoidExcessiveComplexity")]
00111     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00112         "CA1800:DoNotCastUnnecessarily")]
00113     void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00114         switch (connectionId)
00115         {
00116             case 1:
00117                 this.SelectFile = ((System.Windows.Controls.Button) (target));
00118                 return;
00119             case 2:
00120                 this.FilePath = ((System.Windows.Controls.TextBox) (target));
00121                 return;
00122             case 3:

```

```

00119         this.PathText = ((System.Windows.Controls.TextBlock) (target));
00120         return;
00121     case 4:
00122         this.CancelButton = ((System.Windows.Controls.Button) (target));
00123         return;
00124     case 5:
00125         this.LoadButton = ((System.Windows.Controls.Button) (target));
00126         return;
00127     }
00128     this._contentLoaded = true;
00129 }
00130 }
00131 }
00132

```

7.109 Emulator/obj/x86/Release/SaveFile.g.cs File Reference

Classes

- class [Emulator.SaveFile](#)
SaveFile

Namespaces

- namespace [Emulator](#)

7.110 SaveFile.g.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\SaveFile.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00002 "34689CE75633CB3BE5E4FDF3C6E7ECDD6274F88E3F05662C41A2D31C677175A9"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037     /// <summary>
00038     /// SaveFile
00039     /// </summary>
00040     public partial class SaveFile : System.Windows.Window, System.Windows.Markup.IComponentConnector
00041     {

```

```

00042
00043 #line 7 "..\..\..\SaveFile.xaml"
00044     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00045         "CA1823:AvoidUnusedPrivateFields")]
00046     internal System.Windows.Controls.Button SelectFile;
00047 #line default
00048 #line hidden
00049
00050
00051 #line 8 "..\..\..\SaveFile.xaml"
00052     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00053         "CA1823:AvoidUnusedPrivateFields")]
00054     internal System.Windows.Controls.TextBox FilePath;
00055 #line default
00056 #line hidden
00057
00058
00059 #line 9 "..\..\..\SaveFile.xaml"
00060     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00061         "CA1823:AvoidUnusedPrivateFields")]
00062     internal System.Windows.Controls.TextBlock PathText;
00063 #line default
00064 #line hidden
00065
00066
00067 #line 10 "..\..\..\SaveFile.xaml"
00068     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00069         "CA1823:AvoidUnusedPrivateFields")]
00070     internal System.Windows.Controls.Button CancelButton;
00071 #line default
00072 #line hidden
00073
00074
00075 #line 11 "..\..\..\SaveFile.xaml"
00076     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00077         "CA1823:AvoidUnusedPrivateFields")]
00078     internal System.Windows.Controls.Button LoadButton;
00079 #line default
00080 #line hidden
00081
00082     private bool _contentLoaded;
00083
00084     /// <summary>
00085     /// InitializeComponent
00086     /// </summary>
00087     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00088     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00089     public void InitializeComponent() {
00090         if (_contentLoaded) {
00091             return;
00092         }
00093         _contentLoaded = true;
00094         System.Uri resourceLocator = new System.Uri("/Emulator;component/savefile.xaml",
00095             System.UriKind.Relative);
00096 #line 1 "..\..\..\SaveFile.xaml"
00097         System.Windows.Application.LoadComponent(this, resourceLocator);
00098 #line default
00099 #line hidden
00100     }
00101
00102
00103     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00104     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00105
00106     [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00107     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
00108         "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00109     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
00110         "CA1502:AvoidExcessiveComplexity")]
00111     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00112         "CA1800:DoNotCastUnnecessarily")]
00113     void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00114         switch (connectionId)
00115         {
00116             case 1:
00117                 this.SelectFile = ((System.Windows.Controls.Button) (target));
00118                 return;
00119             case 2:
00120                 this.FilePath = ((System.Windows.Controls.TextBox) (target));
00121                 return;
00122             case 3:

```

```

00119         this.PathText = ((System.Windows.Controls.TextBlock) (target));
00120         return;
00121     case 4:
00122         this.CancelButton = ((System.Windows.Controls.Button) (target));
00123         return;
00124     case 5:
00125         this.LoadButton = ((System.Windows.Controls.Button) (target));
00126         return;
00127     }
00128     this._contentLoaded = true;
00129 }
00130 }
00131 }
00132

```

7.111 Emulator/obj/x86/Debug/SaveFile.g.i.cs File Reference

Classes

- class [Emulator.SaveFile](#)
SaveFile

Namespaces

- namespace [Emulator](#)

7.112 SaveFile.g.i.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\SaveFile.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00002 "34689CE75633CB3BE5E4FDF3C6E7ECDD6274F88E3F05662C41A2D31C677175A9"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037     /// <summary>
00038     /// SaveFile
00039     /// </summary>
00040     public partial class SaveFile : System.Windows.Window, System.Windows.Markup.IComponentConnector
00041     {

```



```

00042
00043 #line 7 "..\..\..\SaveFile.xaml"
00044     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00045         "CA1823:AvoidUnusedPrivateFields")]
00046     internal System.Windows.Controls.Button SelectFile;
00047 #line default
00048 #line hidden
00049
00050
00051 #line 8 "..\..\..\SaveFile.xaml"
00052     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00053         "CA1823:AvoidUnusedPrivateFields")]
00054     internal System.Windows.Controls.TextBox FilePath;
00055 #line default
00056 #line hidden
00057
00058
00059 #line 9 "..\..\..\SaveFile.xaml"
00060     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00061         "CA1823:AvoidUnusedPrivateFields")]
00062     internal System.Windows.Controls.TextBlock PathText;
00063 #line default
00064 #line hidden
00065
00066
00067 #line 10 "..\..\..\SaveFile.xaml"
00068     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00069         "CA1823:AvoidUnusedPrivateFields")]
00070     internal System.Windows.Controls.Button CancelButton;
00071 #line default
00072 #line hidden
00073
00074
00075 #line 11 "..\..\..\SaveFile.xaml"
00076     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00077         "CA1823:AvoidUnusedPrivateFields")]
00078     internal System.Windows.Controls.Button LoadButton;
00079 #line default
00080 #line hidden
00081
00082     private bool _contentLoaded;
00083
00084     /// <summary>
00085     /// InitializeComponent
00086     /// </summary>
00087     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00088     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00089     public void InitializeComponent() {
00090         if (_contentLoaded) {
00091             return;
00092         }
00093         _contentLoaded = true;
00094         System.Uri resourceLocator = new System.Uri("/Emulator;component/savefile.xaml",
00095             System.UriKind.Relative);
00096 #line 1 "..\..\..\SaveFile.xaml"
00097         System.Windows.Application.LoadComponent(this, resourceLocator);
00098 #line default
00099 #line hidden
00100     }
00101
00102
00103     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00104     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00105
00106     [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00107     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
00108         "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00109     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
00110         "CA1502:AvoidExcessiveComplexity")]
00111     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00112         "CA1800:DoNotCastUnnecessarily")]
00113     void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00114         switch (connectionId)
00115         {
00116             case 1:
00117                 this.SelectFile = ((System.Windows.Controls.Button) (target));
00118                 return;
00119             case 2:
00120                 this.FilePath = ((System.Windows.Controls.TextBox) (target));
00121                 return;
00122             case 3:

```

```

00119         this.PathText = ((System.Windows.Controls.TextBlock) (target));
00120         return;
00121     case 4:
00122         this.CancelButton = ((System.Windows.Controls.Button) (target));
00123         return;
00124     case 5:
00125         this.LoadButton = ((System.Windows.Controls.Button) (target));
00126         return;
00127     }
00128     this._contentLoaded = true;
00129 }
00130 }
00131 }
00132

```

7.113 Emulator/obj/x86/Publish/SaveFile.g.i.cs File Reference

Classes

- class [Emulator.SaveFile](#)
SaveFile

Namespaces

- namespace [Emulator](#)

7.114 SaveFile.g.i.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\SaveFile.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00002 "34689CE75633CB3BE5E4FDF3C6E7ECDD6274F88E3F05662C41A2D31C677175A9"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037     /// <summary>
00038     /// SaveFile
00039     /// </summary>
00040     public partial class SaveFile : System.Windows.Window, System.Windows.Markup.IComponentConnector
00041     {

```

```

00042
00043 #line 7 "..\..\..\SaveFile.xaml"
00044     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00045         "CA1823:AvoidUnusedPrivateFields")]
00046     internal System.Windows.Controls.Button SelectFile;
00047 #line default
00048 #line hidden
00049
00050
00051 #line 8 "..\..\..\SaveFile.xaml"
00052     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00053         "CA1823:AvoidUnusedPrivateFields")]
00054     internal System.Windows.Controls.TextBox FilePath;
00055 #line default
00056 #line hidden
00057
00058
00059 #line 9 "..\..\..\SaveFile.xaml"
00060     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00061         "CA1823:AvoidUnusedPrivateFields")]
00062     internal System.Windows.Controls.TextBlock PathText;
00063 #line default
00064 #line hidden
00065
00066
00067 #line 10 "..\..\..\SaveFile.xaml"
00068     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00069         "CA1823:AvoidUnusedPrivateFields")]
00070     internal System.Windows.Controls.Button CancelButton;
00071 #line default
00072 #line hidden
00073
00074
00075 #line 11 "..\..\..\SaveFile.xaml"
00076     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00077         "CA1823:AvoidUnusedPrivateFields")]
00078     internal System.Windows.Controls.Button LoadButton;
00079 #line default
00080 #line hidden
00081
00082     private bool _contentLoaded;
00083
00084     /// <summary>
00085     /// InitializeComponent
00086     /// </summary>
00087     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00088     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00089     public void InitializeComponent() {
00090         if (_contentLoaded) {
00091             return;
00092         }
00093         _contentLoaded = true;
00094         System.Uri resourceLocator = new System.Uri("/Emulator;component/savefile.xaml",
00095             System.UriKind.Relative);
00096 #line 1 "..\..\..\SaveFile.xaml"
00097         System.Windows.Application.LoadComponent(this, resourceLocator);
00098 #line default
00099 #line hidden
00100     }
00101
00102
00103     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00104     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00105
00106     [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00107     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
00108         "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00109     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
00110         "CA1502:AvoidExcessiveComplexity")]
00111     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00112         "CA1800:DoNotCastUnnecessarily")]
00113     void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00114         switch (connectionId)
00115         {
00116             case 1:
00117                 this.SelectFile = ((System.Windows.Controls.Button) (target));
00118                 return;
00119             case 2:
00120                 this.FilePath = ((System.Windows.Controls.TextBox) (target));
00121                 return;
00122             case 3:

```

```

00119         this.PathText = ((System.Windows.Controls.TextBlock) (target));
00120         return;
00121     case 4:
00122         this.CancelButton = ((System.Windows.Controls.Button) (target));
00123         return;
00124     case 5:
00125         this.LoadButton = ((System.Windows.Controls.Button) (target));
00126         return;
00127     }
00128     this._contentLoaded = true;
00129 }
00130 }
00131 }
00132

```

7.115 Emulator/obj/x86/Release/SaveFile.g.i.cs File Reference

Classes

- class [Emulator.SaveFile](#)
SaveFile

Namespaces

- namespace [Emulator](#)

7.116 SaveFile.g.i.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\SaveFile.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00002 "34689CE75633CB3BE5E4FDF3C6E7ECDD6274F88E3F05662C41A2D31C677175A9"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037     /// <summary>
00038     /// SaveFile
00039     /// </summary>
00040     public partial class SaveFile : System.Windows.Window, System.Windows.Markup.IComponentConnector
00041     {

```

```

00042
00043 #line 7 "..\..\..\SaveFile.xaml"
00044     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00045         "CA1823:AvoidUnusedPrivateFields")]
00046     internal System.Windows.Controls.Button SelectFile;
00047 #line default
00048 #line hidden
00049
00050
00051 #line 8 "..\..\..\SaveFile.xaml"
00052     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00053         "CA1823:AvoidUnusedPrivateFields")]
00054     internal System.Windows.Controls.TextBox FilePath;
00055 #line default
00056 #line hidden
00057
00058
00059 #line 9 "..\..\..\SaveFile.xaml"
00060     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00061         "CA1823:AvoidUnusedPrivateFields")]
00062     internal System.Windows.Controls.TextBlock PathText;
00063 #line default
00064 #line hidden
00065
00066
00067 #line 10 "..\..\..\SaveFile.xaml"
00068     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00069         "CA1823:AvoidUnusedPrivateFields")]
00070     internal System.Windows.Controls.Button CancelButton;
00071 #line default
00072 #line hidden
00073
00074
00075 #line 11 "..\..\..\SaveFile.xaml"
00076     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00077         "CA1823:AvoidUnusedPrivateFields")]
00078     internal System.Windows.Controls.Button LoadButton;
00079 #line default
00080 #line hidden
00081
00082     private bool _contentLoaded;
00083
00084     /// <summary>
00085     /// InitializeComponent
00086     /// </summary>
00087     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00088     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00089     public void InitializeComponent() {
00090         if (_contentLoaded) {
00091             return;
00092         }
00093         _contentLoaded = true;
00094         System.Uri resourceLocator = new System.Uri("/Emulator;component/savefile.xaml",
00095             System.UriKind.Relative);
00096 #line 1 "..\..\..\SaveFile.xaml"
00097         System.Windows.Application.LoadComponent(this, resourceLocator);
00098 #line default
00099 #line hidden
00100     }
00101
00102
00103     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00104     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00105
00106     [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00107     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
00108         "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00109     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
00110         "CA1502:AvoidExcessiveComplexity")]
00111     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00112         "CA1800:DoNotCastUnnecessarily")]
00113     void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00114         switch (connectionId)
00115         {
00116             case 1:
00117                 this.SelectFile = ((System.Windows.Controls.Button) (target));
00118                 return;
00119             case 2:
00120                 this.FilePath = ((System.Windows.Controls.TextBox) (target));
00121                 return;
00122             case 3:

```

```

00119         this.PathText = ((System.Windows.Controls.TextBlock) (target));
00120         return;
00121     case 4:
00122         this.CancelButton = ((System.Windows.Controls.Button) (target));
00123         return;
00124     case 5:
00125         this.LoadButton = ((System.Windows.Controls.Button) (target));
00126         return;
00127     }
00128     this._contentLoaded = true;
00129 }
00130 }
00131 }
00132

```

7.117 Emulator/obj/x86/Debug/Settings.g.cs File Reference

Classes

- class [Emulator.Settings](#)
Settings

Namespaces

- namespace [Emulator](#)

7.118 Settings.g.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\Settings.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00002 "5C331E215A507ACA3F7FF07CFD574A81287117C06061A7F3A96858A63F0BA78B"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037     /// <summary>
00038     /// Settings
00039     /// </summary>
00040     public partial class Settings : System.Windows.Window, System.Windows.Markup.IComponentConnector
00041     {

```

```

00042
00043 #line 7 "..\..\..\Settings.xaml"
00044 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00045     internal System.Windows.Controls.ComboBox ComPortCombo;
00046
00047 #line default
00048 #line hidden
00049
00050
00051 #line 8 "..\..\..\Settings.xaml"
00052 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00053     internal System.Windows.Controls.TextBlock PortText;
00054
00055 #line default
00056 #line hidden
00057
00058
00059 #line 9 "..\..\..\Settings.xaml"
00060 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00061     internal System.Windows.Controls.Button ApplyButton;
00062
00063 #line default
00064 #line hidden
00065
00066
00067 #line 10 "..\..\..\Settings.xaml"
00068 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00069     internal System.Windows.Controls.Button CloseButton;
00070
00071 #line default
00072 #line hidden
00073
00074     private bool _contentLoaded;
00075
00076     /// <summary>
00077     /// InitializeComponent
00078     /// </summary>
00079     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00080     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00081     public void InitializeComponent() {
00082         if (_contentLoaded) {
00083             return;
00084         }
00085         _contentLoaded = true;
00086         System.Uri resourceLocater = new System.Uri("/Emulator;component/settings.xaml",
System.UriKind.Relative);
00087
00088 #line 1 "..\..\..\Settings.xaml"
00089         System.Windows.Application.LoadComponent(this, resourceLocater);
00090
00091 #line default
00092 #line hidden
00093     }
00094
00095     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00096     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00097
00098     [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00099     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
"CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00100     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
"CA1502:AvoidExcessiveComplexity")]
00101     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1800:DoNotCastUnnecessarily")]
00102     void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00103         switch (connectionId)
00104         {
00105             case 1:
00106                 this.ComPortCombo = ((System.Windows.Controls.ComboBox) (target));
00107
00108 #line 7 "..\..\..\Settings.xaml"
00109                 this.ComPortCombo.DropDownClosed += new
System.EventHandler(this.PortSelectionDropDownClosed);
00110
00111 #line default
00112 #line hidden
00113                 return;
00114             case 2:
00115                 this.PortText = ((System.Windows.Controls.TextBlock) (target));
00116                 return;
00117             case 3:
00118                 this.ApplyButton = ((System.Windows.Controls.Button) (target));
00119                 return;

```

```

00119         case 4:
00120             this.CloseButton = ((System.Windows.Controls.Button) (target));
00121             return;
00122         }
00123         this._contentLoaded = true;
00124     }
00125 }
00126 }
00127

```

7.119 Emulator/obj/x86/Publish/Settings.g.cs File Reference

Classes

- class [Emulator.Settings](#)
Settings

Namespaces

- namespace [Emulator](#)

7.120 Settings.g.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\Settings.xml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00002     "5C331E215A507ACA3F7FF07CFD574A81287117C06061A7F3A96858A63F0BA78B"
00003 //-----
00004 // <auto-generated>
00005 //     This code was generated by a tool.
00006 //     Runtime Version:4.0.30319.42000
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033 namespace Emulator {
00034     /// <summary>
00035     /// Settings
00036     /// </summary>
00037     public partial class Settings : System.Windows.Window, System.Windows.Markup.IComponentConnector
00038     {
00039         #line 7 "..\..\..\Settings.xml"
00040         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00041             "CA1823:AvoidUnusedPrivateFields")]
00042         internal System.Windows.Controls.ComboBox ComPortCombo;
00043

```



```

00046
00047 #line default
00048 #line hidden
00049
00050
00051 #line 8 "..\..\..\Settings.xaml"
00052 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00053     internal System.Windows.Controls.TextBlock PortText;
00054
00055 #line default
00056 #line hidden
00057
00058
00059 #line 9 "..\..\..\Settings.xaml"
00060 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00061     internal System.Windows.Controls.Button ApplyButton;
00062
00063 #line default
00064 #line hidden
00065
00066
00067 #line 10 "..\..\..\Settings.xaml"
00068 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00069     internal System.Windows.Controls.Button CloseButton;
00070
00071 #line default
00072 #line hidden
00073
00074     private bool _contentLoaded;
00075
00076 /// <summary>
00077 /// InitializeComponent
00078 /// </summary>
00079 [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00080 [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00081 public void InitializeComponent() {
00082     if (_contentLoaded) {
00083         return;
00084     }
00085     _contentLoaded = true;
00086     System.Uri resourceLocater = new System.Uri("/Emulator;component/settings.xaml",
System.UriKind.Relative);
00087
00088 #line 1 "..\..\..\Settings.xaml"
00089     System.Windows.Application.LoadComponent(this, resourceLocater);
00090
00091 #line default
00092 #line hidden
00093 }
00094
00095 [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00096 [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00097
00098 [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00099 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
"CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00100 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
"CA1502:AvoidExcessiveComplexity")]
00101 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1800:DoNotCastUnnecessarily")]
00102 void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00103     switch (connectionId)
00104     {
00105         case 1:
00106             this.ComPortCombo = ((System.Windows.Controls.ComboBox) (target));
00107
00108 #line 7 "..\..\..\Settings.xaml"
00109             this.ComPortCombo.DropDownClosed += new
System.EventHandler(this.PortSelectionDropDownClosed);
00110
00111 #line default
00112 #line hidden
00113             return;
00114         case 2:
00115             this.PortText = ((System.Windows.Controls.TextBlock) (target));
00116             return;
00117         case 3:
00118             this.ApplyButton = ((System.Windows.Controls.Button) (target));
00119             return;
00120         case 4:
00121             this.CloseButton = ((System.Windows.Controls.Button) (target));
00122             return;
00123     }
00124     this._contentLoaded = true;

```

```

00124     }
00125     }
00126 }
00127

```

7.121 Emulator/obj/x86/Release/Settings.g.cs File Reference

Classes

- class [Emulator.Settings](#)
Settings

Namespaces

- namespace [Emulator](#)

7.122 Settings.g.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\Settings.xml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
      "5C331E215A507ACA3F7FF07CFD574A81287117C06061A7F3A96858A63F0BA78B"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037     /// <summary>
00038     /// Settings
00039     /// </summary>
00040     public partial class Settings : System.Windows.Window, System.Windows.Markup.IComponentConnector
00041     {
00042
00043         #line 7 "..\..\..\Settings.xml"
00044         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00045             "CA1823:AvoidUnusedPrivateFields")]
00046         internal System.Windows.Controls.ComboBox ComPortCombo;
00047
00048         #line default
00049         #line hidden
00050

```

```

00051 #line 8 "..\..\..\Settings.xaml"
00052     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00053         "CA1823:AvoidUnusedPrivateFields")]
00054     internal System.Windows.Controls.TextBlock PortText;
00055 #line default
00056 #line hidden
00057
00058
00059 #line 9 "..\..\..\Settings.xaml"
00060     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00061         "CA1823:AvoidUnusedPrivateFields")]
00062     internal System.Windows.Controls.Button ApplyButton;
00063 #line default
00064 #line hidden
00065
00066
00067 #line 10 "..\..\..\Settings.xaml"
00068     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00069         "CA1823:AvoidUnusedPrivateFields")]
00070     internal System.Windows.Controls.Button CloseButton;
00071 #line default
00072 #line hidden
00073
00074     private bool _contentLoaded;
00075
00076     /// <summary>
00077     /// InitializeComponent
00078     /// </summary>
00079     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00080     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00081     public void InitializeComponent() {
00082         if (_contentLoaded) {
00083             return;
00084         }
00085         _contentLoaded = true;
00086         System.Uri resourceLocater = new System.Uri("/Emulator;component/settings.xaml",
00087             System.UriKind.Relative);
00088 #line 1 "..\..\..\Settings.xaml"
00089         System.Windows.Application.LoadComponent(this, resourceLocater);
00090 #line default
00091 #line hidden
00092     }
00093
00094
00095     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00096     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00097
00098     [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00099     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
00100         "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00101     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
00102         "CA1502:AvoidExcessiveComplexity")]
00103     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00104         "CA1800:DoNotCastUnnecessarily")]
00105     void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00106         switch (connectionId)
00107         {
00108             case 1:
00109                 this.ComPortCombo = ((System.Windows.Controls.ComboBox)(target));
00110 #line 7 "..\..\..\Settings.xaml"
00111                 this.ComPortCombo.DropDownClosed += new
00112                     System.EventHandler(this.PortSelectionDropDownClosed);
00113 #line default
00114 #line hidden
00115                 return;
00116             case 2:
00117                 this.PortText = ((System.Windows.Controls.TextBlock)(target));
00118                 return;
00119             case 3:
00120                 this.ApplyButton = ((System.Windows.Controls.Button)(target));
00121                 return;
00122             case 4:
00123                 this.CloseButton = ((System.Windows.Controls.Button)(target));
00124                 return;
00125             }
00126         this._contentLoaded = true;
00127     }

```

7.123 Emulator/obj/x86/Debug/Settings.g.i.cs File Reference

Classes

- class [Emulator.Settings](#)
Settings

Namespaces

- namespace [Emulator](#)

7.124 Settings.g.i.cs

[Go to the documentation of this file.](#)

```
00001 #pragma checksum "..\..\..\Settings.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
      "5C331E215A507ACA3F7FF07CFD574A81287117C06061A7F3A96858A63F0BA78B"
00002 //-----
00003 // <auto-generated>
00004 //      This code was generated by a tool.
00005 //      Runtime Version:4.0.30319.42000
00006 //
00007 //      Changes to this file may cause incorrect behavior and will be lost if
00008 //      the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037     /// <summary>
00038     /// Settings
00039     /// </summary>
00040     public partial class Settings : System.Windows.Window, System.Windows.Markup.IComponentConnector
00041     {
00042
00043         #line 7 "..\..\..\Settings.xaml"
00044         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00045             "CA1823:AvoidUnusedPrivateFields")]
00046         internal System.Windows.Controls.ComboBox ComPortCombo;
00047
00048         #line default
00049         #line hidden
00050
00051         #line 8 "..\..\..\Settings.xaml"
00052         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00053             "CA1823:AvoidUnusedPrivateFields")]
00054         internal System.Windows.Controls.TextBlock PortText;
00055
00056         #line default
00057         #line hidden
00058     }
00059 }
```

```

00058
00059 #line 9 "..\..\..\Settings.xaml"
00060 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00061     internal System.Windows.Controls.Button ApplyButton;
00062
00063 #line default
00064 #line hidden
00065
00066
00067 #line 10 "..\..\..\Settings.xaml"
00068 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00069     internal System.Windows.Controls.Button CloseButton;
00070
00071 #line default
00072 #line hidden
00073
00074     private bool _contentLoaded;
00075
00076     /// <summary>
00077     /// InitializeComponent
00078     /// </summary>
00079     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00080     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00081     public void InitializeComponent() {
00082         if (_contentLoaded) {
00083             return;
00084         }
00085         _contentLoaded = true;
00086         System.Uri resourceLocater = new System.Uri("/Emulator;component/settings.xaml",
System.UriKind.Relative);
00087
00088 #line 1 "..\..\..\Settings.xaml"
00089         System.Windows.Application.LoadComponent(this, resourceLocater);
00090
00091 #line default
00092 #line hidden
00093     }
00094
00095     [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00096     [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00097
00098     [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00099     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
"CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00100     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
"CA1502:AvoidExcessiveComplexity")]
00101     [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1800:DoNotCastUnnecessarily")]
00102     void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00103         switch (connectionId)
00104         {
00105             case 1:
00106                 this.ComPortCombo = ((System.Windows.Controls.ComboBox) (target));
00107
00108 #line 7 "..\..\..\Settings.xaml"
00109                 this.ComPortCombo.DropDownClosed += new
System.EventHandler(this.PortSelectionDropDownClosed);
00110
00111 #line default
00112 #line hidden
00113                 return;
00114             case 2:
00115                 this.PortText = ((System.Windows.Controls.TextBlock) (target));
00116                 return;
00117             case 3:
00118                 this.ApplyButton = ((System.Windows.Controls.Button) (target));
00119                 return;
00120             case 4:
00121                 this.CloseButton = ((System.Windows.Controls.Button) (target));
00122                 return;
00123             }
00124         this._contentLoaded = true;
00125     }
00126 }
00127

```

7.125 Emulator/obj/x86/Publish/Settings.g.i.cs File Reference

Classes

- class [Emulator.Settings](#)

*Settings***Namespaces**

- namespace [Emulator](#)

7.126 Settings.g.i.cs

[Go to the documentation of this file.](#)

```
00001 #pragma checksum "..\..\..\Settings.xml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
      "5C331E215A507ACA3F7FF07CFD574A81287117C06061A7F3A96858A63F0BA78B"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037     /// <summary>
00038     /// Settings
00039     /// </summary>
00040     public partial class Settings : System.Windows.Window, System.Windows.Markup.IComponentConnector
00041     {
00042
00043         #line 7 "..\..\..\Settings.xml"
00044         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00045             "CA1823:AvoidUnusedPrivateFields")]
00046         internal System.Windows.Controls.ComboBox ComPortCombo;
00047
00048         #line default
00049         #line hidden
00050
00051         #line 8 "..\..\..\Settings.xml"
00052         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00053             "CA1823:AvoidUnusedPrivateFields")]
00054         internal System.Windows.Controls.TextBlock PortText;
00055
00056         #line default
00057         #line hidden
00058
00059         #line 9 "..\..\..\Settings.xml"
00060         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00061             "CA1823:AvoidUnusedPrivateFields")]
00062         internal System.Windows.Controls.Button ApplyButton;
00063
00064         #line default
00065         #line hidden
```

```

00065
00066
00067 #line 10 "..\..\..\Settings.xaml"
00068 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
00069     internal System.Windows.Controls.Button CloseButton;
00070
00071 #line default
00072 #line hidden
00073
00074     private bool _contentLoaded;
00075
00076 /// <summary>
00077 /// InitializeComponent
00078 /// </summary>
00079 [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00080 [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00081 public void InitializeComponent() {
00082     if (_contentLoaded) {
00083         return;
00084     }
00085     _contentLoaded = true;
00086     System.Uri resourceLocater = new System.Uri("/Emulator;component/settings.xaml",
System.UriKind.Relative);
00087
00088 #line 1 "..\..\..\Settings.xaml"
00089     System.Windows.Application.LoadComponent(this, resourceLocater);
00090
00091 #line default
00092 #line hidden
00093     }
00094
00095 [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00096 [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00097 [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00098 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
"CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00099 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
"CA1502:AvoidExcessiveComplexity")]
00100 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1800:DoNotCastUnnecessarily")]
00101 void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00102     switch (connectionId)
00103     {
00104         case 1:
00105             this.ComPortCombo = ((System.Windows.Controls.ComboBox) (target));
00106
00107 #line 7 "..\..\..\Settings.xaml"
00108             this.ComPortCombo.DropDownClosed += new
System.EventHandler(this.PortSelectionDropDownClosed);
00109
00110 #line default
00111 #line hidden
00112             return;
00113         case 2:
00114             this.PortText = ((System.Windows.Controls.TextBlock) (target));
00115             return;
00116         case 3:
00117             this.ApplyButton = ((System.Windows.Controls.Button) (target));
00118             return;
00119         case 4:
00120             this.CloseButton = ((System.Windows.Controls.Button) (target));
00121             return;
00122     }
00123     this._contentLoaded = true;
00124 }
00125 }
00126 }
00127

```

7.127 Emulator/obj/x86/Release/Settings.g.i.cs File Reference

Classes

- class [Emulator.Settings](#)
[Settings](#)

Namespaces

- namespace [Emulator](#)

7.128 Settings.g.i.cs

[Go to the documentation of this file.](#)

```

00001 #pragma checksum "..\..\..\Settings.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
      "5C331E215A507ACA3F7FF07CFD574A81287117C06061A7F3A96858A63F0BA78B"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using System;
00013 using System.Diagnostics;
00014 using System.Windows;
00015 using System.Windows.Automation;
00016 using System.Windows.Controls;
00017 using System.Windows.Controls.Primitives;
00018 using System.Windows.Data;
00019 using System.Windows.Documents;
00020 using System.Windows.Ink;
00021 using System.Windows.Input;
00022 using System.Windows.Markup;
00023 using System.Windows.Media;
00024 using System.Windows.Media.Animation;
00025 using System.Windows.Media.Effects;
00026 using System.Windows.Media.Imaging;
00027 using System.Windows.Media.Media3D;
00028 using System.Windows.Media.TextFormatting;
00029 using System.Windows.Navigation;
00030 using System.Windows.Shapes;
00031 using System.Windows.Shell;
00032
00033
00034 namespace Emulator {
00035
00036
00037     /// <summary>
00038     /// Settings
00039     /// </summary>
00040     public partial class Settings : System.Windows.Window, System.Windows.Markup.IComponentConnector
00041     {
00042
00043         #line 7 "..\..\..\Settings.xaml"
00044         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00045             "CA1823:AvoidUnusedPrivateFields")]
00046         internal System.Windows.Controls.ComboBox ComPortCombo;
00047
00048         #line default
00049         #line hidden
00050
00051         #line 8 "..\..\..\Settings.xaml"
00052         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00053             "CA1823:AvoidUnusedPrivateFields")]
00054         internal System.Windows.Controls.TextBlock PortText;
00055
00056         #line default
00057         #line hidden
00058
00059         #line 9 "..\..\..\Settings.xaml"
00060         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00061             "CA1823:AvoidUnusedPrivateFields")]
00062         internal System.Windows.Controls.Button ApplyButton;
00063
00064         #line default
00065         #line hidden
00066
00067         #line 10 "..\..\..\Settings.xaml"
00068         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00069             "CA1823:AvoidUnusedPrivateFields")]

```



```

00069         internal System.Windows.Controls.Button CloseButton;
00070
00071 #line default
00072 #line hidden
00073
00074         private bool _contentLoaded;
00075
00076 /// <summary>
00077 /// InitializeComponent
00078 /// </summary>
00079 [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00080 [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00081 public void InitializeComponent() {
00082     if (_contentLoaded) {
00083         return;
00084     }
00085     _contentLoaded = true;
00086     System.Uri resourceLocater = new System.Uri("/Emulator;component/settings.xaml",
System.UriKind.Relative);
00087
00088 #line 1 "..\\..\\..\\Settings.xaml"
00089     System.Windows.Application.LoadComponent(this, resourceLocater);
00090
00091 #line default
00092 #line hidden
00093 }
00094
00095 [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00096 [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00097
00098 [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00099 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
"CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00100 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
"CA1502:AvoidExcessiveComplexity")]
00101 [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1800:DoNotCastUnnecessarily")]
00102 void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00103     switch (connectionId)
00104     {
00105         case 1:
00106             this.ComPortCombo = ((System.Windows.Controls.ComboBox) (target));
00107
00108 #line 7 "..\\..\\..\\Settings.xaml"
00109             this.ComPortCombo.DropDownClosed += new
System.EventHandler(this.PortSelectionDropDownClosed);
00110
00111 #line default
00112 #line hidden
00113             return;
00114         case 2:
00115             this.PortText = ((System.Windows.Controls.TextBlock) (target));
00116             return;
00117         case 3:
00118             this.ApplyButton = ((System.Windows.Controls.Button) (target));
00119             return;
00120         case 4:
00121             this.CloseButton = ((System.Windows.Controls.Button) (target));
00122             return;
00123     }
00124     this._contentLoaded = true;
00125 }
00126 }
00127

```

7.129 Emulator/obj/x86/Release/MemoryMap.g.i.cs File Reference

Classes

- class [Emulator.Window1](#)
Window1

Namespaces

- namespace [Emulator](#)

7.130 MemoryMap.g.i.cs

Go to the documentation of this file.

```

00001 #pragma checksum "..\..\..\MemoryMap.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
      "3CB2ED3482F456B951057C8C46A6AC7CA3E26CA8E4C01AE79554E453DC281448"
00002 //-----
00003 // <auto-generated>
00004 //     This code was generated by a tool.
00005 //     Runtime Version:4.0.30319.42000
00006 //
00007 //     Changes to this file may cause incorrect behavior and will be lost if
00008 //     the code is regenerated.
00009 // </auto-generated>
00010 //-----
00011
00012 using Emulator;
00013 using System;
00014 using System.Diagnostics;
00015 using System.Windows;
00016 using System.Windows.Automation;
00017 using System.Windows.Controls;
00018 using System.Windows.Controls.Primitives;
00019 using System.Windows.Data;
00020 using System.Windows.Documents;
00021 using System.Windows.Ink;
00022 using System.Windows.Input;
00023 using System.Windows.Markup;
00024 using System.Windows.Media;
00025 using System.Windows.Media.Animation;
00026 using System.Windows.Media.Effects;
00027 using System.Windows.Media.Imaging;
00028 using System.Windows.Media.Media3D;
00029 using System.Windows.Media.TextFormatting;
00030 using System.Windows.Navigation;
00031 using System.Windows.Shapes;
00032 using System.Windows.Shell;
00033
00034
00035 namespace Emulator {
00036
00037
00038     /// <summary>
00039     /// Window1
00040     /// </summary>
00041     public partial class Window1 : System.Windows.Window, System.Windows.Markup.IComponentConnector {
00042
00043
00044         #line 49 "..\..\..\MemoryMap.xaml"
00045         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00046             "CA1823:AvoidUnusedPrivateFields")]
00047         internal System.Windows.Controls.DataGrid MemoryMap;
00048
00049         #line default
00050         #line hidden
00051         private bool _contentLoaded;
00052
00053         /// <summary>
00054         /// InitializeComponent
00055         /// </summary>
00056         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00057         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00058         public void InitializeComponent() {
00059             if (_contentLoaded) {
00060                 return;
00061             }
00062             _contentLoaded = true;
00063             System.Uri resourceLocator = new System.Uri("/Emulator;component/memorymap.xaml",
00064                 System.UriKind.Relative);
00065
00066             #line 1 "..\..\..\MemoryMap.xaml"
00067             System.Windows.Application.LoadComponent(this, resourceLocator);
00068
00069             #line default
00070             #line hidden
00071         }
00072
00073         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00074         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00075         [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00076         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
00077             "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00078         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
00079             "CA1502:AvoidExcessiveComplexity")]

```

```

00077         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
00078         "CA1800:DoNotCastUnnecessarily")]
00078     void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
00079         switch (connectionId)
00080         {
00081             case 1:
00082                 this.MemoryMap = ((System.Windows.Controls.DataGrid) (target));
00083                 return;
00084             }
00085         this._contentLoaded = true;
00086     }
00087 }
00088 }
00089

```

7.131 Emulator/obj/x86/Release/Window1.g.i.cs File Reference

Classes

- class [Emulator.Window1](#)
Window1

Namespaces

- namespace [Emulator](#)

7.132 Window1.g.i.cs

Go to the [documentation of this file](#).

```

00001 // Updated by XamlIntelliSenseFileGenerator 25/09/2022 10:56:46
00002 #pragma checksum "..\..\..\Window1.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
00003         "4BA73942B3E4CC642C22E491C94CD66BBC88A7F17A65FB594B61788FC9DFDB08"
00004 //-----
00004 // <auto-generated>
00005 //     This code was generated by a tool.
00006 //     Runtime Version:4.0.30319.42000
00007 //
00008 //     Changes to this file may cause incorrect behavior and will be lost if
00009 //     the code is regenerated.
00010 // </auto-generated>
00011 //-----
00012
00013 using Emulator;
00014 using System;
00015 using System.Diagnostics;
00016 using System.Windows;
00017 using System.Windows.Automation;
00018 using System.Windows.Controls;
00019 using System.Windows.Controls.Primitives;
00020 using System.Windows.Data;
00021 using System.Windows.Documents;
00022 using System.Windows.Ink;
00023 using System.Windows.Input;
00024 using System.Windows.Markup;
00025 using System.Windows.Media;
00026 using System.Windows.Media.Animation;
00027 using System.Windows.Media.Effects;
00028 using System.Windows.Media.Imaging;
00029 using System.Windows.Media.Media3D;
00030 using System.Windows.Media.TextFormatting;
00031 using System.Windows.Navigation;
00032 using System.Windows.Shapes;
00033 using System.Windows.Shell;
00034
00035
00036 namespace Emulator
00037 {
00038
00039
00040     /// <summary>
00041     /// Window1
00042     /// </summary>

```

```

00043     public partial class Window1 : System.Windows.Window, System.Windows.Markup.IComponentConnector
00044     {
00045
00046         private bool _contentLoaded;
00047
00048         /// <summary>
00049         /// InitializeComponent
00050         /// </summary>
00051         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00052         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00053         public void InitializeComponent()
00054         {
00055             if (_contentLoaded)
00056             {
00057                 return;
00058             }
00059             _contentLoaded = true;
00060             System.Uri resourceLocater = new System.Uri("/Emulator;component/window1.xaml",
System.UriKind.Relative);
00061
00062             #line 1 "..\..\..\Window1.xaml"
00063             System.Windows.Application.LoadComponent(this, resourceLocater);
00064
00065             #line default
00066             #line hidden
00067         }
00068
00069         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00070         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00071
00072         [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00073         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
"CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00074         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
"CA1502:AvoidExcessiveComplexity")]
00075         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1800:DoNotCastUnnecessarily")]
00076         void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target)
00077         {
00078             this._contentLoaded = true;
00079         }
00080         internal System.Windows.Controls.DataGrid MemoryMap;
00081     }
00082 }
00083

```

7.133 Emulator/Properties/AssemblyInfo.cs File Reference

7.134 AssemblyInfo.cs

[Go to the documentation of this file.](#)

```

00001 using Emulator;
00002 using System.Reflection;
00003 using System.Resources;
00004 using System.Runtime.InteropServices;
00005 using System.Windows;
00006
00007 // General Information about an assembly is controlled through the following
00008 // set of attributes. Change these attribute values to modify the information
00009 // associated with an assembly.
00010 [assembly: AssemblyTitle(Versioning.Product.Title)]
00011 [assembly: AssemblyDescription(Versioning.Product.Description)]
00012 [assembly: AssemblyConfiguration("")]
00013 [assembly: AssemblyCompany(Versioning.Product.Company)]
00014 [assembly: AssemblyProduct(Versioning.Product.Name)]
00015 [assembly: AssemblyCopyright(Versioning.Product.Copyright)]
00016 [assembly: AssemblyTrademark("")]
00017 [assembly: AssemblyCulture("")]
00018
00019 // Setting ComVisible to false makes the types in this assembly not visible
00020 // to COM components. If you need to access a type in this assembly from
00021 // COM, set the ComVisible attribute to true on that type.
00022 [assembly: ComVisible(false)]
00023
00024 //In order to begin building localizable applications, set
00025 //<UICulture>CultureYouAreCodingWith</UICulture> in your .csproj file
00026 //inside a <PropertyGroup>. For example, if you are using US english
00027 //in your source files, set the <UICulture> to en-US. Then uncomment
00028 //the NeutralResourceLanguage attribute below. Update the "en-US" in

```

```

00029 //the line below to match the UICulture setting in the project file.
00030
00031 //[assembly: NeutralResourcesLanguage("en-US", UltimateResourceFallbackLocation.Satellite)]
00032
00033
00034 [assembly: ThemeInfo(
00035     ResourceDictionaryLocation.None, //where theme specific resource dictionaries are located
00036                                     //(used if a resource is not found in the page,
00037                                     // or application resource dictionaries)
00038     ResourceDictionaryLocation.SourceAssembly //where the generic resource dictionary is located
00039                                               //(used if a resource is not found in the page,
00040                                               // app, or any theme specific resource dictionaries)
00041 )]
00042
00043
00044 // Version information for an assembly consists of the following four values:
00045 //
00046 //         Major Version
00047 //         Minor Version
00048 //         Build Number
00049 //         Revision
00050 //
00051 // You can specify all the values or you can default the Build and Revision Numbers
00052 // by using the '*' as shown below:
00053 // [assembly: AssemblyVersion("1.0.*")]
00054 [assembly: AssemblyVersion(Versioning.Product.VersionString)]
00055 [assembly: AssemblyFileVersion(Versioning.Product.VersionString)]
00056 [assembly: NeutralResourcesLanguage("en-GB")]

```

7.135 Hardware/Properties/AssemblyInfo.cs File Reference

7.136 AssemblyInfo.cs

[Go to the documentation of this file.](#)

```

00001 using Hardware;
00002 using System.Reflection;
00003 using System.Resources;
00004 using System.Runtime.InteropServices;
00005
00006 // General Information about an assembly is controlled through the following
00007 // set of attributes. Change these attribute values to modify the information
00008 // associated with an assembly.
00009 [assembly: AssemblyTitle(Versioning.Product.Title)]
00010 [assembly: AssemblyDescription(Versioning.Product.Description)]
00011 [assembly: AssemblyConfiguration("")]
00012 [assembly: AssemblyCompany(Versioning.Product.Company)]
00013 [assembly: AssemblyProduct("")]
00014 [assembly: AssemblyCopyright(Versioning.Product.Copyright)]
00015 [assembly: AssemblyTrademark("")]
00016 [assembly: AssemblyCulture("")]
00017
00018 // Setting ComVisible to false makes the types in this assembly not visible
00019 // to COM components. If you need to access a type in this assembly from
00020 // COM, set the ComVisible attribute to true on that type.
00021 [assembly: ComVisible(false)]
00022
00023 // The following GUID is for the ID of the typelib if this project is exposed to COM
00024 [assembly: Guid("f4afef76-2e8f-4497-86c6-c903aa70eebd")]
00025
00026 // Version information for an assembly consists of the following four values:
00027 //
00028 //         Major Version
00029 //         Minor Version
00030 //         Build Number
00031 //         Revision
00032 //
00033 // You can specify all the values or you can default the Build and Revision Numbers
00034 // by using the '*' as shown below:
00035 // [assembly: AssemblyVersion("1.0.*")]
00036 [assembly: AssemblyVersion(Versioning.Product.Version)]
00037 [assembly: AssemblyFileVersion(Versioning.Product.Version)]
00038 [assembly: NeutralResourcesLanguage("")]

```

7.137 Emulator/SaveFile.xaml.cs File Reference

Classes

- class [Emulator.SaveFile](#)
[SaveFile](#)

Namespaces

- namespace [Emulator](#)

7.138 SaveFile.xaml.cs

[Go to the documentation of this file.](#)

```

00001 using GalaSoft.MvvmLight.Messaging;
00002
00003 namespace Emulator
00004 {
00005     /// <summary>
00006     /// Interaction logic for SaveState.xaml
00007     /// </summary>
00008     public partial class SaveFile
00009     {
00010         public SaveFile()
00011         {
00012             InitializeComponent();
00013             Messenger.Default.Register<NotificationMessage>(this, NotificationMessageReceived);
00014         }
00015
00016         private void NotificationMessageReceived(NotificationMessage notificationMessage)
00017         {
00018             if (notificationMessage.Notification == "CloseSaveFileWindow")
00019                 Close();
00020         }
00021     }
00022 }

```

7.139 Emulator/Settings.xaml.cs File Reference**Classes**

- class [Emulator.Settings](#)
[Settings](#)

Namespaces

- namespace [Emulator](#)

7.140 Settings.xaml.cs

[Go to the documentation of this file.](#)

```

00001 using Emulator.Model;
00002 using Emulator.ViewModel;
00003 using GalaSoft.MvvmLight.Messaging;
00004 using System;
00005
00006 namespace Emulator
00007 {
00008     /// <summary>
00009     /// Interaction logic for Settings.xaml
00010     /// </summary>
00011     public partial class Settings
00012     {
00013         public Settings()
00014         {
00015             InitializeComponent();
00016             Messenger.Default.Register<NotificationMessage>(this, NotificationMessageReceived);
00017             Messenger.Default.Register<NotificationMessage<SettingsModel>>(this,
NotificationMessageReceived);
00018         }
00019
00020         private void NotificationMessageReceived(NotificationMessage notificationMessage)
00021         {

```

```

00022         if (notificationMessage.Notification == "CloseSettingsWindow")
00023         {
00024             Close();
00025         }
00026     }
00027
00028     private void NotificationMessageReceived(NotificationMessage<SettingsModel>
notificationMessage)
00029     {
00030         if (notificationMessage.Notification == "SettingsWindow")
00031         {
00032             SettingsViewModel.SettingsModel = notificationMessage.Content;
00033             ComPortCombo.SelectedItem = notificationMessage.Content.ComPortName;
00034         }
00035     }
00036
00037     private void PortSelectionDropDownClosed(object sender, EventArgs e)
00038     {
00039         if (!(ComPortCombo.SelectedValue == null))
00040         {
00041             string port = ComPortCombo.SelectedValue.ToString();
00042             SettingsViewModel.ComPortSelection = port;
00043         }
00044     }
00045 }
00046 }

```

7.141 Emulator/ViewModel/MainViewModel.cs File Reference

Classes

- class [Emulator.ViewModel.MainViewModel](#)
The Main [ViewModel](#)

Namespaces

- namespace [Emulator](#)
- namespace [Emulator.ViewModel](#)

Typedefs

- using [W65C02](#) = [Hardware.W65C02](#)
- using [W65C22](#) = [Hardware.W65C22](#)
- using [W65C51](#) = [Hardware.W65C51](#)

7.141.1 Typedef Documentation

7.141.1.1 W65C02 using [W65C02](#) = [Hardware.W65C02](#)

Definition at line 17 of file [MainViewModel.cs](#).

7.141.1.2 W65C22 using [W65C22](#) = [Hardware.W65C22](#)

Definition at line 18 of file [MainViewModel.cs](#).

7.141.1.3 W65C51 `using W65C51 = Hardware.W65C51`

Definition at line 19 of file [MainViewModel.cs](#).

7.142 MainViewModel.cs

[Go to the documentation of this file.](#)

```

00001 using Emulator.Model;
00002 using GalaSoft.MvvmLight;
00003 using GalaSoft.MvvmLight.Command;
00004 using GalaSoft.MvvmLight.Messaging;
00005 using Hardware;
00006 using Microsoft.Win32;
00007 using System;
00008 using System.Collections.Generic;
00009 using System.ComponentModel;
00010 using System.Globalization;
00011 using System.IO;
00012 using System.Linq;
00013 using System.Runtime.Serialization.Formatters.Binary;
00014 using System.Threading;
00015 using System.Windows;
00016 using System.Xml.Serialization;
00017 using W65C02 = Hardware.W65C02;
00018 using W65C22 = Hardware.W65C22;
00019 using W65C51 = Hardware.W65C51;
00020
00021 namespace Emulator.ViewModel
00022 {
00023     /// <summary>
00024     /// The Main ViewModel
00025     /// </summary>
00026     public class MainViewModel : ViewModelBase
00027     {
00028         #region Fields
00029         private readonly BackgroundWorker _backgroundWorker;
00030         private bool _breakpointTriggered;
00031     #endregion
00032
00033     #region Properties
00034     /// <summary>
00035     /// The 62256 RAM.
00036     /// </summary>
00037     private HM62256 HM62256 { get; set; }
00038
00039     /// <summary>
00040     /// The 65C02 Processor.
00041     /// </summary>
00042     public W65C02 W65C02 { get; private set; }
00043
00044     /// <summary>
00045     /// General Purpose I/O, Shift Registers and Timers.
00046     /// </summary>
00047     public W65C22 W65C22 { get; private set; }
00048
00049     /// <summary>
00050     /// Memory management and 65SIB.
00051     /// </summary>
00052     public W65C22 MM65SIB { get; private set; }
00053
00054     /// <summary>
00055     /// The ACIA serial interface.
00056     /// </summary>
00057     public W65C51 W65C51 { get; private set; }
00058
00059     /// <summary>
00060     /// The AT28C010 ROM.
00061     /// </summary>
00062     public AT28CXX AT28C64 { get; private set; }
00063
00064     /// <summary>
00065     /// The AT28C010 ROM.
00066     /// </summary>
00067     public AT28CXX AT28C010 { get; private set; }
00068
00069     /// <summary>
00070     /// The Current Memory Page
00071     /// </summary>
00072     public MultiThreadedObservableCollection<MemoryRowModel> MemoryPage { get; set; }
00073
00074     /// <summary>

```



```

00075 /// The output log
00076 /// </summary>
00077     public MultiThreadedObservableCollection<OutputLog> OutputLog { get; private set; }
00078
00079 /// <summary>
00080 /// The Breakpoints
00081 /// </summary>
00082     public MultiThreadedObservableCollection<Breakpoint> Breakpoints { get; set; }
00083
00084 /// <summary>
00085 /// The Currently Selected Breakpoint
00086 /// </summary>
00087     public Breakpoint SelectedBreakpoint { get; set; }
00088
00089 /// <summary>
00090 /// The currently loaded binary file. (If it is indeed loaded, that is.)
00091 /// </summary>
00092     public RomFileModel RomFile { get; set; }
00093
00094 /// <summary>
00095 /// The Current Disassembly
00096 /// </summary>
00097     public string CurrentDisassembly
00098     {
00099         get
00100         {
00101             if (W65C02.CurrentDisassembly != null)
00102             {
00103                 return string.Format("{0} {1}", W65C02.CurrentDisassembly.OpCodeString,
W65C02.CurrentDisassembly.DisassemblyOutput);
00104             }
00105             else
00106             {
00107                 return string.Empty;
00108             }
00109         }
00110     }
00111
00112 /// <summary>
00113 /// The number of cycles.
00114 /// </summary>
00115     public int NumberOfCycles { get; private set; }
00116
00117 /// <summary>
00118 /// Is the Program Running
00119 /// </summary>
00120     public bool IsRunning
00121     {
00122         get { return W65C02.IsRunning; }
00123         set
00124         {
00125             W65C02.IsRunning = value;
00126             RaisePropertyChanged("IsRunning");
00127         }
00128     }
00129
00130 /// <summary>
00131 /// Is the banked ROM Loaded.
00132 /// </summary>
00133     public bool IsRomLoaded { get; set; }
00134
00135 /// <summary>
00136 /// The Slider CPU Speed
00137 /// </summary>
00138     public int CpuSpeed { get; set; }
00139
00140 /// <summary>
00141 /// The Model used for saving, loading and using data from Settings.xml
00142 /// </summary>
00143     public static SettingsModel SettingsModel { get; set; }
00144
00145 /// <summary>
00146 /// RelayCommand for Stepping through the program one instruction at a time.
00147 /// </summary>
00148     public RelayCommand StepCommand { get; set; }
00149
00150 /// <summary>
00151 /// RelayCommand for opening the Memory View window.
00152 /// </summary>
00153     public RelayCommand MemoryVisualCommand { get; set; }
00154
00155 /// <summary>
00156 /// RelayCommand to Reset the Program back to its initial state.
00157 /// </summary>
00158     public RelayCommand ResetCommand { get; set; }
00159
00160 /// <summary>

```

```

00161 /// Relay Command that Run/Pauses Execution
00162 /// </summary>
00163     public RelayCommand RunPauseCommand { get; set; }
00164
00165     /// <summary>
00166     /// Relay Command that updates the Memory Map when the Page changes
00167     /// </summary>
00168     public RelayCommand UpdateMemoryMapCommand { get; set; }
00169
00170     /// <summary>
00171     /// The Relay Command that adds a new breakpoint
00172     /// </summary>
00173     public RelayCommand AddBreakPointCommand { get; set; }
00174
00175     /// <summary>
00176     /// The Relay Command that opens the About window.
00177     /// </summary>
00178     public RelayCommand AboutCommand { get; set; }
00179
00180     /// <summary>
00181     /// The Relay Command that Removes an existing breakpoint.
00182     /// </summary>
00183     public RelayCommand RemoveBreakPointCommand { get; set; }
00184
00185     /// <summary>
00186     /// The Command that loads or saves the settings.
00187     /// </summary>
00188     public RelayCommand SettingsCommand { get; set; }
00189
00190     /// <summary>
00191     /// The Command that loads or saves the settings.
00192     /// </summary>
00193     public RelayCommand<IClosable> CloseCommand { get; private set; }
00194
00195     /// <summary>
00196     /// The current serial port object name.
00197     /// </summary>
00198     public string CurrentSerialPort
00199     {
00200         get
00201         {
00202             return W65C51.ObjectName;
00203         }
00204     }
00205
00206     /// <summary>
00207     /// The title for the main window.
00208     /// </summary>
00209     public string WindowTitle { get { return Versioning.Product.Title; } }
00210 #endregion
00211
00212 #region public Methods
00213     /// <summary>
00214     /// Creates a new Instance of the MainViewModel.
00215     /// </summary>
00216     public MainViewModel()
00217     {
00218         var _formatter = new XmlSerializer(typeof(SettingsModel));
00219         Stream _stream = new FileStream(FileLocations.SettingsFile, FileMode.OpenOrCreate);
00220         if (!(_stream == null) || (0 >= _stream.Length))
00221         {
00222             SettingsModel = (SettingsModel)_formatter.Deserialize(_stream);
00223             if ((SettingsModel.SettingsVersionMajor < Versioning.SettingsFile.Major) ||
00224                 (SettingsModel.SettingsVersionMinor < Versioning.SettingsFile.Minor) ||
00225                 (SettingsModel.SettingsVersionBuild < Versioning.SettingsFile.Build) ||
00226                 (SettingsModel.SettingsVersionRevision < Versioning.SettingsFile.Revision))
00227             {
00228                 file...",
00229                 "Settings file stale!", MessageBoxButton.OKCancel,
00230                 MessageBoxImage.Warning,
00231                 MessageBoxResult.OK);
00232                 // Close the file, then delete it.
00233                 _stream.Close();
00234                 File.Delete(FileLocations.SettingsFile);
00235                 SettingsModel = SettingsFile.CreateNew();
00236             }
00237         }
00238         else
00239         {
00240             MessageBox.Show("Creating new settings file...");
00241             SettingsModel = SettingsFile.CreateNew();
00242         }
00243         _stream.Close();
00244         HM62256 = new HM62256(MemoryMap.BankedRam.TotalBanks, MemoryMap.BankedRam.Offset,
MemoryMap.BankedRam.Length);

```

```

00245         AT28C64 = new AT28CXX(MemoryMap.SharedRom.Offset, MemoryMap.SharedRom.Length, 1);
00246         AT28C010 = new AT28CXX(MemoryMap.BankedRom.Offset, MemoryMap.BankedRom.Length,
MemoryMap.BankedRom.TotalBanks);
00247         W65C02 = new W65C02();
00248         W65C51 = new W65C51(W65C02, MemoryMap.Devices.ACIA.Offset);
00249         W65C51.Init(SettingsModel.ComPortName.ToString());
00250         W65C22 = new W65C22(W65C02, MemoryMap.Devices.GPIO.Offset, MemoryMap.Devices.GPIO.Length);
00251         W65C22.Init(1000);
00252         MM65SIB = new W65C22(W65C02, MemoryMap.Devices.MM65SIB.Offset,
MemoryMap.Devices.MM65SIB.Length);
00253         MM65SIB.Init(1000);
00254
00255         MemoryMap.Init(W65C02, W65C22, MM65SIB, W65C51, HM62256, AT28C010, AT28C64);
00256
00257         // Now we can load the BIOS.
00258         byte[][] _bios = AT28C64.ReadFile(FileLocations.BiosFile);
00259         if (_bios == null)
00260         {
00261             Environment.Exit(ExitCodes.NO_BIOS);
00262         }
00263         AT28C64.Load(_bios);
00264
00265         AboutCommand = new RelayCommand(About);
00266         AddBreakPointCommand = new RelayCommand(AddBreakPoint);
00267         CloseCommand = new RelayCommand<IClosable>(Close);
00268         MemoryVisualCommand = new RelayCommand(MemoryView);
00269         RemoveBreakPointCommand = new RelayCommand(RemoveBreakPoint);
00270         ResetCommand = new RelayCommand(Reset);
00271         RunPauseCommand = new RelayCommand(RunPause);
00272         SettingsCommand = new RelayCommand(Settings);
00273         StepCommand = new RelayCommand(Step);
00274
00275         Messenger.Default.Register<NotificationMessage>(this, GenericNotification);
00276         Messenger.Default.Register<NotificationMessage<RomFileModel>>(this,
BinaryLoadedNotification);
00277         Messenger.Default.Register<NotificationMessage<SettingsModel>>(this,
SettingsAppliedNotification);
00278         Messenger.Default.Register<NotificationMessage<StateFileModel>>(this,
StateLoadedNotification);
00279
00280         MemoryPage = new MultiThreadedObservableCollection<MemoryRowModel>();
00281         OutputLog = new MultiThreadedObservableCollection<OutputLog>();
00282         Breakpoints = new MultiThreadedObservableCollection<Breakpoint>();
00283
00284         UpdateMemoryPage();
00285
00286         _backgroundWorker = new BackgroundWorker { WorkerSupportsCancellation = true,
WorkerReportsProgress = false };
00287         _backgroundWorker.DoWork += BackgroundWorkerDoWork;
00288         Application.Current.MainWindow.Title = Versioning.Product.Title;
00289         Application.Current.MainWindow.Closing += new CancelEventHandler(OnClose);
00290         Application.Current.MainWindow.Loaded += new RoutedEventHandler(OnLoad);
00291
00292         Reset();
00293     }
00294
00295     public void OnLoad(Object sender, RoutedEventArgs e)
00296     {
00297         #if !DEBUG
00298             if (Versioning.Product.Major < 1)
00299             {
00300                 var result = MessageBox.Show(String.Format("Thank you for using {0}\n" +
"Be warned that this is a beta build.\n" +
"It may break or have bugs.",
Versioning.Product.Name),
Versioning.Product.Title,
MessageBoxButton.OKCancel,
MessageBoxImage.Warning,
MessageBoxResult.None);
00305                 if (result == MessageBoxResult.Cancel)
00306                 {
00307                     // Exit without making any changes.
00308                     Environment.Exit(ExitCodes.NO_ERROR);
00309                 }
00310             }
00311         #endif
00312     }
00313
00314     public void OnClose(Object sender, CancelEventArgs e)
00315     {
00316         e.Cancel = false;
00317         if (IsRunning)
00318         {
00319             MessageBox.Show("You can't quit the emulator while it is actively running!",
"You can't do that!", MessageBoxButton.OK, MessageBoxImage.Stop);
00320             e.Cancel = true;
00321         }
00322         return;

```

```

00323     }
00324     #if !DEBUG
00325     else
00326     {
00327         var result = MessageBox.Show("Are you sure you want to quit the emulator?",
00328                                     "To quit, or not to quit -- that is the question.",
00329                                     MessageBoxButton.YesNo, MessageBoxImage.Question,
00330                                     MessageBoxResult.No);
00331         if (result == MessageBoxResult.No)
00332         {
00333             e.Cancel = true;
00334             return;
00335         }
00336     }
00337 #endif
00338     Stream stream = new FileStream(FileLocations.SettingsFile, FileMode.Create,
00339     FileAccess.Write, FileShare.None);
00339     XmlSerializer XmlFormatter = new XmlSerializer(typeof(SettingsModel));
00340     XmlFormatter.Serialize(stream, MainViewModel.SettingsModel);
00341     stream.Flush();
00342     stream.Close();
00343     W65C51.Fini();
00344 }
00345 #endregion
00346
00347 #region Private Methods
00348 private void Close(IClosable window)
00349 {
00350     if ((window != null) && (!IsRunning))
00351     {
00352         Environment.Exit(ExitCodes.NO_ERROR);
00353     }
00354 }
00355
00356 private void BinaryLoadedNotification(NotificationMessage<RomFileModel> notificationMessage)
00357 {
00358     if (notificationMessage.Notification != "FileLoaded")
00359     {
00360         return;
00361     }
00362
00363     // Load Banked ROM
00364     AT28C010.Load(notificationMessage.Content.Rom);
00365     IsRomLoaded = true;
00366     RaisePropertyChanged("IsRomLoaded");
00367
00368     Reset();
00369 }
00370
00371 private void StateLoadedNotification(NotificationMessage<StateFileModel> notificationMessage)
00372 {
00373     if (notificationMessage.Notification != "StateLoaded")
00374     {
00375         return;
00376     }
00377
00378     Reset();
00379
00380     OutputLog = new
00381     MultiThreadedObservableCollection<OutputLog>(notificationMessage.Content.OutputLog);
00382     RaisePropertyChanged("OutputLog");
00383
00384     NumberOfCycles = notificationMessage.Content.NumberOfCycles;
00385
00386     W65C02 = notificationMessage.Content.W65C02;
00387     W65C22 = notificationMessage.Content.W65C22;
00388     MM65SIB = notificationMessage.Content.MM65SIB;
00389     W65C51 = notificationMessage.Content.W65C51;
00390     AT28C010 = notificationMessage.Content.AT28C010;
00391     AT28C64 = notificationMessage.Content.AT28C64;
00392     UpdateMemoryPage();
00393     UpdateUi();
00394
00395     IsRomLoaded = true;
00396     RaisePropertyChanged("IsRomLoaded");
00397 }
00398 private void GenericNotification(NotificationMessage notificationMessage)
00399 {
00400     if (notificationMessage.Notification == "CloseFile")
00401     {
00402         AT28C010.Clear();
00403         if (IsRunning) { RunPause(); }
00404         IsRomLoaded = false;
00405         RaisePropertyChanged("IsRomLoaded");
00406         return;
00407     }

```

```

00408         else if (notificationMessage.Notification == "LoadFile")
00409         {
00410             var dialog = new OpenFileDialog
00411             {
00412                 DefaultExt = ".bin",
00413                 Filter =
00414                     "All Files (*.bin, *.65C02)|*.bin;*.65C02|Binary
Assembly (*.bin)|" +
00415                     "*.bin|WolfNet 65C02 Emulator Save State
(*.65C02)|*.65C02"
00416             };
00417             var result = dialog.ShowDialog();
00418             if (result != true)
00419             {
00420                 return;
00421             }
00422
00423             if (Path.GetExtension(dialog.FileName.ToUpper()) == ".BIN")
00424             {
00425                 byte[][] _rom = AT28C010.ReadFile(dialog.FileName);
00426
00427                 Messenger.Default.Send(new NotificationMessage<RomFileModel>(new RomFileModel
00428                 {
00429                     Rom = _rom,
00430                     RomBanks = AT28C010.Banks,
00431                     RomBankSize = AT28C010.Length,
00432                     RomFilePath = dialog.FileName,
00433                     RomFileName = Path.GetFileName(dialog.FileName),
00434                     }, "FileLoaded"));
00435             }
00436             else if (Path.GetExtension(dialog.FileName.ToUpper()) == ".6502")
00437             {
00438                 var formatter = new BinaryFormatter();
00439                 Stream stream = new FileStream(dialog.FileName, FileMode.Open);
00440                 var fileModel = (StateFileModel)formatter.Deserialize(stream);
00441
00442                 stream.Close();
00443
00444                 Messenger.Default.Send(new NotificationMessage<StateFileModel>(fileModel,
"StateLoaded"));
00445             }
00446         }
00447         else if (notificationMessage.Notification == "SaveState")
00448         {
00449             var dialog = new SaveFileDialog
00450             {
00451                 DefaultExt = ".65C02",
00452                 Filter =
00453                     "WolfNet W65C02 Emulator Save State
(*.65C02)|*.65C02"
00454             };
00455             var result = dialog.ShowDialog();
00456
00457             if (result != true)
00458             {
00459                 return;
00460             }
00461
00462             var formatter = new BinaryFormatter();
00463             Stream stream = new FileStream(dialog.FileName, FileMode.Create, FileAccess.Write,
FileShare.None);
00464
00465             formatter.Serialize(stream, new StateFileModel
00466             {
00467                 NumberOfCycles = NumberOfCycles,
00468                 OutputLog = OutputLog,
00469                 W65C02 = W65C02,
00470                 W65C22 = W65C22,
00471                 MM65SIB = MM65SIB,
00472                 W65C51 = W65C51,
00473                 AT28C010 = AT28C010,
00474                 AT28C64 = AT28C64,
00475             });
00476             stream.Close();
00477         }
00478         else
00479         {
00480             return;
00481         }
00482     }
00483
00484     private void SettingsAppliedNotifcation(NotificationMessage<SettingsModel>
notificationMessage)
00485     {
00486         if (notificationMessage.Notification != "SettingsApplied")
00487         {
00488             return;

```

```

00489         }
00490
00491         SettingsModel = notificationMessage.Content;
00492         W65C51.Init(notificationMessage.Content.ComPortName);
00493         RaisePropertyChanged("SettingsModel");
00494         UpdateUi();
00495     }
00496
00497     private void Reset()
00498     {
00499         IsRunning = false;
00500
00501         if (_backgroundWorker.IsBusy)
00502             _backgroundWorker.CancelAsync();
00503
00504         // "Reset" the Hardware...
00505         W65C02.Reset();
00506         RaisePropertyChanged("W65C02");
00507         W65C22.Reset();
00508         RaisePropertyChanged("W65C22");
00509         MM65SIB.Reset();
00510         RaisePropertyChanged("MM65SIB");
00511         W65C51.Reset();
00512         RaisePropertyChanged("W65C51");
00513         HM62256.Reset();
00514         RaisePropertyChanged("HM62256");
00515
00516         IsRunning = false;
00517         NumberOfCycles = 0;
00518         RaisePropertyChanged("NumberOfCycles");
00519
00520         UpdateMemoryPage();
00521         RaisePropertyChanged("MemoryPage");
00522
00523         OutputLog.Clear();
00524         RaisePropertyChanged("CurrentDisassembly");
00525
00526         OutputLog.Insert(0, GetOutputLog());
00527         UpdateUi();
00528     }
00529
00530     private void Step()
00531     {
00532         IsRunning = false;
00533
00534         if (_backgroundWorker.IsBusy)
00535             _backgroundWorker.CancelAsync();
00536
00537         StepProcessor();
00538         UpdateMemoryPage();
00539
00540         OutputLog.Insert(0, GetOutputLog());
00541         UpdateUi();
00542     }
00543
00544     private void UpdateUi()
00545     {
00546         RaisePropertyChanged("W65C02");
00547         RaisePropertyChanged("NumberOfCycles");
00548         RaisePropertyChanged("CurrentDisassembly");
00549         RaisePropertyChanged("MemoryPage");
00550     }
00551
00552     private void StepProcessor()
00553     {
00554         W65C02.NextStep();
00555         NumberOfCycles = W65C02.GetCycleCount();
00556     }
00557
00558     private OutputLog GetOutputLog()
00559     {
00560         if (W65C02.CurrentDisassembly == null)
00561         {
00562             return new OutputLog(new Disassembly());
00563         }
00564
00565         return new OutputLog(W65C02.CurrentDisassembly)
00566         {
00567             XRegister = W65C02.XRegister.ToString("X").PadLeft(2, '0'),
00568             YRegister = W65C02.YRegister.ToString("X").PadLeft(2, '0'),
00569             Accumulator = W65C02.Accumulator.ToString("X").PadLeft(2, '0'),
00570             NumberOfCycles = NumberOfCycles,
00571             StackPointer = W65C02.StackPointer.ToString("X").PadLeft(2, '0'),
00572             ProgramCounter = W65C02.ProgramCounter.ToString("X").PadLeft(4, '0'),
00573             CurrentOpCode = W65C02.CurrentOpCode.ToString("X").PadLeft(2, '0')
00574         };
00575     }

```

```

00576
00577     private void RunPause()
00578     {
00579         var isRunning = !IsRunning;
00580
00581         if (isRunning)
00582             _backgroundWorker.RunWorkerAsync();
00583         else
00584             _backgroundWorker.CancelAsync();
00585
00586         IsRunning = !IsRunning;
00587     }
00588
00589     private void BackgroundWorkerDoWork(object sender, DoWorkEventArgs e)
00590     {
00591         var worker = sender as BackgroundWorker;
00592         var outputLogs = new List<OutputLog>();
00593
00594         while (true)
00595         {
00596             if (worker != null && worker.CancellationPending || IsBreakPointTriggered())
00597             {
00598                 e.Cancel = true;
00599
00600                 RaisePropertyChanged("W65C02");
00601
00602                 foreach (var log in outputLogs)
00603                     OutputLog.Insert(0, log);
00604
00605                 UpdateMemoryPage();
00606                 return;
00607             }
00608
00609             StepProcessor();
00610             outputLogs.Add(GetOutputLog());
00611
00612             if (NumberOfCycles % GetLogModValue() == 0)
00613             {
00614                 foreach (var log in outputLogs)
00615                     OutputLog.Insert(0, log);
00616
00617                 outputLogs.Clear();
00618                 UpdateUi();
00619             }
00620             Thread.Sleep(GetSleepValue());
00621         }
00622     }
00623
00624     private bool IsBreakPointTriggered()
00625     {
00626         //This prevents the Run Command from getting stuck after reaching a breakpoint
00627         if (_breakpointTriggered)
00628         {
00629             _breakpointTriggered = false;
00630             return false;
00631         }
00632
00633         foreach (var breakpoint in Breakpoints.Where(x => x.IsEnabled))
00634         {
00635             if (!int.TryParse(breakpoint.Value, NumberStyles.AllowHexSpecifier,
CultureInfo.InvariantCulture, out int value))
00636                 continue;
00637
00638             if (breakpoint.Type == BreakpointType.NumberOfCycleType && value == NumberOfCycles)
00639             {
00640                 _breakpointTriggered = true;
00641                 RunPause();
00642                 return true;
00643             }
00644
00645             if (breakpoint.Type == BreakpointType.ProgramCounterType && value ==
W65C02.ProgramCounter)
00646             {
00647                 _breakpointTriggered = true;
00648                 RunPause();
00649                 return true;
00650             }
00651         }
00652
00653         return false;
00654     }
00655
00656     private int GetLogModValue()
00657     {
00658         switch (CpuSpeed)
00659         {
00660             case 0:

```

```
00661         case 1:
00662         case 2:
00663         case 3:
00664         case 4:
00665         case 5:
00666             return 1;
00667         case 6:
00668             return 5;
00669         case 7:
00670             return 20;
00671         case 8:
00672             return 30;
00673         case 9:
00674             return 40;
00675         case 10:
00676             return 50;
00677         default:
00678             return 5;
00679     }
00680 }
00681
00682 private int GetSleepValue()
00683 {
00684     switch (CpuSpeed)
00685     {
00686         case 0:
00687             return 550;
00688         case 1:
00689             return 550;
00690         case 2:
00691             return 440;
00692         case 3:
00693             return 330;
00694         case 4:
00695             return 220;
00696         case 5:
00697             return 160;
00698         case 6:
00699             return 80;
00700         case 7:
00701             return 40;
00702         case 8:
00703             return 20;
00704         case 9:
00705             return 10;
00706         case 10:
00707             return 5;
00708         default:
00709             return 5;
00710     }
00711 }
00712
00713 private void UpdateMemoryPage()
00714 {
00715     Messenger.Default.Send(new NotificationMessage("UpdateMemoryPage"));
00716 }
00717
00718 private void About()
00719 {
00720     IsRunning = false;
00721
00722     if (_backgroundWorker.IsBusy)
00723         _backgroundWorker.CancelAsync();
00724
00725     MessageBox.Show(string.Format("{0}\n{1}\nVersion: {2}\nCompany: {3}",
Versioning.Product.Name, Versioning.Product.Description, Versioning.Product.VersionString,
Versioning.Product.Company), Versioning.Product.Title);
00726 }
00727
00728 private void Settings()
00729 {
00730     IsRunning = false;
00731
00732     if (_backgroundWorker.IsBusy)
00733         _backgroundWorker.CancelAsync();
00734
00735     Messenger.Default.Send(new NotificationMessage<SettingsModel>(SettingsModel,
"SettingsWindow"));
00736 }
00737
00738 private void MemoryView()
00739 {
00740     Messenger.Default.Send(new NotificationMessage("MemoryVisualWindow"));
00741 }
00742
00743 private void AddBreakPoint()
00744 {
```



```

00745         Breakpoints.Add(new Breakpoint());
00746         RaisePropertyChanged("Breakpoints");
00747     }
00748
00749     private void RemoveBreakPoint()
00750     {
00751         if (SelectedBreakpoint == null)
00752             return;
00753
00754         Breakpoints.Remove(SelectedBreakpoint);
00755         SelectedBreakpoint = null;
00756         RaisePropertyChanged("SelectedBreakpoint");
00757     }
00758 #endregion
00759 }
00760 }

```

7.143 Emulator/ViewModel/MemoryVisualViewModel.cs File Reference

Classes

- class [Emulator.ViewModel.MemoryVisualViewModel](#)
The Main *ViewModel*

Namespaces

- namespace [Emulator](#)
- namespace [Emulator.ViewModel](#)

7.144 MemoryVisualViewModel.cs

[Go to the documentation of this file.](#)

```

00001 using Emulator.Model;
00002 using GalaSoft.MvvmLight;
00003 using GalaSoft.MvvmLight.Command;
00004 using GalaSoft.MvvmLight.Messaging;
00005 using Hardware;
00006 using System;
00007
00008 namespace Emulator.ViewModel
00009 {
00010     /// <summary>
00011     /// The Main ViewModel
00012     /// </summary>
00013     public class MemoryVisualViewModel : ViewModelBase
00014     {
00015         #region Fields
00016         private int _memoryPageOffset;
00017         #endregion
00018
00019         #region Properties
00020         /// <summary>
00021         /// The Current Memory Page
00022         /// </summary>
00023         public MultiThreadedObservableCollection<MemoryRowModel> MemoryPage { get; set; }
00024
00025         /// <summary>
00026         /// The Memory Page number.
00027         /// </summary>
00028         public string MemoryPageOffset
00029         {
00030             get { return _memoryPageOffset.ToString("X"); }
00031             set
00032             {
00033                 if (string.IsNullOrEmpty(value))
00034                     return;
00035                 try
00036                 {
00037                     _memoryPageOffset = Convert.ToInt32(value, 16);
00038                 }
00039                 catch { }
00040             }
00041         }
00042     }
00043 }

```

```

00040     }
00041 }
00042
00043 /// <summary>
00044 /// RelayCommand that updates the Memory Map when the Page changes
00045 /// </summary>
00046 public RelayCommand UpdateMemoryMapCommand { get; set; }
00047 #endregion
00048
00049 #region public Methods
00050 /// <summary>
00051 /// Creates a new Instance of the MemoryVisualViewModel.
00052 /// </summary>
00053 public MemoryVisualViewModel()
00054 {
00055     UpdateMemoryMapCommand = new RelayCommand(UpdateMemoryPage);
00056
00057     Messenger.Default.Register<NotificationMessage>(this, GenericNotification);
00058
00059     MemoryPage = new MultiThreadedObservableCollection<MemoryRowModel>();
00060
00061     UpdateMemoryPage();
00062     UpdateUi();
00063 }
00064
00065 private void GenericNotification(NotificationMessage notificationMessage)
00066 {
00067     if (notificationMessage.Notification == "UpdateMemoryPage")
00068     {
00069         UpdateMemoryPage();
00070         UpdateUi();
00071     }
00072 }
00073
00074 public void UpdateMemoryPage()
00075 {
00076     MemoryPage.Clear();
00077     var offset = _memoryPageOffset * 256;
00078
00079     var multiplier = 0;
00080     for (ushort i = (ushort)offset; i < 256 * (_memoryPageOffset + 1); i++)
00081     {
00082         MemoryPage.Add(new MemoryRowModel
00083         {
00084             Offset = ((16 * multiplier) + offset).ToString("X").PadLeft(4, '0'),
00085             Location00 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00086             Location01 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00087             Location02 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00088             Location03 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00089             Location04 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00090             Location05 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00091             Location06 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00092             Location07 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00093             Location08 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00094             Location09 = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00095             Location0A = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00096             Location0B = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00097             Location0C = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00098             Location0D = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00099             Location0E = MemoryMap.ReadWithoutCycle(i++).ToString("X").PadLeft(2, '0'),
00100             Location0F = MemoryMap.ReadWithoutCycle(i).ToString("X").PadLeft(2, '0'),
00101         });
00102         multiplier++;
00103     }
00104 }
00105 #endregion
00106
00107 #region Private Methods
00108 private void UpdateUi()
00109 {
00110     RaisePropertyChanged("MemoryPage");
00111 }
00112 #endregion
00113 }
00114 }

```

7.145 Emulator/ViewModel/SaveFileViewModel.cs File Reference

Classes

- class [Emulator.ViewModel.SaveFileViewModel](#)

The *ViewModel* Used by the *SaveFileView*

Namespaces

- namespace [Emulator](#)
- namespace [Emulator.ViewModel](#)

7.146 SaveFileViewModel.cs

[Go to the documentation of this file.](#)

```

00001 using Emulator.Model;
00002 using GalaSoft.MvvmLight;
00003 using GalaSoft.MvvmLight.Command;
00004 using GalaSoft.MvvmLight.Ioc;
00005 using GalaSoft.MvvmLight.Messaging;
00006 using Microsoft.Win32;
00007 using System.IO;
00008 using System.Runtime.Serialization.Formatters.Binary;
00009
00010 namespace Emulator.ViewModel
00011 {
00012     /// <summary>
00013     /// The ViewModel Used by the SaveFileView
00014     /// </summary>
00015     public class SaveFileViewModel : ViewModelBase
00016     {
00017         private readonly StateFileModel _stateFileModel;
00018
00019         #region Properties
00020         /// <summary>
00021         /// The Relay Command called when saving a file
00022         /// </summary>
00023         public RelayCommand SaveFileCommand { get; set; }
00024
00025         /// <summary>
00026         /// The Relay Command called when closing a file
00027         /// </summary>
00028         public RelayCommand CloseCommand { get; set; }
00029
00030         /// <summary>
00031         /// The Relay Command called when Selecting a file
00032         /// </summary>
00033         public RelayCommand SelectFileCommand { get; set; }
00034
00035         /// <summary>
00036         /// The file to be saved
00037         /// </summary>
00038         public string Filename { get; set; }
00039
00040         /// <summary>
00041         /// Tells the UI that that a file has been selected and can be saved.
00042         /// </summary>
00043         public bool SaveEnabled { get { return !string.IsNullOrEmpty(Filename); } }
00044         #endregion
00045
00046         #region Public Methods
00047         /// <summary>
00048         /// Instantiates a new instance of the SaveFileViewModel. This is used by the IOC to create the
00049         /// default instance.
00050         /// </summary>
00051         [PreferredConstructor]
00052         public SaveFileViewModel()
00053         {
00054         }
00055
00056         /// <summary>
00057         /// Instantiates a new instance of the SaveFileViewModel
00058         /// </summary>
00059         /// <param name="stateFileModel">The StateFileModel to be serialized to a file</param>
00060         public SaveFileViewModel(StateFileModel stateFileModel)
00061         {
00062             SaveFileCommand = new RelayCommand(Save);
00063             CloseCommand = new RelayCommand(Close);
00064             SelectFileCommand = new RelayCommand(Select);
00065             _stateFileModel = stateFileModel;
00066         }
00067         #endregion
00068
00069         #region Private Methods
00070         private void Save()
00071         {
00072             var formatter = new BinaryFormatter();

```

```

00073         Stream stream = new FileStream(Filename, FileMode.Create, FileAccess.Write,
    FileShare.None);
00074         formatter.Serialize(stream, _stateFileModel);
00075         stream.Close();
00076
00077         Close();
00078     }
00079
00080     private static void Close()
00081     {
00082         Messenger.Default.Send(new NotificationMessage("CloseSaveFileWindow"));
00083     }
00084
00085     private void Select()
00086     {
00087         var dialog = new SaveFileDialog { DefaultExt = ".6502", Filter = "WolfNet W65C02 Emulator
Save State (*.6502)|*.6502" };
00088
00089         var result = dialog.ShowDialog();
00090
00091         if (result != true)
00092             return;
00093
00094         Filename = dialog.FileName;
00095         RaisePropertyChanged("Filename");
00096         RaisePropertyChanged("SaveEnabled");
00097     }
00098 }
00099 #endregion
00100 }
00101 }

```

7.147 Emulator/ViewModel/SettingsViewModel.cs File Reference

Classes

- class [Emulator.ViewModel.SettingsViewModel](#)
The *ViewModel* Used by the *SaveFileView*

Namespaces

- namespace [Emulator](#)
- namespace [Emulator.ViewModel](#)

7.148 SettingsViewModel.cs

[Go to the documentation of this file.](#)

```

00001 using Emulator.Model;
00002 using GalaSoft.MvvmLight;
00003 using GalaSoft.MvvmLight.Command;
00004 using GalaSoft.MvvmLight.Ioc;
00005 using GalaSoft.MvvmLight.Messaging;
00006 using System.Collections.ObjectModel;
00007 using System.IO.Ports;
00008
00009 namespace Emulator.ViewModel
00010 {
00011     /// <summary>
00012     /// The ViewModel Used by the SaveFileView
00013     /// </summary>
00014     public class SettingsViewModel : ViewModelBase
00015     {
00016         #region Properties
00017         /// <summary>
00018         /// The Relay Command called when saving a file
00019         /// </summary>
00020         public RelayCommand ApplyCommand { get; set; }
00021
00022         /// <summary>
00023         /// The Relay Command called when closing a file
00024         /// </summary>

```

```

00025         public RelayCommand CloseCommand { get; set; }
00026
00027     /// <summary>
00028     /// Tells the UI that that a file has been selected and can be saved.
00029     /// </summary>
00030     public bool ApplyEnabled { get { return
!string.IsNullOrEmpty(Emulator.FileLocations.SettingsFile); } }
00031
00032     /// <summary>
00033     /// Creates a new instance of PortList, the list of all COM ports available to the computer
00034     /// </summary>
00035     ///
00036     public ObservableCollection<string> PortList { get { return _PortList; } }
00037     private readonly ObservableCollection<string> _PortList = new ObservableCollection<string>();
00038
00039     public static string ComPortSelection { get; set; }
00040     public static SettingsModel SettingsModel { get; set; }
00041 #endregion
00042
00043 #region Public Methods
00044     /// <summary>
00045     /// Instantiates a new instance of the SettingsViewModel. This is used by the IOC to create the
    default instance.
00046     /// </summary>
00047     [PreferredConstructor]
00048     public SettingsViewModel()
00049     {
00050
00051     }
00052
00053     /// <summary>
00054     /// Instantiates a new instance of the SettingsViewModel
00055     /// </summary>
00056     /// <param name="settingsModel">The SettingsFileModel to be serialized to a file</param>
00057     public SettingsViewModel(SettingsModel settingsModel)
00058     {
00059         ApplyCommand = new RelayCommand(Apply);
00060         CloseCommand = new RelayCommand(Close);
00061         ComPortSelection = settingsModel.ComPortName;
00062
00063         UpdatePortList();
00064     }
00065
00066     /// <summary>
00067     /// Updates PortList with the COM ports available to the computer
00068     /// </summary>
00069     public void UpdatePortList()
00070     {
00071         PortList.Clear();
00072         foreach (string s in SerialPort.GetPortNames())
00073         {
00074             PortList.Add(s);
00075         }
00076         RaisePropertyChanged("PortList");
00077     }
00078 #endregion
00079
00080 #region Private Methods
00081     private void Apply()
00082     {
00083         Messenger.Default.Send(new NotificationMessage<SettingsModel>(new SettingsModel
00084         {
00085             SettingsVersionMajor = Versioning.SettingsFile.Major,
00086             SettingsVersionMinor = Versioning.SettingsFile.Minor,
00087             SettingsVersionBuild = Versioning.SettingsFile.Build,
00088             SettingsVersionRevision = Versioning.SettingsFile.Revision,
00089             ComPortName = ComPortSelection,
00090         }, "SettingsApplied"));
00091         Messenger.Default.Send(new NotificationMessage("CloseSettingsWindow"));
00092     }
00093
00094     private static void Close()
00095     {
00096         Messenger.Default.Send(new NotificationMessage("CloseSettingsWindow"));
00097     }
00098 #endregion
00099 }
00100 }

```

7.149 Emulator/ViewModel/ViewModelLocator.cs File Reference

Classes

- class [Emulator.ViewModel.ViewModelLocator](#)

This class contains static references to all the view models in the application and provides an entry point for the bindings.

Namespaces

- namespace [Emulator](#)
- namespace [Emulator.ViewModel](#)

7.150 ViewModelLocator.cs

[Go to the documentation of this file.](#)

```

00001  /*
00002  In App.xaml:
00003  <Application.Resources>
00004  <vm:ViewModelLocator xmlns:vm="clr-namespace:Emulator"
00005  x:Key="Locator" />
00006  </Application.Resources>
00007
00008  In the View:
00009  DataContext="{Binding Source={StaticResource Locator}, Path=ViewModelName}"
00010
00011  You can also use Blend to do all this with the tool's support.
00012  See http://www.galasoft.ch/mvvm
00013  */
00014
00015  using GalaSoft.MvvmLight.Ioc;
00016  using Microsoft.Practices.ServiceLocation;
00017
00018  namespace Emulator.ViewModel
00019  {
00020  /// <summary>
00021  /// This class contains static references to all the view models in the
00022  /// application and provides an entry point for the bindings.
00023  /// </summary>
00024  public class ViewModelLocator
00025  {
00026  /// <summary>
00027  /// Initializes a new instance of the ViewModelLocator class.
00028  /// </summary>
00029  public ViewModelLocator()
00030  {
00031      ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);
00032
00033      SimpleIoc.Default.Register<MainViewModel>();
00034      SimpleIoc.Default.Register<SettingsViewModel>();
00035      SimpleIoc.Default.Register<MemoryVisualViewModel>();
00036  }
00037
00038  /// <summary>
00039  /// The MainViewModel Instance
00040  /// </summary>
00041  public MainViewModel Main
00042  {
00043      get { return ServiceLocator.Current.GetInstance<MainViewModel>(); }
00044  }
00045
00046  /// <summary>
00047  /// The SettingsViewModel Instance
00048  /// </summary>
00049  public SettingsViewModel Settings
00050  {
00051      get { return ServiceLocator.Current.GetInstance<SettingsViewModel>(); }
00052  }
00053
00054  /// <summary>
00055  /// The SaveFileViewModel Instance
00056  /// </summary>
00057  public MemoryVisualViewModel MemoryVisual
00058  {
00059      get { return ServiceLocator.Current.GetInstance<MemoryVisualViewModel>(); }
00060  }
00061
00062  /// <summary>
00063  /// The Cleanup Method
00064  /// </summary>
00065  public static void Cleanup()
00066  {

```

```

00067 /// <todo>
00068 /// Clear the ViewModels
00069 /// </todo>
00070     }
00071 }
00072 }

```

7.151 Hardware/Classes/AddressingMode.cs File Reference

Namespaces

- namespace [Hardware](#)

Enumerations

- enum [Hardware.AddressingMode](#)

The addressing modes used by the 6502 Processor

7.152 AddressingMode.cs

[Go to the documentation of this file.](#)

```

00001 namespace Hardware
00002 {
00003     /// <summary>
00004     /// The addressing modes used by the 6502 Processor
00005     /// </summary>
00006     public enum AddressingMode
00007     {
00008         /// <summary>
00009         /// In this mode a full address is given to operation on IE: Memory byte[] { 0x60, 0x00, 0xFF } The
00010         /// would perform an ADC operation and Add the value at ADDRESS 0xFF00 to the accumulator.
00011         /// The address is always LSB first
00012         /// </summary>
00013         Absolute = 1,
00014         /// <summary>
00015         /// In this mode a full address is given to operation on IE: Memory byte[] { 0x7D, 0x00, 0xFF } The
00016         /// full value would then be added to the X Register.
00017         /// If the X register was 0x01 then the address would be 0xFF01. and the value stored there would
00018         /// have an ADC operation performed on it and the value would
00019         /// be added to the accumulator.
00020         /// </summary>
00021         AbsoluteX = 2,
00022         /// <summary>
00023         /// In this mode a full address is given to operation on IE: Memory byte[] { 0x79, 0x00, 0xFF } The
00024         /// full value would then be added to the Y Register.
00025         /// If the Y register was 0x01 then the address would be 0xFF01. and the value stored there would
00026         /// have an ADC operation performed on it and the value would
00027         /// be added to the accumulator
00028         /// </summary>
00029         AbsoluteY = 3,
00030         /// <summary>
00031         /// In this mode the instruction operates on the accumulator. No operands are needed.
00032         /// </summary>
00033         Accumulator = 4,
00034         /// <summary>
00035         /// In this mode, the value to operate on immediately follows the instruction. IE: Memory byte[] {
00036         /// 0x69, 0x01 }
00037         /// would perform an ADC operation and Add 0x01 directly to the accumulator
00038         /// </summary>
00039         Immediate = 5,
00040         /// <summary>
00041         /// No address is needed for this mode. EX: BRK (Break), CLC (Clear Carry Flag) etc
00042         /// </summary>
00043         Implied = 6,
00044         /// <summary>
00045         /// In this mode assume the following
00046         /// Memory = { 0x61, 0x02, 0x04, 0x00, 0x03 }
00047         /// RegisterX = 0x01
00048         /// 1. Take the sum of the X Register and the value after the opcode 0x01 + 0x01 = 0x02.
00049         /// 2. Starting at position 0x02 get an address (0x04,0x00) = 0x0004
00050         /// 3. Perform the ADC operation and Add the value at 0x0005 to the accumulator

```

```

00046 /// Note: if the Zero Page address is greater than 0xff then roll over the value. IE 0x101 rolls
        over to 0x01
00047 /// </summary>
00048 IndirectX = 7,
00049 /// <summary>
00050 /// In this mode assume the following
00051 /// Memory = { 0x61, 0x02, 0x04, 0x00, 0x03 }
00052 /// RegisterY = 0x01
00053 /// 1. Starting at position 0x02 get an address (0x04,0x00) = 0x0004
00054 /// 2. Take the sum of the Y Register and the absolute address 0x01+0x0004 = 0x0005
00055 /// 3. Perform the ADC operation and Add the value at 0x0005 to the accumulator
00056 /// Note: if the address is great that 0xffff then roll over IE: 0x10001 rolls over to 0x01
00057 /// </summary>
00058 IndirectY = 8,
00059 /// <summary>
00060 /// JMP is the only operation that uses this mode. In this mode an absolute address is specified that
        points to the location of the absolute address we want to jump to.
00061 /// </summary>
00062 Indirect = 9,
00063 /// <summary>
00064 /// This Mode Changes the PC. It allows the program to change the location of the PC by 127 in either
        direction.
00065 /// </summary>
00066 Relative = 10,
00067 /// <summary>
00068 /// In this mode, a zero page address of the value to operate on is specified. This mode can only
        operation on values between 0x0 and 0xFF, or those that sit on the zero page of memory. IE: Memory
        byte[] { 0x69, 0x02, 0x01 }
00069 /// would perform an ADC operation and Add 0x01 directly to the Accumulator
00070 /// </summary>
00071 ZeroPage = 11,
00072 /// <summary>
00073 /// In this mode, a zero page address of the value to operate on is specified, however the value of
        the X register is added to the address IE: Memory byte[] { 0x86, 0x02, 0x01, 0x67, 0x04, 0x01 }
00074 /// In this example we store a value of 0x01 into the X register, then we would perform an ADC
        operation using the address of 0x04+0x01=0x05 and Add the result of 0x01 directly to the Accumulator
00075 /// </summary>
00076 ZeroPageX = 12,
00077 /// <summary>
00078 /// This works the same as ZeroPageX except it uses the Y register instead of the X register.
00079 /// </summary>
00080 ZeroPageY = 13,
00081 }
00082 }

```

7.153 Hardware/Classes/Disassembly.cs File Reference

Classes

- class [Hardware.Disassembly](#)
Used to help simulating. This class contains the disassembly properties.

Namespaces

- namespace [Hardware](#)

7.154 Disassembly.cs

[Go to the documentation of this file.](#)

```

00001 using System;
00002
00003 namespace Hardware
00004 {
00005     /// <summary>
00006     /// Used to help simulating. This class contains the disassembly properties.
00007     /// </summary>
00008     [Serializable]
00009     public class Disassembly
00010     {
00011     /// <summary>
00012     /// The low Address
00013     /// </summary>

```



```

00014         public string LowAddress { get; set; }
00015
00016     /// <summary>
00017     /// The High Address
00018     /// </summary>
00019     public string HighAddress { get; set; }
00020
00021     /// <summary>
00022     /// The string representation of the OpCode
00023     /// </summary>
00024     public string OpCodeString { get; set; }
00025
00026     /// <summary>
00027     /// The disassembly of the current step
00028     /// </summary>
00029     public string DisassemblyOutput { get; set; }
00030     }
00031 }

```

7.155 Hardware/Classes/MemoryMap.cs File Reference

Classes

- class [Hardware.MemoryMap](#)
- class [Hardware.MemoryMap.BankedRam](#)
- class [Hardware.MemoryMap.DeviceArea](#)
- class [Hardware.MemoryMap.BankedRom](#)
- class [Hardware.MemoryMap.SharedRom](#)
- class [Hardware.MemoryMap.Devices](#)
- class [Hardware.MemoryMap.Devices.ACIA](#)
- class [Hardware.MemoryMap.Devices.GPIO](#)
- class [Hardware.MemoryMap.Devices.MM65SIB](#)

Namespaces

- namespace [Hardware](#)

7.156 MemoryMap.cs

[Go to the documentation of this file.](#)

```

00001 using System;
00002
00003 namespace Hardware
00004 {
00005     public class MemoryMap
00006     {
00007         public static class BankedRam
00008         {
00009             private static int _Offset = 0x0000;
00010             private static int _Length = 0x7FFF;
00011
00012             public static int TotalLength = (BankSize * TotalBanks) - 1;
00013             public static int BankSize = (int)(Length + 1);
00014             public static byte TotalBanks = 16;
00015
00016             public static int Offset { get { return _Offset; } }
00017             public static int Length { get { return _Length; } }
00018         }
00019
00020         public static class DeviceArea
00021         {
00022             private static int _Offset = 0xD000;
00023             private static int _Length = 0x00FF;
00024
00025             /// <summary>
00026             /// The end of memory
00027             /// </summary>
00028             public static int End { get { return Offset + Length; } }

```

```

00029         public static int Offset { get { return _Offset; } }
00030         public static int Length { get { return _Length; } }
00031     }
00032
00033     public static class BankedRom
00034     {
00035         private static int _Offset = 0x8000;
00036         private static int _Length = 0x3FFF;
00037
00038         public static byte TotalBanks = 16;
00039
00040         public static int Offset { get { return _Offset; } }
00041         public static int Length { get { return _Length; } }
00042     }
00043
00044     public static class SharedRom
00045     {
00046         private static int _Offset = 0xE000;
00047         private static int _Length = 0x1FFF;
00048
00049         public static byte TotalBanks = 1;
00050
00051         public static int Offset { get { return _Offset; } }
00052         public static int Length { get { return _Length; } }
00053     }
00054
00055     public static class Devices
00056     {
00057         public static class ACIA
00058         {
00059             public static int Length = 0x03;
00060             public static byte Offset = 0x10;
00061         }
00062
00063         public static class GPIO
00064         {
00065             public static int Length = 0x0F;
00066             public static byte Offset = 0x20;
00067         }
00068
00069         public static class MM65SIB
00070         {
00071             public static int Length = 0x0F;
00072             public static byte Offset = 0x30;
00073         }
00074     }
00075
00076     public static readonly int Length = 0xFFFF;
00077
00078     private static W65C02 Processor { get; set; }
00079     private static W65C22 GPIO { get; set; }
00080     private static W65C22 MM65SIB { get; set; }
00081     private static W65C51 ACIA { get; set; }
00082     private static AT28CXx SharedROM { get; set; }
00083     private static AT28CXx BankedROM { get; set; }
00084     private static HM62256 BankedRAM { get; set; }
00085
00086     public static void Init(W65C02 processor, W65C22 gpio, W65C22 mm65sib, W65C51 acia, HM62256
bankedRam, AT28CXx bankedRom, AT28CXx sharedRom)
00087     {
00088         Processor = processor;
00089         GPIO = gpio;
00090         MM65SIB = mm65sib;
00091         ACIA = acia;
00092         SharedROM = sharedRom;
00093         BankedROM = bankedRom;
00094         BankedRAM = bankedRam;
00095     }
00096
00097     /// <summary>
00098     /// Returns the byte at the given address.
00099     /// </summary>
00100     /// <param name="address">The address to return</param>
00101     /// <returns>the byte being returned</returns>
00102     public static byte Read(int address)
00103     {
00104         var value = ReadWithoutCycle(address);
00105         Processor.IncrementCycleCount();
00106         return value;
00107     }
00108
00109     /// <summary>
00110     /// Returns the byte at the given address without incrementing the cycle count.
00111     /// </summary>
00112     /// <param name="address">The address to return</param>
00113     /// <returns>the byte being returned</returns>
00114     public static byte ReadWithoutCycle(int address)

```

```

00115     {
00116         if ((ACIA.Offset <= address) && (address <= (ACIA.Offset + ACIA.Length)))
00117         {
00118             return ACIA.Read(address);
00119         }
00120         else if ((GPIO.Offset <= address) && (address <= (GPIO.Offset + GPIO.Length)))
00121         {
00122             return GPIO.Read(address);
00123         }
00124         else if ((MM65SIB.Offset <= address) && (address <= (MM65SIB.Offset + MM65SIB.Length)))
00125         {
00126             return MM65SIB.Read(address);
00127         }
00128         else if ((DeviceArea.Offset <= address) && (address <= DeviceArea.End))
00129         {
00130             return 0x00;
00131         }
00132         else if ((SharedROM.Offset <= address) && (address <= SharedROM.End))
00133         {
00134             return SharedROM.Read(address);
00135         }
00136         else if ((BankedROM.Offset <= address) && (address <= BankedROM.End))
00137         {
00138             return BankedROM.Read(address);
00139         }
00140         else if ((BankedRAM.Offset <= address) && (address <= BankedRAM.End))
00141         {
00142             return BankedRAM.Read(address);
00143         }
00144         else
00145         {
00146             return 0x00;
00147         }
00148     }
00149
00150     /// <summary>
00151     /// Writes data to the given address.
00152     /// </summary>
00153     /// <param name="address">The address to write data to.</param>
00154     /// <param name="data">The data to write.</param>
00155     public static void Write(int address, byte data)
00156     {
00157         Processor.IncrementCycleCount();
00158         WriteWithoutCycle(address, data);
00159     }
00160
00161     /// <summary>
00162     /// Writes data to the given address without incrementing the cycle count.
00163     /// </summary>
00164     /// <param name="address">The address to write data to.</param>
00165     /// <param name="data">The data to write.</param>
00166     public static void WriteWithoutCycle(int address, byte data)
00167     {
00168         if ((ACIA.Offset <= address) && (address <= (ACIA.Offset + ACIA.Length)))
00169         {
00170             ACIA.Write(address, data);
00171         }
00172         else if ((GPIO.Offset <= address) && (address <= (GPIO.Offset + GPIO.Length)))
00173         {
00174             GPIO.Write(address, data);
00175         }
00176         else if ((MM65SIB.Offset <= address) && (address <= (MM65SIB.Offset + MM65SIB.Length)))
00177         {
00178             MM65SIB.Write(address, data);
00179         }
00180         else if ((SharedROM.Offset <= address) && (address <= (SharedROM.Offset +
SharedROM.Length)))
00181         {
00182             SharedROM.Write(address, data);
00183         }
00184         else if ((BankedROM.Offset <= address) && (address <= (BankedROM.Offset +
BankedROM.Length)))
00185         {
00186             BankedROM.Write(address, data);
00187         }
00188         else if ((BankedRAM.Offset <= address) && (address <= (BankedRAM.Offset +
BankedRAM.Length)))
00189         {
00190             BankedRAM.Write(address, data);
00191         }
00192         else
00193         {
00194             throw new ApplicationException(String.Format("Cannot write to address: {0}",
address));
00195         }
00196     }
00197 }

```

```
00198 }
```

7.157 Hardware/Classes/Utility.cs File Reference

Classes

- class [Hardware.Utility](#)

Namespaces

- namespace [Hardware](#)

7.158 Utility.cs

[Go to the documentation of this file.](#)

```
00001 using System.ComponentModel;
00002
00003 namespace Hardware
00004 {
00005     public static class Utility
00006     {
00007         public static string ConvertOpCodeIntoString(this int i)
00008         {
00009             switch (i)
00010             {
00011                 case 0x69: //ãADCãImmediate
00012                 case 0x65: //ãADCãZeroãPage
00013                 case 0x75: //ãADCãZeroãPageãX
00014                 case 0x6D: //ãADCãAbsolute
00015                 case 0x7D: //ãADCãAbsoluteãX
00016                 case 0x79: //ãADCãAbsoluteãY
00017                 case 0x61: //ãADCãIndirectãX
00018                 case 0x71: //ãADCãIndirectãY
00019                 {
00020                     return "ADC";
00021                 }
00022                 case 0x29: //ãANDãImmediate
00023                 case 0x25: //ãANDãZeroãPage
00024                 case 0x35: //ãANDãZeroãPageãX
00025                 case 0x2D: //ãANDãAbsolute
00026                 case 0x3D: //ãANDãAbsoluteãX
00027                 case 0x39: //ãANDãAbsoluteãY
00028                 case 0x21: //ãANDãIndirectãX
00029                 case 0x31: //ãANDãIndirectãY
00030                 {
00031                     return "AND";
00032                 }
00033                 case 0x0A: //ãASLãAccumulator
00034                 case 0x06: //ãASLãZeroãPage
00035                 case 0x16: //ãASLãZeroãPageãX
00036                 case 0x0E: //ãASLãAbsolute
00037                 case 0x1E: //ãASLãAbsoluteãX
00038                 {
00039                     return "ASL";
00040                 }
00041                 case 0x90: //ãBCCãRelative
00042                 {
00043                     return "BCC";
00044                 }
00045                 case 0xB0: //ãBCSãRelative
00046                 {
00047                     return "BCS";
00048                 }
00049                 case 0xF0: //ãBEQãRelative
00050                 {
00051                     return "BEQ";
00052                 }
00053                 case 0x24: //ãBITãZeroãPage
00054                 case 0x2C: //ãBITãAbsolute
00055                 {
00056                     return "BIT";
00057                 }
00058                 case 0x30: //ãBMIãRelative
```

```

00059         {
00060             return "BMI";
00061         }
00062     case 0xD0: // ãBNEãRelative
00063     {
00064         return "BNE";
00065     }
00066     case 0x10: // ãBPLãRelative
00067     {
00068         return "BPL";
00069     }
00070     case 0x00: // ãBRKãImplied
00071     {
00072         return "BRK";
00073     }
00074     case 0x50: // BVC Relative
00075     {
00076         return "BCV";
00077     }
00078     case 0x70: //BVS Relative
00079     {
00080         return "BVS";
00081     }
00082     case 0x18: // ãCLCãImplied
00083     {
00084         return "CLC";
00085     }
00086     case 0xD8: // ãCLDãImplied
00087     {
00088         return "CLD";
00089     }
00090     case 0x58: // ãCLIãImplied
00091     {
00092         return "CLI";
00093     }
00094     case 0xB8: // ãCLVãImplied
00095     {
00096         return "CLV";
00097     }
00098     case 0xC9: // ãCMPãImmediate
00099     case 0xC5: // ãCMPãZeroãPage
00100     case 0xD5: // ãCMPãZeroãPageãX
00101     case 0xCD: // ãCMPãAbsolute
00102     case 0xDD: // ãCMPãAbsoluteãX
00103     case 0xD9: // ãCMPãAbsoluteãY
00104     case 0xC1: // ãCMPãIndirectãX
00105     case 0xD1: // ãCMPãIndirectãY
00106     {
00107         return "CMP";
00108     }
00109     case 0xE0: // ãCPXãImmediate
00110     case 0xE4: // ãCPXãZeroãPage
00111     case 0xEC: // ãCPXãAbsolute
00112     {
00113         return "CPX";
00114     }
00115     case 0xC0: // ãCPYãImmediate
00116     case 0xC4: // ãCPYãZeroãPage
00117     case 0xCC: // ãCPYãAbsolute
00118     {
00119         return "CPY";
00120     }
00121     case 0xC6: // ãDECãZeroãPage
00122     case 0xD6: // ãDECãZeroãPageãX
00123     case 0xCE: // ãDECãAbsolute
00124     case 0xDE: // ãDECãAbsoluteãX
00125     {
00126         return "DEC";
00127     }
00128     case 0xCA: // ãDEXãImplied
00129     {
00130         return "DEX";
00131     }
00132     case 0x88: // ãDEYãImplied
00133     {
00134         return "DEY";
00135     }
00136     case 0x49: // ãEORãImmediate
00137     case 0x45: // ãEORãZeroãPage
00138     case 0x55: // ãEORãZeroãPageãX
00139     case 0x4D: // ãEORãAbsolute
00140     case 0x5D: // ãEORãAbsoluteãX
00141     case 0x59: // ãEORãAbsoluteãY
00142     case 0x41: // ãEORãIndirectãX
00143     case 0x51: // ãEORãIndirectãY
00144     {
00145         return "EOR";

```

```

00146         }
00147         case 0xE6: //äINCäZeroäPage
00148         case 0xF6: //äINCäZeroäPageäX
00149         {
00150             return "INC";
00151         }
00152         case 0xE8: //äINXäImplied
00153         {
00154             return "INX";
00155         }
00156         case 0xC8: //äINYäImplied
00157         {
00158             return "INY";
00159         }
00160         case 0xEE: //äINCäAbsolute
00161         case 0xFE: //äINCäAbsoluteäX
00162         {
00163             return "INC";
00164         }
00165         case 0x4C: //äJMPäAbsolute
00166         case 0x6C: //äJMPäIndirect
00167         {
00168             return "JMP";
00169         }
00170         case 0x20: //äJSRäAbsolute
00171         {
00172             return "JSR";
00173         }
00174         case 0xA9: //äLDAäImmediate
00175         case 0xA5: //äLDAäZeroäPage
00176         case 0xB5: //äLDAäZeroäPageäX
00177         case 0xAD: //äLDAäAbsolute
00178         case 0xBD: //äLDAäAbsoluteäX
00179         case 0xB9: //äLDAäAbsoluteäY
00180         case 0xA1: //äLDAäIndirectäX
00181         case 0xB1: //äLDAäIndirectäY
00182         {
00183             return "LDA";
00184         }
00185         case 0xA2: //äLDXäImmediate
00186         case 0xA6: //äLDXäZeroäPage
00187         case 0xB6: //äLDXäZeroäPageäY
00188         case 0xAE: //äLDXäAbsolute
00189         case 0xBE: //äLDXäAbsoluteäY
00190         {
00191             return "LDX";
00192         }
00193         case 0xA0: //äLDYäImmediate
00194         case 0xA4: //äLDYäZeroäPage
00195         case 0xB4: //äLDYäZeroäPageäY
00196         case 0xAC: //äLDYäAbsolute
00197         case 0xBC: //äLDYäAbsoluteäY
00198         {
00199             return "LDY";
00200         }
00201         case 0x4A: //äLSRäAccumulator
00202         case 0x46: //äLSRäZeroäPage
00203         case 0x56: //äLSRäZeroäPageäX
00204         case 0x4E: //äLSRäAbsolute
00205         case 0x5E: //äLSRäAbsoluteäX
00206         {
00207             return "LSR";
00208         }
00209         case 0xEA: //äNOPäImplied
00210         {
00211             return "NOP";
00212         }
00213         case 0x09: //äORAäImmediate
00214         case 0x05: //äORAäZeroäPage
00215         case 0x15: //äORAäZeroäPageäX
00216         case 0x0D: //äORAäAbsolute
00217         case 0x1D: //äORAäAbsoluteäX
00218         case 0x19: //äORAäAbsoluteäY
00219         case 0x01: //äORAäIndirectäX
00220         case 0x11: //äORAäIndirectäY
00221         {
00222             return "ORA";
00223         }
00224         case 0x48: //äPHAäImplied
00225         {
00226             return "PHA";
00227         }
00228         case 0x08: //äPHPäImplied
00229         {
00230             return "PHP";
00231         }
00232         case 0x68: //äPLAäImplied

```

```

00233         {
00234             return "PLA";
00235         }
00236     case 0x28: //äPLPäImplied
00237     {
00238         return "PLP";
00239     }
00240     case 0x2A: //äROLäAccumulator
00241     case 0x26: //äROLäZeroäPage
00242     case 0x36: //äROLäZeroäPageäX
00243     case 0x2E: //äROLäAbsolute
00244     case 0x3E: //äROLäAbsoluteäX
00245     {
00246         return "ROL";
00247     }
00248     case 0x6A: //äRORäAccumulator
00249     case 0x66: //äRORäZeroäPage
00250     case 0x76: //äRORäZeroäPageäX
00251     case 0x6E: //äRORäAbsolute
00252     case 0x7E: //äRORäAbsoluteäX
00253     {
00254         return "ROR";
00255     }
00256     case 0x40: //äRTIäImplied
00257     {
00258         return "RTI";
00259     }
00260     case 0x60: //äRTSäImplied
00261     {
00262         return "RTS";
00263     }
00264     case 0xE9: //äSBCäImmediate
00265     case 0xE5: //äSBCäZeroäPage
00266     case 0xF5: //äSBCäZeroäPageäX
00267     case 0xED: //äSBCäAbsolute
00268     case 0xFD: //äSBCäAbsoluteäX
00269     case 0xF9: //äSBCäAbsoluteäY
00270     case 0xE1: //äSBCäIndirectäX
00271     case 0xF1: //äSBCäIndirectäY
00272     {
00273         return "SBC";
00274     }
00275     case 0x38: //äSECäImplied
00276     {
00277         return "SEC";
00278     }
00279     case 0xF8: //äSEDäImplied
00280     {
00281         return "SED";
00282     }
00283     case 0x78: //äSEIäImplied
00284     {
00285         return "SEI";
00286     }
00287     case 0x85: //äSTAäZeroPage
00288     case 0x95: //äSTAäZeroäPageäX
00289     case 0x8D: //äSTAäAbsolute
00290     case 0x9D: //äSTAäAbsoluteäX
00291     case 0x99: //äSTAäAbsoluteäY
00292     case 0x81: //äSTAäIndirectäX
00293     case 0x91: //äSTAäIndirectäY
00294     {
00295         return "STA";
00296     }
00297     case 0x86: //äSTXäZeroäPage
00298     case 0x96: //äSTXäZeroäPageäY
00299     case 0x8E: //äSTXäAbsolute
00300     {
00301         return "STX";
00302     }
00303     case 0x84: //äSTYäZeroäPage
00304     case 0x94: //äSTYäZeroäPageäX
00305     case 0x8C: //äSTYäAbsolute
00306     {
00307         return "STY";
00308     }
00309     case 0xAA: //äTAXäImplied
00310     {
00311         return "TAX";
00312     }
00313     case 0xA8: //äTAYäImplied
00314     {
00315         return "TAY";
00316     }
00317     case 0xBA: //äTSXäImplied
00318     {
00319         return "TSX";

```

```

00320         }
00321         case 0x8A: //äTXAäImplied
00322         {
00323             return "TXA";
00324         }
00325         case 0x9A: //äTXSäImplied
00326         {
00327             return "TXS";
00328         }
00329         case 0x98: //äTYAäImplied
00330         {
00331             return "TYA";
00332         }
00333         default:
00334             throw new InvalidEnumArgumentException(string.Format("A Valid Conversion does not
exist for OpCode {0}", i.ToString("X")));
00335     }
00336 }
00337 }
00338 }
00339 }

```

7.159 Hardware/Hardware/AT28CXX.cs File Reference

Classes

- class [Hardware.AT28CXX](#)
An implementation of a [W65C02](#) Processor.

Namespaces

- namespace [Hardware](#)

7.160 AT28CXX.cs

[Go to the documentation of this file.](#)

```

00001 using System;
00002 using System.IO;
00003
00004 namespace Hardware
00005 {
00006     /// <summary>
00007     /// An implementation of a W65C02 Processor.
00008     /// </summary>
00009     [Serializable]
00010     public class AT28CXX
00011     {
00012         ///All of the properties here are public and read only to facilitate ease of debugging and
testing.
00013         #region Properties
00014         /// <summary>
00015         /// The ROM.
00016         /// </summary>
00017         public byte[][] Memory { get; private set; }
00018
00019         /// <summary>
00020         /// The total number of banks on the ROM.
00021         /// </summary>
00022         public byte Banks { get; private set; }
00023
00024         /// <summary>
00025         /// The bank the ROM is currently using.
00026         /// </summary>
00027         public byte CurrentBank { get; private set; }
00028
00029         /// <summary>
00030         /// The memory offset
00031         /// </summary>
00032         public int Offset { get; private set; }
00033
00034         /// <summary>
00035         /// The end of memory

```



```

00036 /// </summary>
00037 public int End { get { return Offset + Length; } }
00038
00039 /// <summary>
00040 /// The memory length
00041 /// </summary>
00042 public int Length { get; private set; }
00043
00044 /// <summary>
00045 /// The processor reference
00046 /// </summary>
00047 public W65C02 Processor { get; private set; }
00048 #endregion
00049
00050 #region Public Methods
00051 /// <summary>
00052 /// Default Constructor, Instantiates a new instance of the processor.
00053 /// </summary>
00054 public AT28CXX(int offset, int length, byte banks)
00055 {
00056     Memory = new byte[banks][];
00057     for (int i = 0; i < banks; i++)
00058     {
00059         Memory[i] = new byte[length + 1];
00060     }
00061     Offset = offset;
00062     Length = length;
00063     Banks = banks;
00064     CurrentBank = 0;
00065 }
00066
00067 /// <summary>
00068 /// Loads a program into ROM.
00069 /// </summary>
00070 /// <param name="data">The program to be loaded</param>
00071 public void Load(byte[][] data)
00072 {
00073     for (byte i = 0; i < Banks; i++)
00074     {
00075         Load(i, data[i]);
00076     }
00077 }
00078
00079 /// <summary>
00080 /// Loads a program into ROM.
00081 /// </summary>
00082 /// <param name="bank">The bank to load data to.</param>
00083 /// <param name="data">The data to be loaded to ROM.</param>
00084 public void Load(byte bank, byte[] data)
00085 {
00086     for (int i = 0; i <= Length; i++)
00087     {
00088         Memory[bank][i] = data[i];
00089     }
00090 }
00091
00092 public byte[][] ReadFile(string filename)
00093 {
00094     byte[][] bios = new byte[Banks][];
00095     try
00096     {
00097         FileStream file = new FileStream(filename, FileMode.Open, FileAccess.Read);
00098         for (int i = 0; i < Banks; i++)
00099         {
00100             bios[i] = new byte[Length + 1];
00101             for (int j = 0; j <= Length; j++)
00102             {
00103                 bios[i][j] = new byte();
00104                 bios[i][j] = (byte)file.ReadByte();
00105             }
00106         }
00107     }
00108     catch (Exception)
00109     {
00110         return null;
00111     }
00112     return bios;
00113 }
00114
00115 /// <summary>
00116 /// Returns the byte at a given address without incrementing the cycle. Useful for test harness.
00117 /// </summary>
00118 /// <param name="bank">The bank to read data from.</param>
00119 /// <param name="address"></param>
00120 /// <returns>the byte being returned</returns>
00121 public byte Read(int address)
00122 {

```

```

00123         return Memory[CurrentBank][address - Offset];
00124     }
00125
00126     /// <summary>
00127     /// Writes data to the given address without incrementing the cycle.
00128     /// </summary>
00129     /// <param name="bank">The bank to load data to.</param>
00130     /// <param name="address">The address to write data to</param>
00131     /// <param name="data">The data to write</param>
00132     public void Write(int address, byte data)
00133     {
00134         _ = address;
00135         _ = data;
00136         return;
00137     }
00138
00139     /// <summary>
00140     /// Dumps the entire memory object. Used when saving the memory state
00141     /// </summary>
00142     /// <returns>2 dimensional array of data analogous to the ROM of the computer.</returns>
00143     public byte[][] DumpMemory()
00144     {
00145         return Memory;
00146     }
00147
00148     /// <summary>
00149     /// Dumps the selected ROM bank.
00150     /// </summary>
00151     /// <param name="bank">The bank to dump data from.</param>
00152     /// <returns>Array that represents the selected ROM bank.</returns>
00153     public byte[] DumpMemory(byte bank)
00154     {
00155         byte[] _tempMemory = new byte[MemoryMap.BankedRom.Length + 1];
00156         for (var i = 0; i < MemoryMap.BankedRom.Length; i++)
00157         {
00158             _tempMemory[i] = Memory[bank][i];
00159         }
00160         return _tempMemory;
00161     }
00162
00163     /// <summary>
00164     /// Clears the ROM.
00165     /// </summary>
00166     public void Clear()
00167     {
00168         for (byte i = 0; i < Banks; i++)
00169         {
00170             for (int j = 0; j < Length; j++)
00171             {
00172                 Memory[i][j] = 0x00;
00173             }
00174         }
00175     }
00176 #endregion
00177 }
00178 }

```

7.161 Hardware/Hardware/HM62256.cs File Reference

Classes

- class [Hardware.HM62256](#)

Namespaces

- namespace [Hardware](#)

7.162 HM62256.cs

[Go to the documentation of this file.](#)

```

00001 namespace Hardware
00002 {
00003     public class HM62256

```

```

00004     {
00005     /// <summary>
00006     /// The memory area.
00007     /// </summary>
00008     public byte[][] Memory { get; set; }
00009
00010     /// <summary>
00011     /// The memory offset.
00012     /// </summary>
00013     public int Offset { get; set; }
00014
00015     /// <summary>
00016     /// The memory length.
00017     /// </summary>
00018     public int Length { get; set; }
00019
00020     /// <summary>
00021     /// The location of the end of memory.
00022     /// </summary>
00023     public int End { get { return Offset + Length; } }
00024
00025     /// <summary>
00026     /// The number of banks the memory has.
00027     /// </summary>
00028     public byte Banks { get; set; }
00029
00030     /// <summary>
00031     /// The currently selected bank.
00032     /// </summary>
00033     public byte CurrentBank { get; set; }
00034
00035     /// <summary>
00036     /// Called whenever a new 62256 object is required.
00037     /// </summary>
00038     /// <param name="banks">Number of banks the new memory will have.</param>
00039     /// <param name="offset">Offset of the new memory in the address space.</param>
00040     /// <param name="length">Length of each bank of memory.</param>
00041     public HM62256(byte banks, int offset, int length)
00042     {
00043         Memory = new byte[banks][];
00044         for (int i = 0; i < banks; i++)
00045         {
00046             Memory[i] = new byte[length + 1];
00047         }
00048         Length = length;
00049         Banks = banks;
00050         Offset = offset;
00051         CurrentBank = 0;
00052     }
00053
00054     /// <summary>
00055     /// Called whenever the emulated computer is reset.
00056     /// </summary>
00057     public void Reset()
00058     {
00059         Clear();
00060     }
00061
00062     /// <summary>
00063     /// Clears the memory.
00064     /// </summary>
00065     public void Clear()
00066     {
00067         for (var i = 0; i < Banks; i++)
00068         {
00069             for (var j = 0; j < Memory.Length; j++)
00070             {
00071                 Memory[i][j] = 0x00;
00072             }
00073         }
00074     }
00075
00076     /// <summary>
00077     /// Returns the byte at a given address without incrementing the cycle. Useful for test harness.
00078     /// </summary>
00079     /// <param name="bank">The bank to read data from.</param>
00080     /// <param name="address"></param>
00081     /// <returns>The byte being read.</returns>
00082     public byte Read(int address)
00083     {
00084         return Memory[CurrentBank][address - Offset];
00085     }
00086
00087     /// <summary>
00088     /// Writes data to the given address without incrementing the cycle.
00089     /// </summary>
00090     /// <param name="bank">The bank to load data to.</param>

```

```

00091 /// <param name="address">The address to write data to</param>
00092 /// <param name="data">The data to write</param>
00093     public void Write(int address, byte data)
00094     {
00095         Memory[CurrentBank][address - Offset] = data;
00096     }
00097
00098 /// <summary>
00099 /// Dumps the entire memory object. Used when saving the memory state
00100 /// </summary>
00101 /// <returns>Jagged array representing the banked memory.</returns>
00102     public byte[][] DumpMemory()
00103     {
00104         return Memory;
00105     }
00106 }
00107 }

```

7.163 Hardware/Hardware/W65C02.cs File Reference

Classes

- class [Hardware.W65C02](#)
An implementation of a [W65C02 Processor](#).

Namespaces

- namespace [Hardware](#)

7.164 W65C02.cs

[Go to the documentation of this file.](#)

```

00001 using NLog;
00002 using System;
00003 using System.ComponentModel;
00004 using System.Globalization;
00005
00006 namespace Hardware
00007 {
00008     /// <summary>
00009     /// An implementation of a W65C02 Processor.
00010     /// </summary>
00011     [Serializable]
00012     public class W65C02
00013     {
00014         #region Fields
00015         private readonly ILogger _logger = LogManager.GetLogger("Processor");
00016         private int _programCounter;
00017         private int _stackPointer;
00018         private int _cycleCount;
00019         private bool _previousInterrupt;
00020         private bool _interrupt;
00021
00022         /// <summary>
00023         /// Checks shether the emulated computer is running or not.
00024         /// </summary>
00025         public bool isRunning;
00026         #endregion
00027
00028         #region Properties
00029         /// <summary>
00030         /// The Accumulator. This value is implemented as an integer intead of a byte.
00031         /// This is done so we can detect wrapping of the value and set the correct number of cycles.
00032         /// </summary>
00033         public int Accumulator { get; protected set; }
00034
00035         /// <summary>
00036         /// The X Index Register
00037         /// </summary>
00038         public int XRegister { get; private set; }
00039
00040         /// <summary>

```

```

00041 /// The Y Index Register
00042 /// </summary>
00043     public int YRegister { get; private set; }
00044
00045 /// <summary>
00046 /// The Current Op Code being executed by the system
00047 /// </summary>
00048     public int CurrentOpCode { get; private set; }
00049
00050 /// <summary>
00051 /// The disassembly of the current operation. This value is only set when the CPU is built in debug
    mode.
00052 /// </summary>
00053     public Disassembly CurrentDisassembly { get; private set; }
00054
00055 /// <summary>
00056 /// Points to the Current Address of the instruction being executed by the system.
00057 /// The PC wraps when the value is greater than 65535, or less than 0.
00058 /// </summary>
00059     public int ProgramCounter
00060     {
00061         get { return _programCounter; }
00062         private set { _programCounter = WrapProgramCounter(value); }
00063     }
00064
00065 /// <summary>
00066 /// Points to the Current Position of the Stack.
00067 /// This value is a 00-FF value but is offset to point to the location in memory where the stack
    resides.
00068 /// </summary>
00069     public int StackPointer
00070     {
00071         get { return _stackPointer; }
00072         private set
00073         {
00074             if (value > 0xFF)
00075                 _stackPointer = value - 0x100;
00076             else if (value < 0x00)
00077                 _stackPointer = value + 0x100;
00078             else
00079                 _stackPointer = value;
00080         }
00081     }
00082
00083 /// <summary>
00084 /// An external action that occurs when the cycle count is incremented
00085 /// </summary>
00086     public Action CycleCountIncrementedAction { get; set; }
00087
00088     //Status Registers
00089 /// <summary>
00090 /// This is the carry flag. when adding, if the result is greater than 255 or 99 in BCD Mode, then
    this bit is enabled.
00091 /// In subtraction this is reversed and set to false if a borrow is required IE the result is less
    than 0
00092 /// </summary>
00093     public bool CarryFlag { get; protected set; }
00094
00095 /// <summary>
00096 /// Is true if one of the registers is set to zero.
00097 /// </summary>
00098     public bool ZeroFlag { get; private set; }
00099
00100 /// <summary>
00101 /// This determines if Interrupts are currently disabled.
00102 /// This flag is turned on during a reset to prevent an interrupt from occurring during
    startup/initialization.
00103 /// If this flag is true, then the IRQ pin is ignored.
00104 /// </summary>
00105     public bool DisableInterruptFlag { get; private set; }
00106
00107 /// <summary>
00108 /// Binary Coded Decimal Mode is set/cleared via this flag.
00109 /// when this mode is in effect, a byte represents a number from 0-99.
00110 /// </summary>
00111     public bool DecimalFlag { get; private set; }
00112
00113 /// <summary>
00114 /// This property is set when an overflow occurs. An overflow happens if the high bit(7) changes
    during the operation. Remember that values from 128-256 are negative values
00115 /// as the high bit is set to 1.
00116 /// Examples:
00117 /// 64 + 64 = -128
00118 /// -128 + -128 = 0
00119 /// </summary>
00120     public bool OverflowFlag { get; protected set; }
00121

```

```

00122 /// <summary>
00123 /// Set to true if the result of an operation is negative in ADC and SBC operations.
00124 /// Remember that 128-256 represent negative numbers when doing signed math.
00125 /// In shift operations the sign holds the carry.
00126 /// </summary>
00127     public bool NegativeFlag { get; private set; }
00128
00129 /// <summary>
00130 /// Set to true when an NMI should occur
00131 /// </summary>
00132     public bool TriggerNmi { get; set; }
00133
00134 /// Set to true when an IRQ has occurred and is being processed by the CPU
00135     public bool TriggerIRQ { get; private set; }
00136 #endregion
00137
00138 #region Public Methods
00139 /// <summary>
00140 /// Default Constructor, Instantiates a new instance of the processor.
00141 /// </summary>
00142     public W65C02()
00143     {
00144         StackPointer = 0x100;
00145         CycleCountIncrementedAction = () => { };
00146     }
00147
00148 /// <summary>
00149 /// Initializes the processor to its default state.
00150 /// </summary>
00151     public void Reset()
00152     {
00153         ResetCycleCount();
00154         StackPointer = 0x1FD;
00155         //Set the Program Counter to the Reset Vector Address.
00156         ProgramCounter = 0xFFFC;
00157         //Reset the Program Counter to the Address contained in the Reset Vector
00158         ProgramCounter = (MemoryMap.Read(ProgramCounter) | (MemoryMap.Read(ProgramCounter + 1) <
00159             8));
00160         CurrentOpCode = MemoryMap.Read(ProgramCounter);
00161         //SetDisassembly();
00162         DisableInterruptFlag = true;
00163         _previousInterrupt = false;
00164         TriggerNmi = false;
00165         TriggerIRQ = false;
00166     }
00167
00168 /// <summary>
00169 /// Performs the next step on the processor
00170 /// </summary>
00171     public void NextStep()
00172     {
00173         SetDisassembly();
00174
00175         //Have to read this first otherwise it causes tests to fail on a NES
00176         CurrentOpCode = MemoryMap.Read(ProgramCounter);
00177
00178         ProgramCounter++;
00179
00180         ExecuteOpCode();
00181
00182         if (_previousInterrupt)
00183         {
00184             if (TriggerNmi)
00185             {
00186                 ProcessNMI();
00187                 TriggerNmi = false;
00188             }
00189             else if (TriggerIRQ)
00190             {
00191                 ProcessIRQ();
00192                 TriggerIRQ = false;
00193             }
00194         }
00195     }
00196
00197 /// <summary>
00198 /// The InterruptRequest or IRQ
00199 /// </summary>
00200     public void InterruptRequest()
00201     {
00202         TriggerIRQ = true;
00203     }
00204
00205 /// <summary>
00206 /// Gets the Number of Cycles that have elapsed
00207 /// </summary>
00208 /// <returns>The number of elapsed cycles</returns>

```

```

00208     public int GetCycleCount ()
00209     {
00210         return _cycleCount;
00211     }
00212
00213     /// <summary>
00214     /// Increments the Cycle Count, causes a CycleCountIncrementedAction to fire.
00215     /// </summary>
00216     public void IncrementCycleCount ()
00217     {
00218         _cycleCount++;
00219         CycleCountIncrementedAction();
00220
00221         _previousInterrupt = _interrupt;
00222         _interrupt = TriggerNmi || (TriggerIRQ && !DisableInterruptFlag);
00223     }
00224
00225     /// <summary>
00226     /// Resets the Cycle Count back to 0
00227     /// </summary>
00228     public void ResetCycleCount ()
00229     {
00230         _cycleCount = 0;
00231     }
00232 #endregion
00233
00234 #region Private Methods
00235     /// <summary>
00236     /// Executes an Opcode
00237     /// </summary>
00238     private void ExecuteOpCode ()
00239     {
00240         //The x+ cycles denotes that if a page wrap occurs, then an additional cycle is consumed.
00241         //The x++ cycles denotes that 1 cycle is added when a branch occurs and it on the same
00242         //page, and two cycles are added if its on a different page./
00243         //This is handled inside the GetValueFromMemory Method
00244         switch (CurrentOpCode)
00245         {
00246             #region Add / Subtract Operations
00247             //ADC Add With Carry, Immediate, 2 Bytes, 2 Cycles
00248             case 0x69:
00249             {
00250                 AddWithCarryOperation(AddressingMode.Immediate);
00251                 break;
00252             }
00253             //ADC Add With Carry, Zero Page, 2 Bytes, 3 Cycles
00254             case 0x65:
00255             {
00256                 AddWithCarryOperation(AddressingMode.ZeroPage);
00257                 break;
00258             }
00259             //ADC Add With Carry, Zero Page X, 2 Bytes, 4 Cycles
00260             case 0x75:
00261             {
00262                 AddWithCarryOperation(AddressingMode.ZeroPageX);
00263                 break;
00264             }
00265             //ADC Add With Carry, Absolute, 3 Bytes, 4 Cycles
00266             case 0x6D:
00267             {
00268                 AddWithCarryOperation(AddressingMode.Absolute);
00269                 break;
00270             }
00271             //ADC Add With Carry, Absolute X, 3 Bytes, 4+ Cycles
00272             case 0x7D:
00273             {
00274                 AddWithCarryOperation(AddressingMode.AbsoluteX);
00275                 break;
00276             }
00277             //ADC Add With Carry, Absolute Y, 3 Bytes, 4+ Cycles
00278             case 0x79:
00279             {
00280                 AddWithCarryOperation(AddressingMode.AbsoluteY);
00281                 break;
00282             }
00283             //ADC Add With Carry, Indexed Indirect, 2 Bytes, 6 Cycles
00284             case 0x61:
00285             {
00286                 AddWithCarryOperation(AddressingMode.IndirectX);
00287                 break;
00288             }
00289             //ADC Add With Carry, Indexed Indirect, 2 Bytes, 5+ Cycles
00290             case 0x71:
00291             {
00292                 AddWithCarryOperation(AddressingMode.IndirectY);
00293                 break;
00294             }
00295         }
00296     }

```

```

00294         //SBC Subtract with Borrow, Immediate, 2 Bytes, 2 Cycles
00295         case 0xE9:
00296         {
00297             SubtractWithBorrowOperation(AddressingMode.Immediate);
00298             break;
00299         }
00300         //SBC Subtract with Borrow, Zero Page, 2 Bytes, 3 Cycles
00301         case 0xE5:
00302         {
00303             SubtractWithBorrowOperation(AddressingMode.ZeroPage);
00304             break;
00305         }
00306         //SBC Subtract with Borrow, Zero Page X, 2 Bytes, 4 Cycles
00307         case 0xF5:
00308         {
00309             SubtractWithBorrowOperation(AddressingMode.ZeroPageX);
00310             break;
00311         }
00312         //SBC Subtract with Borrow, Absolute, 3 Bytes, 4 Cycles
00313         case 0xED:
00314         {
00315             SubtractWithBorrowOperation(AddressingMode.Absolute);
00316             break;
00317         }
00318         //SBC Subtract with Borrow, Absolute X, 3 Bytes, 4+ Cycles
00319         case 0xFD:
00320         {
00321             SubtractWithBorrowOperation(AddressingMode.AbsoluteX);
00322             break;
00323         }
00324         //SBC Subtract with Borrow, Absolute Y, 3 Bytes, 4+ Cycles
00325         case 0xF9:
00326         {
00327             SubtractWithBorrowOperation(AddressingMode.AbsoluteY);
00328             break;
00329         }
00330         //SBC Subtract with Borrow, Indexed Indirect, 2 Bytes, 6 Cycles
00331         case 0xE1:
00332         {
00333             SubtractWithBorrowOperation(AddressingMode.IndirectX);
00334             break;
00335         }
00336         //SBC Subtract with Borrow, Indexed Indirect, 2 Bytes, 5+ Cycles
00337         case 0xF1:
00338         {
00339             SubtractWithBorrowOperation(AddressingMode.IndirectY);
00340             break;
00341         }
00342     #endregion
00343
00344     #region Branch Operations
00345         //BCC Branch if Carry is Clear, Relative, 2 Bytes, 2++ Cycles
00346         case 0x90:
00347         {
00348             BranchOperation(!CarryFlag);
00349             break;
00350         }
00351     }
00352         //BCS Branch if Carry is Set, Relative, 2 Bytes, 2++ Cycles
00353         case 0xB0:
00354         {
00355             BranchOperation(CarryFlag);
00356             break;
00357         }
00358         //BEQ Branch if Zero is Set, Relative, 2 Bytes, 2++ Cycles
00359         case 0xF0:
00360         {
00361             BranchOperation(ZeroFlag);
00362             break;
00363         }
00364     }
00365         // BMI Branch if Negative Set
00366         case 0x30:
00367         {
00368             BranchOperation(NegativeFlag);
00369             break;
00370         }
00371         //BNE Branch if Zero is Not Set, Relative, 2 Bytes, 2++ Cycles
00372         case 0xD0:
00373         {
00374             BranchOperation(!ZeroFlag);
00375             break;
00376         }
00377         // BPL Branch if Negative Clear, 2 Bytes, 2++ Cycles
00378         case 0x10:
00379         {
00380             BranchOperation(!NegativeFlag);

```



```

00381         break;
00382     }
00383     // BVC Branch if Overflow Clear, 2 Bytes, 2++ Cycles
00384     case 0x50:
00385     {
00386         BranchOperation(!OverflowFlag);
00387         break;
00388     }
00389     // BVS Branch if Overflow Set, 2 Bytes, 2++ Cycles
00390     case 0x70:
00391     {
00392         BranchOperation(OverflowFlag);
00393         break;
00394     }
00395 #endregion
00396
00397 #region BitWise Comparison Operations
00398 //AND Compare Memory with Accumulator, Immediate, 2 Bytes, 2 Cycles
00399 case 0x29:
00400 {
00401     AndOperation(AddressingMode.Immediate);
00402     break;
00403 }
00404 //AND Compare Memory with Accumulator, Zero Page, 2 Bytes, 3 Cycles
00405 case 0x25:
00406 {
00407     AndOperation(AddressingMode.ZeroPage);
00408     break;
00409 }
00410 //AND Compare Memory with Accumulator, Zero PageX, 2 Bytes, 3 Cycles
00411 case 0x35:
00412 {
00413     AndOperation(AddressingMode.ZeroPageX);
00414     break;
00415 }
00416 //AND Compare Memory with Accumulator, Absolute, 3 Bytes, 4 Cycles
00417 case 0x2D:
00418 {
00419     AndOperation(AddressingMode.Absolute);
00420     break;
00421 }
00422 //AND Compare Memory with Accumulator, AbsoluteX 3 Bytes, 4+ Cycles
00423 case 0x3D:
00424 {
00425     AndOperation(AddressingMode.AbsoluteX);
00426     break;
00427 }
00428 //AND Compare Memory with Accumulator, AbsoluteY, 3 Bytes, 4+ Cycles
00429 case 0x39:
00430 {
00431     AndOperation(AddressingMode.AbsoluteY);
00432     break;
00433 }
00434 //AND Compare Memory with Accumulator, IndexedIndirect, 2 Bytes, 6 Cycles
00435 case 0x21:
00436 {
00437     AndOperation(AddressingMode.IndirectX);
00438     break;
00439 }
00440 //AND Compare Memory with Accumulator, IndirectIndexed, 2 Bytes, 5 Cycles
00441 case 0x31:
00442 {
00443     AndOperation(AddressingMode.IndirectY);
00444     break;
00445 }
00446 //BIT Compare Memory with Accumulator, Zero Page, 2 Bytes, 3 Cycles
00447 case 0x24:
00448 {
00449     BitOperation(AddressingMode.ZeroPage);
00450     break;
00451 }
00452 //BIT Compare Memory with Accumulator, Absolute, 2 Bytes, 4 Cycles
00453 case 0x2C:
00454 {
00455     BitOperation(AddressingMode.Absolute);
00456     break;
00457 }
00458 //EOR Exclusive OR Memory with Accumulator, Immediate, 2 Bytes, 2 Cycles
00459 case 0x49:
00460 {
00461     EorOperation(AddressingMode.Immediate);
00462     break;
00463 }
00464 //EOR Exclusive OR Memory with Accumulator, Zero Page, 2 Bytes, 3 Cycles
00465 case 0x45:
00466 {
00467     EorOperation(AddressingMode.ZeroPage);

```

```

00468         break;
00469     }
00470     //EOR Exclusive OR Memory with Accumulator, Zero Page X, 2 Bytes, 4 Cycles
00471     case 0x55:
00472     {
00473         EorOperation(AddressingMode.ZeroPageX);
00474         break;
00475     }
00476     //EOR Exclusive OR Memory with Accumulator, Absolute, 3 Bytes, 4 Cycles
00477     case 0x4D:
00478     {
00479         EorOperation(AddressingMode.Absolute);
00480         break;
00481     }
00482     //EOR Exclusive OR Memory with Accumulator, Absolute X, 3 Bytes, 4+ Cycles
00483     case 0x5D:
00484     {
00485         EorOperation(AddressingMode.AbsoluteX);
00486         break;
00487     }
00488     //EOR Exclusive OR Memory with Accumulator, Absolute Y, 3 Bytes, 4+ Cycles
00489     case 0x59:
00490     {
00491         EorOperation(AddressingMode.AbsoluteY);
00492         break;
00493     }
00494     //EOR Exclusive OR Memory with Accumulator, IndexedIndirect, 2 Bytes 6 Cycles
00495     case 0x41:
00496     {
00497         EorOperation(AddressingMode.IndirectX);
00498         break;
00499     }
00500     //EOR Exclusive OR Memory with Accumulator, IndirectIndexed, 2 Bytes 5 Cycles
00501     case 0x51:
00502     {
00503         EorOperation(AddressingMode.IndirectY);
00504         break;
00505     }
00506     //ORA Compare Memory with Accumulator, Immediate, 2 Bytes, 2 Cycles
00507     case 0x09:
00508     {
00509         OrOperation(AddressingMode.Immediate);
00510         break;
00511     }
00512     //ORA Compare Memory with Accumulator, Zero Page, 2 Bytes, 2 Cycles
00513     case 0x05:
00514     {
00515         OrOperation(AddressingMode.ZeroPage);
00516         break;
00517     }
00518     //ORA Compare Memory with Accumulator, Zero PageX, 2 Bytes, 4 Cycles
00519     case 0x15:
00520     {
00521         OrOperation(AddressingMode.ZeroPageX);
00522         break;
00523     }
00524     //ORA Compare Memory with Accumulator, Absolute, 3 Bytes, 4 Cycles
00525     case 0x0D:
00526     {
00527         OrOperation(AddressingMode.Absolute);
00528         break;
00529     }
00530     //ORA Compare Memory with Accumulator, AbsoluteX 3 Bytes, 4+ Cycles
00531     case 0x1D:
00532     {
00533         OrOperation(AddressingMode.AbsoluteX);
00534         break;
00535     }
00536     //ORA Compare Memory with Accumulator, AbsoluteY, 3 Bytes, 4+ Cycles
00537     case 0x19:
00538     {
00539         OrOperation(AddressingMode.AbsoluteY);
00540         break;
00541     }
00542     //ORA Compare Memory with Accumulator, IndexedIndirect, 2 Bytes, 6 Cycles
00543     case 0x01:
00544     {
00545         OrOperation(AddressingMode.IndirectX);
00546         break;
00547     }
00548     //ORA Compare Memory with Accumulator, IndirectIndexed, 2 Bytes, 5 Cycles
00549     case 0x11:
00550     {
00551         OrOperation(AddressingMode.IndirectY);
00552         break;
00553     }
00554     #endregion

```

```

00555
00556 #region Clear Flag Operations
00557     //CLC Clear Carry Flag, Implied, 1 Byte, 2 Cycles
00558     case 0x18:
00559     {
00560         CarryFlag = false;
00561         IncrementCycleCount();
00562         break;
00563     }
00564     //CLD Clear Decimal Flag, Implied, 1 Byte, 2 Cycles
00565     case 0xD8:
00566     {
00567         DecimalFlag = false;
00568         IncrementCycleCount();
00569         break;
00570     }
00571     //CLI Clear Interrupt Flag, Implied, 1 Byte, 2 Cycles
00572     case 0x58:
00573     {
00574         DisableInterruptFlag = false;
00575         IncrementCycleCount();
00576         break;
00577     }
00578     //CLV Clear Overflow Flag, Implied, 1 Byte, 2 Cycles
00579     case 0xB8:
00580     {
00581         OverflowFlag = false;
00582         IncrementCycleCount();
00583         break;
00584     }
00585     }
00586 #endregion
00587
00588 #region Compare Operations
00589     //CMP Compare Accumulator with Memory, Immediate, 2 Bytes, 2 Cycles
00590     case 0xC9:
00591     {
00592         CompareOperation(AddressingMode.Immediate, Accumulator);
00593         break;
00594     }
00595     //CMP Compare Accumulator with Memory, Zero Page, 2 Bytes, 3 Cycles
00596     case 0xC5:
00597     {
00598         CompareOperation(AddressingMode.ZeroPage, Accumulator);
00599         break;
00600     }
00601     //CMP Compare Accumulator with Memory, Zero Page x, 2 Bytes, 4 Cycles
00602     case 0xD5:
00603     {
00604         CompareOperation(AddressingMode.ZeroPageX, Accumulator);
00605         break;
00606     }
00607     //CMP Compare Accumulator with Memory, Absolute, 3 Bytes, 4 Cycles
00608     case 0xCD:
00609     {
00610         CompareOperation(AddressingMode.Absolute, Accumulator);
00611         break;
00612     }
00613     //CMP Compare Accumulator with Memory, Absolute X, 2 Bytes, 4 Cycles
00614     case 0xDD:
00615     {
00616         CompareOperation(AddressingMode.AbsoluteX, Accumulator);
00617         break;
00618     }
00619     //CMP Compare Accumulator with Memory, Absolute Y, 2 Bytes, 4 Cycles
00620     case 0xD9:
00621     {
00622         CompareOperation(AddressingMode.AbsoluteY, Accumulator);
00623         break;
00624     }
00625     //CMP Compare Accumulator with Memory, Indirect X, 2 Bytes, 6 Cycles
00626     case 0xC1:
00627     {
00628         CompareOperation(AddressingMode.IndirectX, Accumulator);
00629         break;
00630     }
00631     //CMP Compare Accumulator with Memory, Indirect Y, 2 Bytes, 5 Cycles
00632     case 0xD1:
00633     {
00634         CompareOperation(AddressingMode.IndirectY, Accumulator);
00635         break;
00636     }
00637     //CPX Compare Accumulator with X Register, Immediate, 2 Bytes, 2 Cycles
00638     case 0xE0:
00639     {
00640

```

```

00642         CompareOperation(AddressingMode.Immediate, XRegister);
00643         break;
00644     }
00645     //CPX Compare Accumulator with X Register, Zero Page, 2 Bytes, 3 Cycles
00646     case 0xE4:
00647     {
00648         CompareOperation(AddressingMode.ZeroPage, XRegister);
00649         break;
00650     }
00651     //CPX Compare Accumulator with X Register, Absolute, 3 Bytes, 4 Cycles
00652     case 0xEC:
00653     {
00654         CompareOperation(AddressingMode.Absolute, XRegister);
00655         break;
00656     }
00657     //CPY Compare Accumulator with Y Register, Immediate, 2 Bytes, 2 Cycles
00658     case 0xC0:
00659     {
00660         CompareOperation(AddressingMode.Immediate, YRegister);
00661         break;
00662     }
00663     //CPY Compare Accumulator with Y Register, Zero Page, 2 Bytes, 3 Cycles
00664     case 0xC4:
00665     {
00666         CompareOperation(AddressingMode.ZeroPage, YRegister);
00667         break;
00668     }
00669     //CPY Compare Accumulator with Y Register, Absolute, 3 Bytes, 4 Cycles
00670     case 0xCC:
00671     {
00672         CompareOperation(AddressingMode.Absolute, YRegister);
00673         break;
00674     }
00675 #endregion
00676
00677 #region Increment/Decrement Operations
00678     //DEC Decrement Memory by One, Zero Page, 2 Bytes, 5 Cycles
00679     case 0xC6:
00680     {
00681         ChangeMemoryByOne(AddressingMode.ZeroPage, true);
00682         break;
00683     }
00684     //DEC Decrement Memory by One, Zero Page X, 2 Bytes, 6 Cycles
00685     case 0xD6:
00686     {
00687         ChangeMemoryByOne(AddressingMode.ZeroPageX, true);
00688         break;
00689     }
00690     //DEC Decrement Memory by One, Absolute, 3 Bytes, 6 Cycles
00691     case 0xCE:
00692     {
00693         ChangeMemoryByOne(AddressingMode.Absolute, true);
00694         break;
00695     }
00696     //DEC Decrement Memory by One, Absolute X, 3 Bytes, 7 Cycles
00697     case 0xDE:
00698     {
00699         ChangeMemoryByOne(AddressingMode.AbsoluteX, true);
00700         IncrementCycleCount();
00701         break;
00702     }
00703     //DEX Decrement X Register by One, Implied, 1 Bytes, 2 Cycles
00704     case 0xCA:
00705     {
00706         ChangeRegisterByOne(true, true);
00707         break;
00708     }
00709     //DEY Decrement Y Register by One, Implied, 1 Bytes, 2 Cycles
00710     case 0x88:
00711     {
00712         ChangeRegisterByOne(false, true);
00713         break;
00714     }
00715     //INC Increment Memory by One, Zero Page, 2 Bytes, 5 Cycles
00716     case 0xE6:
00717     {
00718         ChangeMemoryByOne(AddressingMode.ZeroPage, false);
00719         break;
00720     }
00721     //INC Increment Memory by One, Zero Page X, 2 Bytes, 6 Cycles
00722     case 0xF6:
00723     {
00724         ChangeMemoryByOne(AddressingMode.ZeroPageX, false);
00725         break;
00726     }
00727     //INC Increment Memory by One, Absolute, 3 Bytes, 6 Cycles
00728     case 0xEE:

```

```

00729         {
00730             ChangeMemoryByOne(AddressingMode.Absolute, false);
00731             break;
00732         }
00733         //INC Increment Memory by One, Absolute X, 3 Bytes, 7 Cycles
00734         case 0xFE:
00735         {
00736             ChangeMemoryByOne(AddressingMode.AbsoluteX, false);
00737             IncrementCycleCount();
00738             break;
00739         }
00740         //INX Increment X Register by One, Implied, 1 Bytes, 2 Cycles
00741         case 0xE8:
00742         {
00743             ChangeRegisterByOne(true, false);
00744             break;
00745         }
00746         //INY Increment Y Register by One, Implied, 1 Bytes, 2 Cycles
00747         case 0xC8:
00748         {
00749             ChangeRegisterByOne(false, false);
00750             break;
00751         }
00752     #endregion
00753
00754     #region GOTO and GOSUB Operations
00755     //JMP Jump to New Location, Absolute 3 Bytes, 3 Cycles
00756     case 0x4C:
00757     {
00758         ProgramCounter = GetAddressByAddressingMode(AddressingMode.Absolute);
00759         break;
00760     }
00761     //JMP Jump to New Location, Indirect 3 Bytes, 5 Cycles
00762     case 0x6C:
00763     {
00764         ProgramCounter = GetAddressByAddressingMode(AddressingMode.Absolute);
00765
00766         if ((ProgramCounter & 0xFF) == 0xFF)
00767         {
00768             //Get the first half of the address
00769             int address = MemoryMap.Read(ProgramCounter);
00770
00771             //Get the second half of the address, due to the issue with page boundary
00772             //it reads from the wrong location!
00773             address += 256 * MemoryMap.Read(ProgramCounter - 255);
00774             ProgramCounter = address;
00775         }
00776         else
00777         {
00778             ProgramCounter = GetAddressByAddressingMode(AddressingMode.Absolute);
00779         }
00780         break;
00781     }
00782     //JSR Jump to SubRoutine, Absolute, 3 Bytes, 6 Cycles
00783     case 0x20:
00784     {
00785         JumpToSubRoutineOperation();
00786         break;
00787     }
00788     //BRK Simulate IRQ, Implied, 1 Byte, 7 Cycles
00789     case 0x00:
00790     {
00791         BreakOperation(true, 0xFFFE);
00792         break;
00793     }
00794     //RTI Return From Interrupt, Implied, 1 Byte, 6 Cycles
00795     case 0x40:
00796     {
00797         ReturnFromInterruptOperation();
00798         break;
00799     }
00800     //RTS Return From Subroutine, Implied, 1 Byte, 6 Cycles
00801     case 0x60:
00802     {
00803         ReturnFromSubRoutineOperation();
00804         break;
00805     }
00806 #endregion
00807
00808 #region Load Value From Memory Operations
00809 //LDA Load Accumulator with Memory, Immediate, 2 Bytes, 2 Cycles
00810 case 0xA9:
00811 {
00812     Accumulator =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.Immediate));
00813     SetZeroFlag(Accumulator);

```

```

00814             SetNegativeFlag(Accumulator);
00815             break;
00816         }
00817         //LDA Load Accumulator with Memory, Zero Page, 2 Bytes, 3 Cycles
00818         case 0xA5:
00819         {
00820             Accumulator =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.ZeroPage));
00821             SetZeroFlag(Accumulator);
00822             SetNegativeFlag(Accumulator);
00823             break;
00824         }
00825         //LDA Load Accumulator with Memory, Zero Page X, 2 Bytes, 4 Cycles
00826         case 0xB5:
00827         {
00828             Accumulator =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.ZeroPageX));
00829             SetZeroFlag(Accumulator);
00830             SetNegativeFlag(Accumulator);
00831             break;
00832         }
00833         //LDA Load Accumulator with Memory, Absolute, 3 Bytes, 4 Cycles
00834         case 0xAD:
00835         {
00836             Accumulator =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.Absolute));
00837             SetZeroFlag(Accumulator);
00838             SetNegativeFlag(Accumulator);
00839             break;
00840         }
00841         //LDA Load Accumulator with Memory, Absolute X, 3 Bytes, 4+ Cycles
00842         case 0xBD:
00843         {
00844             Accumulator =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.AbsoluteX));
00845             SetZeroFlag(Accumulator);
00846             SetNegativeFlag(Accumulator);
00847             break;
00848         }
00849         //LDA Load Accumulator with Memory, Absolute Y, 3 Bytes, 4+ Cycles
00850         case 0xB9:
00851         {
00852             Accumulator =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.AbsoluteY));
00853             SetZeroFlag(Accumulator);
00854             SetNegativeFlag(Accumulator);
00855             break;
00856         }
00857         //LDA Load Accumulator with Memory, Index Indirect, 2 Bytes, 6 Cycles
00858         case 0xA1:
00859         {
00860             Accumulator =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.IndirectX));
00861             SetZeroFlag(Accumulator);
00862             SetNegativeFlag(Accumulator);
00863             break;
00864         }
00865         //LDA Load Accumulator with Memory, Indirect Index, 2 Bytes, 5+ Cycles
00866         case 0xB1:
00867         {
00868             Accumulator =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.IndirectY));
00869             SetZeroFlag(Accumulator);
00870             SetNegativeFlag(Accumulator);
00871             break;
00872         }
00873         //LDX Load X with memory, Immediate, 2 Bytes, 2 Cycles
00874         case 0xA2:
00875         {
00876             XRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.Immediate));
00877             SetZeroFlag(XRegister);
00878             SetNegativeFlag(XRegister);
00879             break;
00880         }
00881         //LDX Load X with memory, Zero Page, 2 Bytes, 3 Cycles
00882         case 0xA6:
00883         {
00884             XRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.ZeroPage));
00885             SetZeroFlag(XRegister);
00886             SetNegativeFlag(XRegister);
00887             break;
00888         }
00889         //LDX Load X with memory, Zero Page Y, 2 Bytes, 4 Cycles
00890         case 0xB6:
00891         {

```

```

00892             XRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.ZeroPageY));
00893             SetZeroFlag(XRegister);
00894             SetNegativeFlag(XRegister);
00895             break;
00896         }
00897         //LDX Load X with memory, Absolute, 3 Bytes, 4 Cycles
00898         case 0xAE:
00899         {
00900             XRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.Absolute));
00901             SetZeroFlag(XRegister);
00902             SetNegativeFlag(XRegister);
00903             break;
00904         }
00905         //LDX Load X with memory, Absolute Y, 3 Bytes, 4+ Cycles
00906         case 0xBE:
00907         {
00908             XRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.AbsoluteY));
00909             SetZeroFlag(XRegister);
00910             SetNegativeFlag(XRegister);
00911             break;
00912         }
00913         //LDY Load Y with memory, Immediate, 2 Bytes, 2 Cycles
00914         case 0xA0:
00915         {
00916             YRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.Immediate));
00917             SetZeroFlag(YRegister);
00918             SetNegativeFlag(YRegister);
00919             break;
00920         }
00921         //LDY Load Y with memory, Zero Page, 2 Bytes, 3 Cycles
00922         case 0xA4:
00923         {
00924             YRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.ZeroPage));
00925             SetZeroFlag(YRegister);
00926             SetNegativeFlag(YRegister);
00927             break;
00928         }
00929         //LDY Load Y with memory, Zero Page X, 2 Bytes, 4 Cycles
00930         case 0xB4:
00931         {
00932             YRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.ZeroPageX));
00933             SetZeroFlag(YRegister);
00934             SetNegativeFlag(YRegister);
00935             break;
00936         }
00937         //LDY Load Y with memory, Absolute, 3 Bytes, 4 Cycles
00938         case 0xAC:
00939         {
00940             YRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.Absolute));
00941             SetZeroFlag(YRegister);
00942             SetNegativeFlag(YRegister);
00943             break;
00944         }
00945         //LDY Load Y with memory, Absolute X, 3 Bytes, 4+ Cycles
00946         case 0xBC:
00947         {
00948             YRegister =
MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.AbsoluteX));
00949             SetZeroFlag(YRegister);
00950             SetNegativeFlag(YRegister);
00951             break;
00952         }
00953     #endregion
00954
00955     #region Push/Pull Stack
00956     //PHA Push Accumulator onto Stack, Implied, 1 Byte, 3 Cycles
00957     case 0x48:
00958     {
00959         MemoryMap.Read(ProgramCounter + 1);
00960
00961         PokeStack((byte)Accumulator);
00962         StackPointer--;
00963         IncrementCycleCount();
00964         break;
00965     }
00966
00967     //PHP Push Flags onto Stack, Implied, 1 Byte, 3 Cycles
00968     case 0x08:
00969     {
00970         MemoryMap.Read(ProgramCounter + 1);

```

```

00971
00972         PushFlagsOperation();
00973         StackPointer--;
00974         IncrementCycleCount();
00975         break;
00976     }
00977     //PLA Pull Accumulator from Stack, Implied, 1 Byte, 4 Cycles
00978     case 0x68:
00979     {
00980         MemoryMap.Read(ProgramCounter + 1);
00981         StackPointer++;
00982         IncrementCycleCount();
00983
00984         Accumulator = PeekStack();
00985         SetNegativeFlag(Accumulator);
00986         SetZeroFlag(Accumulator);
00987
00988         IncrementCycleCount();
00989         break;
00990     }
00991     //PLP Pull Flags from Stack, Implied, 1 Byte, 4 Cycles
00992     case 0x28:
00993     {
00994         MemoryMap.Read(ProgramCounter + 1);
00995
00996         StackPointer++;
00997         IncrementCycleCount();
00998
00999         PullFlagsOperation();
01000
01001         IncrementCycleCount();
01002         break;
01003     }
01004     //TSX Transfer Stack Pointer to X Register, 1 Bytes, 2 Cycles
01005     case 0xBA:
01006     {
01007         XRegister = StackPointer;
01008
01009         SetNegativeFlag(XRegister);
01010         SetZeroFlag(XRegister);
01011         IncrementCycleCount();
01012         break;
01013     }
01014     //TXS Transfer X Register to Stack Pointer, 1 Bytes, 2 Cycles
01015     case 0x9A:
01016     {
01017         StackPointer = (byte)XRegister;
01018         IncrementCycleCount();
01019         break;
01020     }
01021 #endregion
01022
01023 #region Set Flag Operations
01024     //SEC Set Carry, Implied, 1 Bytes, 2 Cycles
01025     case 0x38:
01026     {
01027         CarryFlag = true;
01028         IncrementCycleCount();
01029         break;
01030     }
01031     //SED Set Interrupt, Implied, 1 Bytes, 2 Cycles
01032     case 0xF8:
01033     {
01034         DecimalFlag = true;
01035         IncrementCycleCount();
01036         break;
01037     }
01038     //SEI Set Interrupt, Implied, 1 Bytes, 2 Cycles
01039     case 0x78:
01040     {
01041         DisableInterruptFlag = true;
01042         IncrementCycleCount();
01043         break;
01044     }
01045 #endregion
01046
01047 #region Shift/Rotate Operations
01048     //ASL Shift Left 1 Bit Memory or Accumulator, Accumulator, 1 Bytes, 2 Cycles
01049     case 0x0A:
01050     {
01051         AslOperation(AddressingMode.Accumulator);
01052         break;
01053     }
01054     //ASL Shift Left 1 Bit Memory or Accumulator, Zero Page, 2 Bytes, 5 Cycles
01055     case 0x06:
01056     {
01057         AslOperation(AddressingMode.ZeroPage);

```



```

01058         break;
01059     }
01060     //ASL Shift Left 1 Bit Memory or Accumulator, Zero PageX, 2 Bytes, 6 Cycles
01061     case 0x16:
01062     {
01063         AslOperation(AddressingMode.ZeroPageX);
01064         break;
01065     }
01066     //ASL Shift Left 1 Bit Memory or Accumulator, Absolute, 3 Bytes, 6 Cycles
01067     case 0x0E:
01068     {
01069         AslOperation(AddressingMode.Absolute);
01070         break;
01071     }
01072     //ASL Shift Left 1 Bit Memory or Accumulator, AbsoluteX, 3 Bytes, 7 Cycles
01073     case 0x1E:
01074     {
01075         AslOperation(AddressingMode.AbsoluteX);
01076         IncrementCycleCount();
01077         break;
01078     }
01079     //LSR Shift Left 1 Bit Memory or Accumulator, Accumulator, 1 Bytes, 2 Cycles
01080     case 0x4A:
01081     {
01082         LsrOperation(AddressingMode.Accumulator);
01083         break;
01084     }
01085     //LSR Shift Left 1 Bit Memory or Accumulator, Zero Page, 2 Bytes, 5 Cycles
01086     case 0x46:
01087     {
01088         LsrOperation(AddressingMode.ZeroPage);
01089         break;
01090     }
01091     //LSR Shift Left 1 Bit Memory or Accumulator, Zero PageX, 2 Bytes, 6 Cycles
01092     case 0x56:
01093     {
01094         LsrOperation(AddressingMode.ZeroPageX);
01095         break;
01096     }
01097     //LSR Shift Left 1 Bit Memory or Accumulator, Absolute, 3 Bytes, 6 Cycles
01098     case 0x4E:
01099     {
01100         LsrOperation(AddressingMode.Absolute);
01101         break;
01102     }
01103     //LSR Shift Left 1 Bit Memory or Accumulator, AbsoluteX, 3 Bytes, 7 Cycles
01104     case 0x5E:
01105     {
01106         LsrOperation(AddressingMode.AbsoluteX);
01107         IncrementCycleCount();
01108         break;
01109     }
01110     //ROL Rotate Left 1 Bit Memory or Accumulator, Accumulator, 1 Bytes, 2 Cycles
01111     case 0x2A:
01112     {
01113         RolOperation(AddressingMode.Accumulator);
01114         break;
01115     }
01116     //ROL Rotate Left 1 Bit Memory or Accumulator, Zero Page, 2 Bytes, 5 Cycles
01117     case 0x26:
01118     {
01119         RolOperation(AddressingMode.ZeroPage);
01120         break;
01121     }
01122     //ROL Rotate Left 1 Bit Memory or Accumulator, Zero PageX, 2 Bytes, 6 Cycles
01123     case 0x36:
01124     {
01125         RolOperation(AddressingMode.ZeroPageX);
01126         break;
01127     }
01128     //ROL Rotate Left 1 Bit Memory or Accumulator, Absolute, 3 Bytes, 6 Cycles
01129     case 0x2E:
01130     {
01131         RolOperation(AddressingMode.Absolute);
01132         break;
01133     }
01134     //ROL Rotate Left 1 Bit Memory or Accumulator, AbsoluteX, 3 Bytes, 7 Cycles
01135     case 0x3E:
01136     {
01137         RolOperation(AddressingMode.AbsoluteX);
01138         IncrementCycleCount();
01139         break;
01140     }
01141     //ROR Rotate Right 1 Bit Memory or Accumulator, Accumulator, 1 Bytes, 2 Cycles
01142     case 0x6A:
01143     {
01144         RorOperation(AddressingMode.Accumulator);

```

```

01145         break;
01146     }
01147     //ROR Rotate Right 1 Bit Memory or Accumulator, Zero Page, 2 Bytes, 5 Cycles
01148     case 0x66:
01149     {
01150         RorOperation(AddressingMode.ZeroPage);
01151         break;
01152     }
01153     //ROR Rotate Right 1 Bit Memory or Accumulator, Zero PageX, 2 Bytes, 6 Cycles
01154     case 0x76:
01155     {
01156         RorOperation(AddressingMode.ZeroPageX);
01157         break;
01158     }
01159     //ROR Rotate Right 1 Bit Memory or Accumulator, Absolute, 3 Bytes, 6 Cycles
01160     case 0x6E:
01161     {
01162         RorOperation(AddressingMode.Absolute);
01163         break;
01164     }
01165     //ROR Rotate Right 1 Bit Memory or Accumulator, AbsoluteX, 3 Bytes, 7 Cycles
01166     case 0x7E:
01167     {
01168         RorOperation(AddressingMode.AbsoluteX);
01169         IncrementCycleCount();
01170         break;
01171     }
01172 #endregion
01173
01174 #region Store Value In Memory Operations
01175     //STA Store Accumulator In Memory, Zero Page, 2 Bytes, 3 Cycles
01176     case 0x85:
01177     {
01178         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.ZeroPage),
01179 (byte)Accumulator);
01179         break;
01180     }
01181     //STA Store Accumulator In Memory, Zero Page X, 2 Bytes, 4 Cycles
01182     case 0x95:
01183     {
01184         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.ZeroPageX),
01185 (byte)Accumulator);
01185         break;
01186     }
01187     //STA Store Accumulator In Memory, Absolute, 3 Bytes, 4 Cycles
01188     case 0x8D:
01189     {
01190         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.Absolute),
01191 (byte)Accumulator);
01191         break;
01192     }
01193     //STA Store Accumulator In Memory, Absolute X, 3 Bytes, 5 Cycles
01194     case 0x9D:
01195     {
01196         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.AbsoluteX),
01197 (byte)Accumulator);
01197         IncrementCycleCount();
01198         break;
01199     }
01200     //STA Store Accumulator In Memory, Absolute Y, 3 Bytes, 5 Cycles
01201     case 0x99:
01202     {
01203         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.AbsoluteY),
01204 (byte)Accumulator);
01204         IncrementCycleCount();
01205         break;
01206     }
01207     //STA Store Accumulator In Memory, Indexed Indirect, 2 Bytes, 6 Cycles
01208     case 0x81:
01209     {
01210         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.IndirectX),
01211 (byte)Accumulator);
01211         break;
01212     }
01213     //STA Store Accumulator In Memory, Indirect Indexed, 2 Bytes, 6 Cycles
01214     case 0x91:
01215     {
01216         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.IndirectY),
01217 (byte)Accumulator);
01217         IncrementCycleCount();
01218         break;
01219     }
01220     //STX Store Index X, Zero Page, 2 Bytes, 3 Cycles
01221     case 0x86:
01222     {
01223         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.ZeroPage),
01224 (byte)XRegister);

```

```

01224             break;
01225         }
01226         //STX Store Index X, Zero Page Y, 2 Bytes, 4 Cycles
01227         case 0x96:
01228         {
01229             MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.ZeroPageY),
01230 (byte)XRegister);
01231             break;
01232         }
01233         //STX Store Index X, Absolute, 3 Bytes, 4 Cycles
01234         case 0x8E:
01235         {
01236             MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.Absolute),
01237 (byte)XRegister);
01238             break;
01239         }
01240         //STY Store Index Y, Zero Page X, 2 Bytes, 3 Cycles
01241         case 0x84:
01242         {
01243             MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.ZeroPageX),
01244 (byte)YRegister);
01245             break;
01246         }
01247         //STY Store Index Y, Absolute, 2 Bytes, 4 Cycles
01248         case 0x8C:
01249         {
01250             MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.Absolute),
01251 (byte)YRegister);
01252             break;
01253         }
01254     }
01255 #endregion
01256 #region Transfer Operations
01257 //TAX Transfer Accumulator to X Register, Implied, 1 Bytes, 2 Cycles
01258 case 0xAA:
01259 {
01260     IncrementCycleCount();
01261     XRegister = Accumulator;
01262     SetNegativeFlag(XRegister);
01263     SetZeroFlag(XRegister);
01264     break;
01265 }
01266 //TAY Transfer Accumulator to Y Register, 1 Bytes, 2 Cycles
01267 case 0xA8:
01268 {
01269     IncrementCycleCount();
01270     YRegister = Accumulator;
01271     SetNegativeFlag(YRegister);
01272     SetZeroFlag(YRegister);
01273     break;
01274 }
01275 //TXA Transfer X Register to Accumulator, Implied, 1 Bytes, 2 Cycles
01276 case 0x8A:
01277 {
01278     IncrementCycleCount();
01279     Accumulator = XRegister;
01280     SetNegativeFlag(Accumulator);
01281     SetZeroFlag(Accumulator);
01282     break;
01283 }
01284 //TYA Transfer Y Register to Accumulator, Implied, 1 Bytes, 2 Cycles
01285 case 0x98:
01286 {
01287     IncrementCycleCount();
01288     Accumulator = YRegister;
01289     SetNegativeFlag(Accumulator);
01290     SetZeroFlag(Accumulator);
01291     break;
01292 }
01293 #endregion
01294 //NOP Operation, Implied, 1 Byte, 2 Cycles
01295 case 0xEA:
01296 {
01297     IncrementCycleCount();
01298     break;
01299 }

```

```

01306         }
01307
01308         default:
01309             throw new NotSupportedException(string.Format("The OpCode {0} is not supported",
CurrentOpCode));
01310     }
01311 }
01312
01313 /// <summary>
01314 /// Sets the IsSignNegative register
01315 /// </summary>
01316 /// <param name="value"></param>
01317 protected void SetNegativeFlag(int value)
01318 {
01319     //on the 6502, any value greater than 127 is negative. 128 = 1000000 in Binary. the 8th
bit is set, therefore the number is a negative number.
01320     NegativeFlag = value > 127;
01321 }
01322
01323 /// <summary>
01324 /// Sets the IsResultZero register
01325 /// </summary>
01326 /// <param name="value"></param>
01327 protected void SetZeroFlag(int value)
01328 {
01329     ZeroFlag = value == 0;
01330 }
01331
01332 /// <summary>
01333 /// Uses the AddressingMode to return the correct address based on the mode.
01334 /// Note: This method will not increment the program counter for any mode.
01335 /// Note: This method will return an error if called for either the immediate or accumulator modes.
01336 /// </summary>
01337 /// <param name="addressingMode">The addressing Mode to use</param>
01338 /// <returns>The memory Location</returns>
01339 protected int GetAddressByAddressingMode(AddressingMode addressingMode)
01340 {
01341     int address;
01342     int highByte;
01343     switch (addressingMode)
01344     {
01345         case (AddressingMode.Absolute):
01346             return (MemoryMap.Read(ProgramCounter++) | (MemoryMap.Read(ProgramCounter++) <
8));
01347     }
01348
01349     case AddressingMode.AbsoluteX:
01350     {
01351         //Get the low half of the address
01352         address = MemoryMap.Read(ProgramCounter++);
01353
01354         //Get the high byte
01355         highByte = MemoryMap.Read(ProgramCounter++);
01356
01357         //We crossed a page boundary, so an extra read has occurred.
01358         //However, if this is an ASL, LSR, DEC, INC, ROR, ROL or STA operation, we do
not decrease it by 1.
01359         if (address + XRegister > 0xFF)
01360         {
01361             switch (CurrentOpCode)
01362             {
01363                 case 0x1E:
01364                 case 0xDE:
01365                 case 0xFE:
01366                 case 0x5E:
01367                 case 0x3E:
01368                 case 0x7E:
01369                 case 0x9D:
01370                 {
01371                     //This is a MemoryMap.Read Fetch Write Operation, so we don't
make the extra read.
01372                     return ((highByte < 8 | address) + XRegister) & 0xFFFF;
01373                 }
01374                 default:
01375                 {
01376                     MemoryMap.Read((((highByte < 8 | address) + XRegister) - 0xFF)
& 0xFFFF);
01377                     break;
01378                 }
01379             }
01380         }
01381
01382         return ((highByte < 8 | address) + XRegister) & 0xFFFF;
01383     }
01384     case AddressingMode.AbsoluteY:
01385     {
01386         //Get the low half of the address

```

```

01387         address = MemoryMap.Read(ProgramCounter++);
01388
01389         //Get the high byte
01390         highByte = MemoryMap.Read(ProgramCounter++);
01391
01392         //We crossed a page boundary, so decrease the number of cycles by 1 if the
operation is not STA
01393         if (address + YRegister > 0xFF && CurrentOpCode != 0x99)
01394         {
01395             MemoryMap.Read((((highByte < 8 | address) + YRegister) - 0xFF) & 0xFFFF);
01396         }
01397
01398         //Bitshift the high byte into place, AND with FFFF to handle wrapping.
01399         return ((highByte < 8 | address) + YRegister) & 0xFFFF;
01400     }
01401     case AddressingMode.Immediate:
01402     {
01403         return ProgramCounter++;
01404     }
01405     case AddressingMode.IndirectX:
01406     {
01407         //Get the location of the address to retrieve
01408         address = MemoryMap.Read(ProgramCounter++);
01409         MemoryMap.Read(address);
01410
01411         address += XRegister;
01412
01413         //Now get the final Address. The is not a zero page address either.
01414         var finalAddress = MemoryMap.Read((address & 0xFF) | (MemoryMap.Read((address
+ 1) & 0xFF) < 8));
01415         return finalAddress;
01416     }
01417     case AddressingMode.IndirectY:
01418     {
01419         address = MemoryMap.Read(ProgramCounter++);
01420
01421         var finalAddress = MemoryMap.Read(address) + (MemoryMap.Read((address + 1) &
0xFF) < 8);
01422
01423         if ((finalAddress & 0xFF) + YRegister > 0xFF && CurrentOpCode != 0x91)
01424         {
01425             MemoryMap.Read((finalAddress + YRegister - 0xFF) & 0xFFFF);
01426         }
01427
01428         return (finalAddress + YRegister) & 0xFFFF;
01429     }
01430     case AddressingMode.Relative:
01431     {
01432         return ProgramCounter;
01433     }
01434     case (AddressingMode.ZeroPage):
01435     {
01436         address = MemoryMap.Read(ProgramCounter++);
01437         return address;
01438     }
01439     case (AddressingMode.ZeroPageX):
01440     {
01441         address = MemoryMap.Read(ProgramCounter++);
01442         MemoryMap.Read(address);
01443
01444         address += XRegister;
01445         address &= 0xFF;
01446
01447         //This address wraps if its greater than 0xFF
01448         if (address > 0xFF)
01449         {
01450             address -= 0x100;
01451             return address;
01452         }
01453
01454         return address;
01455     }
01456     case (AddressingMode.ZeroPageY):
01457     {
01458         address = MemoryMap.Read(ProgramCounter++);
01459         MemoryMap.Read(address);
01460
01461         address += YRegister;
01462         address &= 0xFF;
01463
01464         return address;
01465     }
01466     default:
01467         throw new InvalidOperationException(string.Format("The Address Mode '{0}' does not
require an address", addressingMode));
01468     }
01469 }

```

```

01470
01471 /// <summary>
01472 /// Moves the ProgramCounter in a given direction based on the value inputted
01473 ///
01474 /// </summary>
01475 private void MoveProgramCounterByRelativeValue(byte valueToMove)
01476 {
01477     var movement = valueToMove > 127 ? (valueToMove - 255) : valueToMove;
01478
01479     var newProgramCounter = ProgramCounter + movement;
01480
01481     //This makes sure that we always land on the correct spot for a positive number
01482     if (movement >= 0)
01483         newProgramCounter++;
01484
01485     //We Crossed a Page Boundary. So we increment the cycle counter by one. The +1 is
    because we always check from the end of the instruction not the beginning
01486     if (((ProgramCounter + 1 ^ newProgramCounter) & 0xff00) != 0x0000)
01487     {
01488         IncrementCycleCount();
01489     }
01490
01491     ProgramCounter = newProgramCounter;
01492     MemoryMap.Read(ProgramCounter);
01493 }
01494
01495 /// <summary>
01496 /// Returns a the value from the stack without changing the position of the stack pointer
01497 /// </summary>
01498 /// <returns>The value at the current Stack Pointer</returns>
01499 private byte PeekStack()
01500 {
01501     //The stack lives at 0x100-0x1FF, but the value is only a byte so it needs to be
    translated
01502     return MemoryMap.Read(StackPointer + 0x100);
01503 }
01504
01505 /// <summary>
01506 /// Write a value directly to the stack without modifying the Stack Pointer
01507 /// </summary>
01508 ///
01509 /// <param name="value">The value to be written to the stack</param>
01510 private void PokeStack(byte value)
01511 {
01512     //The stack lives at 0x100-0x1FF, but the value is only a byte so it needs to be
    translated
01513     MemoryMap.Write(StackPointer + 0x100, value);
01514 }
01515
01516 /// <summary>
01517 /// Converts the Flags into its byte representation.
01518 /// </summary>
01519 /// <param name="setBreak">Determines if the break flag should be set during conversion. IRQ does not
    set the flag on the stack, but PHP and BRK do</param>
01520 /// <returns></returns>
01521 private byte ConvertFlagsToByte(bool setBreak)
01522 {
01523     return (byte)((CarryFlag ? 0x01 : 0) + (ZeroFlag ? 0x02 : 0) + (DisableInterruptFlag ?
    0x04 : 0) +
01524         (DecimalFlag ? 8 : 0) + (setBreak ? 0x10 : 0) + 0x20 + (OverflowFlag ? 0x40 : 0)
    + (NegativeFlag ? 0x80 : 0));
01525 }
01526
01527 private void SetDisassembly()
01528 {
01529     var addressMode = GetAddressingMode();
01530
01531     var currentProgramCounter = ProgramCounter;
01532
01533     currentProgramCounter = WrapProgramCounter(++currentProgramCounter);
01534     int? address1 = MemoryMap.Read(currentProgramCounter);
01535
01536     currentProgramCounter = WrapProgramCounter(++currentProgramCounter);
01537     int? address2 = MemoryMap.Read(currentProgramCounter);
01538
01539     string disassembledStep = string.Empty;
01540
01541     switch (addressMode)
01542     {
01543         case AddressingMode.Absolute:
01544         {
01545             disassembledStep = string.Format("${0}{1}",
    address2.Value.ToString("X").PadLeft(2, '0'), address1.Value.ToString("X").PadLeft(2, '0'));
01546             break;
01547         }
01548         case AddressingMode.AbsoluteX:
01549         {

```

```

01550             disassembledStep = string.Format("${0}{1},X",
address2.Value.ToString("X").PadLeft(2, '0'), address1.Value.ToString("X").PadLeft(2, '0'));
01551             break;
01552         }
01553         case AddressingMode.AbsoluteY:
01554         {
01555             disassembledStep = string.Format("${0}{1},Y",
address2.Value.ToString("X").PadLeft(2, '0'), address1.Value.ToString("X").PadLeft(2, '0'));
01556             break;
01557         }
01558         case AddressingMode.Accumulator:
01559         {
01560             address1 = null;
01561             address2 = null;
01562
01563             disassembledStep = "A";
01564             break;
01565         }
01566         case AddressingMode.Immediate:
01567         {
01568             disassembledStep = string.Format("#${0}",
address1.Value.ToString("X").PadLeft(4, '0'));
01569             address2 = null;
01570             break;
01571         }
01572         case AddressingMode.Implied:
01573         {
01574             address1 = null;
01575             address2 = null;
01576             break;
01577         }
01578         case AddressingMode.Indirect:
01579         {
01580             disassembledStep = string.Format("${0}{1}",
address2.Value.ToString("X").PadLeft(2, '0'), address1.Value.ToString("X").PadLeft(2, '0'));
01581             break;
01582         }
01583         case AddressingMode.IndirectX:
01584         {
01585             address2 = null;
01586
01587             disassembledStep = string.Format("${0},X",
address1.Value.ToString("X").PadLeft(2, '0'));
01588             break;
01589         }
01590         case AddressingMode.IndirectY:
01591         {
01592             address2 = null;
01593
01594             disassembledStep = string.Format("${0},Y",
address1.Value.ToString("X").PadLeft(2, '0'));
01595             break;
01596         }
01597         case AddressingMode.Relative:
01598         {
01599             var valueToMove = (byte)address1.Value;
01600
01601             var movement = valueToMove > 127 ? (valueToMove - 255) : valueToMove;
01602
01603             var newProgramCounter = ProgramCounter + movement;
01604
01605             //This makes sure that we always land on the correct spot for a positive
number
01606             if (movement >= 0)
01607                 newProgramCounter++;
01608
01609             var stringAddress = ProgramCounter.ToString("X").PadLeft(4, '0');
01610
01611             address1 = int.Parse(stringAddress.Substring(0, 2),
NumberStyles.AllowHexSpecifier);
01612             address2 = int.Parse(stringAddress.Substring(2, 2),
NumberStyles.AllowHexSpecifier);
01613
01614             disassembledStep = string.Format("${0}",
newProgramCounter.ToString("X").PadLeft(4, '0'));
01615             break;
01616         }
01617         case AddressingMode.ZeroPage:
01618         {
01619             address2 = null;
01620
01621             disassembledStep = string.Format("${0}",
address1.Value.ToString("X").PadLeft(2, '0'));
01622             break;
01623         }
01624         case AddressingMode.ZeroPageX:

```

```

01626         {
01627             address2 = null;
01628
01629             disassembledStep = string.Format("${0},X",
address1.Value.ToString("X").PadLeft(2, '0'));
01630             break;
01631         }
01632         case AddressingMode.ZeroPageY:
01633         {
01634             address2 = null;
01635
01636             disassembledStep = string.Format("${0},Y",
address1.Value.ToString("X").PadLeft(4, '0'));
01637             break;
01638         }
01639         default:
01640             throw new InvalidEnumArgumentException("Invalid Addressing Mode");
01641     }
01642 }
01643
01644 CurrentDisassembly = new Disassembly
01645 {
01646     HighAddress = address2.HasValue ? address2.Value.ToString("X").PadLeft(2, '0') :
string.Empty,
01648     LowAddress = address1.HasValue ? address1.Value.ToString("X").PadLeft(2, '0') :
string.Empty,
01649     OpCodeString = CurrentOpCode.ConvertOpCodeIntoString(),
01650     DisassemblyOutput = disassembledStep
01651 };
01652
01653 _logger.Debug("{0} : {1}{2}{3} {4} {5} A: {6} X: {7} Y: {8} SP {9} N: {10} V: {11} B:
{12} D: {13} I: {14} Z: {15} C: {16}",
01654     ProgramCounter.ToString("X").PadLeft(4, '0'),
01655     CurrentOpCode.ToString("X").PadLeft(2, '0'),
01656     CurrentDisassembly.LowAddress,
01657     CurrentDisassembly.HighAddress,
01658     CurrentDisassembly.OpCodeString,
01659     CurrentDisassembly.DisassemblyOutput.PadRight(10, ' '),
01660     Accumulator.ToString("X").PadLeft(3, '0'),
01661     XRegister.ToString("X").PadLeft(3, '0'),
01662     YRegister.ToString("X").PadLeft(3, '0'),
01663     StackPointer.ToString("X").PadLeft(3, '0'),
01664     Convert.ToInt16(NegativeFlag),
01665     Convert.ToInt16(OverflowFlag),
01666     0,
01667     Convert.ToInt16(DecimalFlag),
01668     Convert.ToInt16(DisableInterruptFlag),
01669     Convert.ToInt16(ZeroFlag),
01670     Convert.ToInt16(CarryFlag));
01671 }
01672
01673 private int WrapProgramCounter(int value)
01674 {
01675     return value & 0xFFFF;
01676 }
01677
01678 private AddressingMode GetAddressingMode()
01679 {
01680     switch (CurrentOpCode)
01681     {
01682     case 0x0D: //ORA
01683     case 0x2D: //AND
01684     case 0x4D: //EOR
01685     case 0x6D: //ADC
01686     case 0x8D: //STA
01687     case 0xAD: //LDA
01688     case 0xCD: //CMP
01689     case 0xED: //SBC
01690     case 0x0E: //ASL
01691     case 0x2E: //ROL
01692     case 0x4E: //LSR
01693     case 0x6E: //ROR
01694     case 0x8E: //SDX
01695     case 0xAE: //LDX
01696     case 0xCE: //DEC
01697     case 0xEE: //INC
01698     case 0x2C: //Bit
01699     case 0x4C: //JMP
01700     case 0x8C: //STY
01701     case 0xAC: //LDY
01702     case 0xCC: //CPY
01703     case 0xEC: //CPX
01704     case 0x20: //JSR
01705     }

```



```

01708         return AddressingMode.Absolute;
01709     }
01710     case 0x1D: //ORA
01711     case 0x3D: //AND
01712     case 0x5D: //EOR
01713     case 0x7D: //ADC
01714     case 0x9D: //STA
01715     case 0xBD: //LDA
01716     case 0xDD: //CMP
01717     case 0xFD: //SBC
01718     case 0xBC: //LDY
01719     case 0xFE: //INC
01720     case 0x1E: //ASL
01721     case 0x3E: //ROL
01722     case 0x5E: //LSR
01723     case 0x7E: //ROR
01724     {
01725         return AddressingMode.AbsoluteX;
01726     }
01727     case 0x19: //ORA
01728     case 0x39: //AND
01729     case 0x59: //EOR
01730     case 0x79: //ADC
01731     case 0x99: //STA
01732     case 0xB9: //LDA
01733     case 0xD9: //CMP
01734     case 0xF9: //SBC
01735     case 0xBE: //LDX
01736     {
01737         return AddressingMode.AbsoluteY;
01738     }
01739     case 0x0A: //ASL
01740     case 0x4A: //LSR
01741     case 0x2A: //ROL
01742     case 0x6A: //ROR
01743     {
01744         return AddressingMode.Accumulator;
01745     }
01746
01747     case 0x09: //ORA
01748     case 0x29: //AND
01749     case 0x49: //EOR
01750     case 0x69: //ADC
01751     case 0xA0: //LDY
01752     case 0xC0: //CPY
01753     case 0xE0: //CMP
01754     case 0xA2: //LDX
01755     case 0xA9: //LDA
01756     case 0xC9: //CMP
01757     case 0xE9: //SBC
01758     {
01759         return AddressingMode.Immediate;
01760     }
01761     case 0x00: //BRK
01762     case 0x18: //CLC
01763     case 0xD8: //CLD
01764     case 0x58: //CLI
01765     case 0xB8: //CLV
01766     case 0xDE: //DEC
01767     case 0xCA: //DEX
01768     case 0x88: //DEY
01769     case 0xE8: //INX
01770     case 0xC8: //INY
01771     case 0xEA: //NOP
01772     case 0x48: //PHA
01773     case 0x08: //PHP
01774     case 0x68: //PLA
01775     case 0x28: //PLP
01776     case 0x40: //RTI
01777     case 0x60: //RTS
01778     case 0x38: //SEC
01779     case 0xF8: //SED
01780     case 0x78: //SEI
01781     case 0xAA: //TAX
01782     case 0xA8: //TAY
01783     case 0xBA: //TSX
01784     case 0x8A: //TXA
01785     case 0x9A: //TXS
01786     case 0x98: //TYA
01787     {
01788         return AddressingMode.Implied;
01789     }
01790     case 0x6C:
01791     {
01792         return AddressingMode.Indirect;
01793     }
01794

```

```

01795         case 0x61: //ADC
01796         case 0x21: //AND
01797         case 0xC1: //CMP
01798         case 0x41: //EOR
01799         case 0xA1: //LDA
01800         case 0x01: //ORA
01801         case 0xE1: //SBC
01802         case 0x81: //STA
01803         {
01804             return AddressingMode.IndirectX;
01805         }
01806         case 0x71: //ADC
01807         case 0x31: //AND
01808         case 0xD1: //CMP
01809         case 0x51: //EOR
01810         case 0xB1: //LDA
01811         case 0x11: //ORA
01812         case 0xF1: //SBC
01813         case 0x91: //STA
01814         {
01815             return AddressingMode.IndirectY;
01816         }
01817         case 0x90: //BCC
01818         case 0xB0: //BCS
01819         case 0xF0: //BEQ
01820         case 0x30: //BMI
01821         case 0xD0: //BNE
01822         case 0x10: //BPL
01823         case 0x50: //BVC
01824         case 0x70: //BVS
01825         {
01826             return AddressingMode.Relative;
01827         }
01828         case 0x65: //ADC
01829         case 0x25: //AND
01830         case 0x06: //ASL
01831         case 0x24: //BIT
01832         case 0xC5: //CMP
01833         case 0xE4: //CPX
01834         case 0xC4: //CPY
01835         case 0xC6: //DEC
01836         case 0x45: //EOR
01837         case 0xE6: //INC
01838         case 0xA5: //LDA
01839         case 0xA6: //LDX
01840         case 0xA4: //LDY
01841         case 0x46: //LSR
01842         case 0x05: //ORA
01843         case 0x26: //ROL
01844         case 0x66: //ROR
01845         case 0xE5: //SBC
01846         case 0x85: //STA
01847         case 0x86: //STX
01848         case 0x84: //STY
01849         {
01850             return AddressingMode.ZeroPage;
01851         }
01852         case 0x75: //ADC
01853         case 0x35: //AND
01854         case 0x16: //ASL
01855         case 0xD5: //CMP
01856         case 0xD6: //DEC
01857         case 0x55: //EOR
01858         case 0xF6: //INC
01859         case 0xB5: //LDA
01860         case 0xB6: //LDX
01861         case 0xB4: //LDY
01862         case 0x56: //LSR
01863         case 0x15: //ORA
01864         case 0x36: //ROL
01865         case 0x76: //ROR
01866         case 0xF5: //SBC
01867         case 0x95: //STA
01868         case 0x96: //STX
01869         case 0x94: //STY
01870         {
01871             return AddressingMode.ZeroPageX;
01872         }
01873         default:
01874             throw new NotSupportedException(string.Format("Opcode {0} is not supported",
CurrentOpCode));
01875     }
01876 }
01877
01878 #region Op Code Operations
01879 /// <summary>
01880 /// The ADC - Add Memory to Accumulator with Carry Operation

```

```

01881 /// </summary>
01882 /// <param name="addressingMode">The addressing mode used to perform this operation.</param>
01883 protected void AddWithCarryOperation(AddressingMode addressingMode)
01884 {
01885     //Accumulator, Carry = Accumulator + ValueInMemoryLocation + Carry
01886     var memoryValue = MemoryMap.Read(GetAddressByAddressingMode(addressingMode));
01887     var newValue = memoryValue + Accumulator + (CarryFlag ? 1 : 0);
01888
01889
01890     OverflowFlag = (((Accumulator ^ newValue) & 0x80) != 0) && (((Accumulator ^ memoryValue) &
01891 0x80) == 0);
01892
01893     if (DecimalFlag)
01894     {
01895         newValue = int.Parse(memoryValue.ToString("x")) + int.Parse(Accumulator.ToString("x"))
01896 + (CarryFlag ? 1 : 0);
01897
01898         if (newValue > 99)
01899         {
01900             CarryFlag = true;
01901             newValue -= 100;
01902         }
01903         else
01904         {
01905             CarryFlag = false;
01906         }
01907         newValue = (int)Convert.ToInt64(string.Concat("0x", newValue), 16);
01908     }
01909     else
01910     {
01911         if (newValue > 255)
01912         {
01913             CarryFlag = true;
01914             newValue -= 256;
01915         }
01916         else
01917         {
01918             CarryFlag = false;
01919         }
01920     }
01921     SetZeroFlag(newValue);
01922     SetNegativeFlag(newValue);
01923
01924     Accumulator = newValue;
01925 }
01926
01927 /// <summary>
01928 /// The AND - Compare Memory with Accumulator operation
01929 /// </summary>
01930 /// <param name="addressingMode">The addressing mode being used</param>
01931 private void AndOperation(AddressingMode addressingMode)
01932 {
01933     Accumulator = MemoryMap.Read(GetAddressByAddressingMode(addressingMode)) & Accumulator;
01934
01935     SetZeroFlag(Accumulator);
01936     SetNegativeFlag(Accumulator);
01937 }
01938
01939 /// <summary>
01940 /// The ASL - Shift Left One Bit (Memory or Accumulator)
01941 /// </summary>
01942 /// <param name="addressingMode">The addressing Mode being used</param>
01943 public void AslOperation(AddressingMode addressingMode)
01944 {
01945     int value;
01946     var memoryAddress = 0;
01947     if (addressingMode == AddressingMode.Accumulator)
01948     {
01949         MemoryMap.Read(ProgramCounter + 1);
01950         value = Accumulator;
01951     }
01952     else
01953     {
01954         memoryAddress = GetAddressByAddressingMode(addressingMode);
01955         value = MemoryMap.Read(memoryAddress);
01956     }
01957
01958     //Dummy Write
01959     if (addressingMode != AddressingMode.Accumulator)
01960     {
01961         MemoryMap.Write(memoryAddress, (byte)value);
01962     }
01963
01964     //If the 7th bit is set, then we have a carry
01965     CarryFlag = ((value & 0x80) != 0);

```

```

01966
01967         //The And here ensures that if the value is greater than 255 it wraps properly.
01968         value = (value << 1) & 0xFE;
01969
01970         SetNegativeFlag(value);
01971         SetZeroFlag(value);
01972
01973
01974         if (addressingMode == AddressingMode.Accumulator)
01975             Accumulator = value;
01976         else
01977         {
01978             MemoryMap.Write(memoryAddress, (byte)value);
01979         }
01980     }
01981
01982     /// <summary>
01983     /// Performs the different branch operations.
01984     /// </summary>
01985     /// <param name="performBranch">Is a branch required</param>
01986     private void BranchOperation(bool performBranch)
01987     {
01988         var value = MemoryMap.Read(GetAddressByAddressingMode(AddressingMode.Relative));
01989
01990         if (!performBranch)
01991         {
01992             ProgramCounter++;
01993             return;
01994         }
01995
01996         MoveProgramCounterByRelativeValue(value);
01997     }
01998
01999     /// <summary>
02000     /// The bit operation, does an & comparison between a value in memory and the accumulator
02001     /// </summary>
02002     /// <param name="addressingMode"></param>
02003     private void BitOperation(AddressingMode addressingMode)
02004     {
02005
02006         var memoryValue = MemoryMap.Read(GetAddressByAddressingMode(addressingMode));
02007         var valueToCompare = memoryValue & Accumulator;
02008
02009         OverflowFlag = (memoryValue & 0x40) != 0;
02010
02011         SetNegativeFlag(memoryValue);
02012         SetZeroFlag(valueToCompare);
02013     }
02014
02015     /// <summary>
02016     /// The compare operation. This operation compares a value in memory with a value passed into it.
02017     /// </summary>
02018     /// <param name="addressingMode">The addressing mode to use</param>
02019     /// <param name="comparisonValue">The value to compare against memory</param>
02020     private void CompareOperation(AddressingMode addressingMode, int comparisonValue)
02021     {
02022         var memoryValue = MemoryMap.Read(GetAddressByAddressingMode(addressingMode));
02023         var comparedValue = comparisonValue - memoryValue;
02024
02025         if (comparedValue < 0)
02026             comparedValue += 0x10000;
02027
02028         SetZeroFlag(comparedValue);
02029
02030         CarryFlag = memoryValue <= comparisonValue;
02031         SetNegativeFlag(comparedValue);
02032     }
02033
02034     /// <summary>
02035     /// Changes a value in memory by 1
02036     /// </summary>
02037     /// <param name="addressingMode">The addressing mode to use</param>
02038     /// <param name="decrement">If the operation is decrementing or incrementing the vaulue by 1</param>
02039     private void ChangeMemoryByOne(AddressingMode addressingMode, bool decrement)
02040     {
02041         var memoryLocation = GetAddressByAddressingMode(addressingMode);
02042         var memory = MemoryMap.Read(memoryLocation);
02043
02044         MemoryMap.Write(memoryLocation, memory);
02045
02046         if (decrement)
02047             memory -= 1;
02048         else
02049             memory += 1;
02050
02051         SetZeroFlag(memory);
02052         SetNegativeFlag(memory);

```

```

02053
02054
02055         MemoryMap.Write(memoryLocation, memory);
02056     }
02057
02058     /// <summary>
02059     /// Changes a value in either the X or Y register by 1
02060     /// </summary>
02061     /// <param name="useXRegister">If the operation is using the X or Y register</param>
02062     /// <param name="decrement">If the operation is decrementing or incrementing the value by 1 </param>
02063     private void ChangeRegisterByOne(bool useXRegister, bool decrement)
02064     {
02065         var value = useXRegister ? XRegister : YRegister;
02066
02067         if (decrement)
02068             value -= 1;
02069         else
02070             value += 1;
02071
02072         if (value < 0x00)
02073             value += 0x100;
02074         else if (value > 0xFF)
02075             value -= 0x100;
02076
02077         SetZeroFlag(value);
02078         SetNegativeFlag(value);
02079         IncrementCycleCount();
02080
02081         if (useXRegister)
02082             XRegister = value;
02083         else
02084             YRegister = value;
02085     }
02086
02087     /// <summary>
02088     /// The EOR Operation, Performs an Exclusive OR Operation against the Accumulator and a value in
02089     memory
02090     /// </summary>
02091     /// <param name="addressingMode">The addressing mode to use</param>
02092     private void EorOperation(AddressingMode addressingMode)
02093     {
02094         Accumulator = Accumulator ^ MemoryMap.Read(GetAddressByAddressingMode(addressingMode));
02095
02096         SetNegativeFlag(Accumulator);
02097         SetZeroFlag(Accumulator);
02098     }
02099     /// <summary>
02100     /// The LSR Operation. Performs a Left shift operation on a value in memory
02101     /// </summary>
02102     /// <param name="addressingMode">The addressing mode to use</param>
02103     private void LsrOperation(AddressingMode addressingMode)
02104     {
02105         int value;
02106         var memoryAddress = 0;
02107         if (addressingMode == AddressingMode.Accumulator)
02108         {
02109             MemoryMap.Read(ProgramCounter + 1);
02110             value = Accumulator;
02111         }
02112         else
02113         {
02114             memoryAddress = GetAddressByAddressingMode(addressingMode);
02115             value = MemoryMap.Read(memoryAddress);
02116         }
02117
02118         //Dummy Write
02119         if (addressingMode != AddressingMode.Accumulator)
02120         {
02121             MemoryMap.Write(memoryAddress, (byte)value);
02122         }
02123
02124         NegativeFlag = false;
02125
02126         //If the Zero bit is set, we have a carry
02127         CarryFlag = (value & 0x01) != 0;
02128
02129         value = (value >> 1);
02130
02131         SetZeroFlag(value);
02132         if (addressingMode == AddressingMode.Accumulator)
02133             Accumulator = value;
02134         else
02135         {
02136             MemoryMap.Write(memoryAddress, (byte)value);
02137         }
02138     }

```

```

02139
02140 /// <summary>
02141 /// The Or Operation. Performs an Or Operation with the accumulator and a value in memory
02142 /// </summary>
02143 /// <param name="addressingMode">The addressing mode to use</param>
02144 private void OrOperation(AddressingMode addressingMode)
02145 {
02146     Accumulator = Accumulator | MemoryMap.Read(GetAddressByAddressingMode(addressingMode));
02147
02148     SetNegativeFlag(Accumulator);
02149     SetZeroFlag(Accumulator);
02150 }
02151
02152 /// <summary>
02153 /// The ROL operation. Performs a rotate left operation on a value in memory.
02154 /// </summary>
02155 /// <param name="addressingMode">The addressing mode to use</param>
02156 private void RolOperation(AddressingMode addressingMode)
02157 {
02158     int value;
02159     var memoryAddress = 0;
02160     if (addressingMode == AddressingMode.Accumulator)
02161     {
02162         //Dummy MemoryMap.Read
02163         MemoryMap.Read(ProgramCounter + 1);
02164         value = Accumulator;
02165     }
02166     else
02167     {
02168         memoryAddress = GetAddressByAddressingMode(addressingMode);
02169         value = MemoryMap.Read(memoryAddress);
02170     }
02171
02172     //Dummy Write
02173     if (addressingMode != AddressingMode.Accumulator)
02174     {
02175         MemoryMap.Write(memoryAddress, (byte)value);
02176     }
02177
02178     //Store the carry flag before shifting it
02179     var newCarry = (0x80 & value) != 0;
02180
02181     //The And here ensures that if the value is greater than 255 it wraps properly.
02182     value = (value << 1) & 0xFE;
02183
02184     if (CarryFlag)
02185         value = value | 0x01;
02186
02187     CarryFlag = newCarry;
02188
02189     SetZeroFlag(value);
02190     SetNegativeFlag(value);
02191
02192
02193     if (addressingMode == AddressingMode.Accumulator)
02194         Accumulator = value;
02195     else
02196     {
02197         MemoryMap.Write(memoryAddress, (byte)value);
02198     }
02199 }
02200
02201 /// <summary>
02202 /// The ROR operation. Performs a rotate right operation on a value in memory.
02203 /// </summary>
02204 /// <param name="addressingMode">The addressing mode to use</param>
02205 private void RorOperation(AddressingMode addressingMode)
02206 {
02207     int value;
02208     var memoryAddress = 0;
02209     if (addressingMode == AddressingMode.Accumulator)
02210     {
02211         //Dummy MemoryMap.Read
02212         MemoryMap.Read(ProgramCounter + 1);
02213         value = Accumulator;
02214     }
02215     else
02216     {
02217         memoryAddress = GetAddressByAddressingMode(addressingMode);
02218         value = MemoryMap.Read(memoryAddress);
02219     }
02220
02221     //Dummy Write
02222     if (addressingMode != AddressingMode.Accumulator)
02223     {
02224         MemoryMap.Write(memoryAddress, (byte)value);
02225     }

```

```

02226
02227         //Store the carry flag before shifting it
02228         var newCarry = (0x01 & value) != 0;
02229
02230         value = (value » 1);
02231
02232         //If the carry flag is set then 0x
02233         if (CarryFlag)
02234             value = value | 0x80;
02235
02236         CarryFlag = newCarry;
02237
02238         SetZeroFlag(value);
02239         SetNegativeFlag(value);
02240
02241         if (addressingMode == AddressingMode.Accumulator)
02242             Accumulator = value;
02243         else
02244         {
02245             MemoryMap.Write(memoryAddress, (byte)value);
02246         }
02247     }
02248
02249     /// <summary>
02250     /// The SBC operation. Performs a subtract with carry operation on the accumulator and a value in
02251     /// memory.
02252     /// <param name="addressingMode">The addressing mode to use</param>
02253     protected void SubtractWithBorrowOperation(AddressingMode addressingMode)
02254     {
02255         var memoryValue = MemoryMap.Read(GetAddressByAddressingMode(addressingMode));
02256         var newValue = DecimalFlag ? int.Parse(Accumulator.ToString("x")) -
int.Parse(memoryValue.ToString("x")) - (CarryFlag ? 0 : 1) : Accumulator - memoryValue - (CarryFlag
? 0 : 1);
02257
02258         CarryFlag = newValue >= 0;
02259
02260         if (DecimalFlag)
02261         {
02262             if (newValue < 0)
02263                 newValue += 100;
02264
02265             newValue = (int)Convert.ToInt64(string.Concat("0x", newValue), 16);
02266         }
02267         else
02268         {
02269             OverflowFlag = (((Accumulator ^ newValue) & 0x80) != 0) && (((Accumulator ^
memoryValue) & 0x80) != 0);
02270
02271             if (newValue < 0)
02272                 newValue += 256;
02273         }
02274
02275         SetNegativeFlag(newValue);
02276         SetZeroFlag(newValue);
02277
02278         Accumulator = newValue;
02279     }
02280
02281     /// <summary>
02282     /// The PSP Operation. Pushes the Status Flags to the stack
02283     /// </summary>
02284     private void PushFlagsOperation()
02285     {
02286         PokeStack(ConvertFlagsToByte(true));
02287     }
02288
02289     /// <summary>
02290     /// The PLP Operation. Pull the status flags off the stack on sets the flags accordingly.
02291     /// </summary>
02292     private void PullFlagsOperation()
02293     {
02294         var flags = PeekStack();
02295         CarryFlag = (flags & 0x01) != 0;
02296         ZeroFlag = (flags & 0x02) != 0;
02297         DisableInterruptFlag = (flags & 0x04) != 0;
02298         DecimalFlag = (flags & 0x08) != 0;
02299         OverflowFlag = (flags & 0x40) != 0;
02300         NegativeFlag = (flags & 0x80) != 0;
02301
02302     }
02303
02304
02305     /// <summary>
02306     /// The JSR routine. Jumps to a subroutine.
02307     /// </summary>
02308     private void JumpToSubRoutineOperation()

```

```

02309     {
02310         IncrementCycleCount();
02311
02312         //Put the high value on the stack, this should be the address after our operation -1
02313         //The RTS operation increments the PC by 1 which is why we don't move 2
02314         PokeStack((byte)((ProgramCounter + 1) >> 8) & 0xFF));
02315         StackPointer--;
02316         IncrementCycleCount();
02317
02318         PokeStack((byte)((ProgramCounter + 1) & 0xFF));
02319         StackPointer--;
02320         IncrementCycleCount();
02321
02322         ProgramCounter = GetAddressByAddressingMode(AddressingMode.Absolute);
02323     }
02324
02325     /// <summary>
02326     /// The RTS routine. Called when returning from a subroutine.
02327     /// </summary>
02328     private void ReturnFromSubRoutineOperation()
02329     {
02330         MemoryMap.Read(++ProgramCounter);
02331         StackPointer++;
02332         IncrementCycleCount();
02333
02334         var lowBit = PeekStack();
02335         StackPointer++;
02336         IncrementCycleCount();
02337
02338         var highBit = PeekStack() << 8;
02339         IncrementCycleCount();
02340
02341         ProgramCounter = (highBit | lowBit) + 1;
02342         IncrementCycleCount();
02343     }
02344
02345     /// <summary>
02346     /// The BRK routine. Called when a BRK occurs.
02347     /// </summary>
02348     private void BreakOperation(bool isBrk, int vector)
02349     {
02350         MemoryMap.Read(++ProgramCounter);
02351
02352         //Put the high value on the stack
02353         //When we RTI the address will be incremented by one, and the address after a break will
02354         not be used.
02355         PokeStack((byte)((ProgramCounter) >> 8) & 0xFF));
02356         StackPointer--;
02357         IncrementCycleCount();
02358
02359         //Put the low value on the stack
02360         PokeStack((byte)((ProgramCounter) & 0xFF));
02361         StackPointer--;
02362         IncrementCycleCount();
02363
02364         //We only set the Break Flag is a Break Occurs
02365         if (isBrk)
02366             PokeStack((byte)(ConvertFlagsToByte(true) | 0x10));
02367         else
02368             PokeStack(ConvertFlagsToByte(false));
02369
02370         StackPointer--;
02371         IncrementCycleCount();
02372
02373         DisableInterruptFlag = true;
02374
02375         ProgramCounter = (MemoryMap.Read(vector + 1) << 8) | MemoryMap.Read(vector);
02376
02377         _previousInterrupt = false;
02378     }
02379
02380     /// <summary>
02381     /// The RTI routine. Called when returning from a BRK operation.
02382     /// Note: when called after a BRK operation the Program Counter is not set to the location after the
02383     /// BRK,
02384     /// it is set +1
02385     /// </summary>
02386     private void ReturnFromInterruptOperation()
02387     {
02388         MemoryMap.Read(++ProgramCounter);
02389         StackPointer++;
02390         IncrementCycleCount();
02391
02392         PullFlagsOperation();
02393         StackPointer++;
02394         IncrementCycleCount();
02395     }

```



```

02394
02395         var lowBit = PeekStack();
02396         StackPointer++;
02397         IncrementCycleCount();
02398
02399         var highBit = PeekStack() << 8;
02400         IncrementCycleCount();
02401
02402         ProgramCounter = (highBit | lowBit);
02403     }
02404
02405     /// <summary>
02406     /// This is ran anytime an NMI occurs
02407     /// </summary>
02408     private void ProcessNMI()
02409     {
02410         ProgramCounter--;
02411         BreakOperation(false, 0xFFFFA);
02412         CurrentOpCode = MemoryMap.Read(ProgramCounter);
02413
02414         SetDisassembly();
02415     }
02416
02417     /// <summary>
02418     /// This is ran anytime an IRQ occurs
02419     /// </summary>
02420     private void ProcessIRQ()
02421     {
02422         if (DisableInterruptFlag)
02423             return;
02424
02425         ProgramCounter--;
02426         BreakOperation(false, 0xFFFFE);
02427         CurrentOpCode = MemoryMap.Read(ProgramCounter);
02428
02429         SetDisassembly();
02430     }
02431 #endregion
02432
02433 #endregion
02434     }
02435 }

```

7.165 Hardware/Hardware/W65C22.cs File Reference

Classes

- class [Hardware.W65C22](#)
An implementation of a [W65C22](#) VIA.

Namespaces

- namespace [Hardware](#)

7.166 W65C22.cs

[Go to the documentation of this file.](#)

```

00001 using System;
00002 using System.Timers;
00003
00004 namespace Hardware
00005 {
00006     /// <summary>
00007     /// An implementation of a W65C22 VIA.
00008     /// </summary>
00009     [Serializable]
00010     public class W65C22
00011     {
00012         #region Fields
00013         public readonly bool T1IsIRQ = false;
00014         public readonly bool T2IsIRQ = true;
00015         public int T1CL = 0x04;

```

```

00016         public int T1CH = 0x05;
00017         public int T2CL = 0x08;
00018         public int T2CH = 0x09;
00019         public int ACR = 0x0B;
00020         public int IFR = 0x0D;
00021         public int IER = 0x0E;
00022
00023         public byte ACR_T1TC = (byte)(1 << 7);
00024         public byte ACR_T2TC = (byte)(1 << 6);
00025
00026         public byte IFR_T2 = (byte)(1 << 5);
00027         public byte IFR_T1 = (byte)(1 << 6);
00028         public byte IFR_INT = (byte)(1 << 7);
00029
00030         public byte IER_T2 = (byte)(1 << 5);
00031         public byte IER_T1 = (byte)(1 << 6);
00032         public byte IER_EN = (byte)(1 << 7);
00033 #endregion
00034
00035 #region Properties
00036 /// <summary>
00037 /// The memory area.
00038 /// </summary>
00039         public byte[] Memory { get; set; }
00040
00041 /// <summary>
00042 /// The memory offset of the device.
00043 /// </summary>
00044         public int Offset { get; set; }
00045
00046 /// <summary>
00047 /// The length of the device memory.
00048 /// </summary>
00049         public int Length { get; set; }
00050
00051 /// <summary>
00052 /// The end of memory
00053 /// </summary>
00054         public int End { get { return Offset + Length; } }
00055
00056 /// <summary>
00057 /// T1 timer control
00058 /// </summary>
00059         public bool T1TimerControl
00060         {
00061             get { return T1Object.AutoReset; }
00062             set { T1Object.AutoReset = value; }
00063         }
00064
00065 /// <summary>
00066 /// T2 timer control.
00067 /// </summary>
00068         public bool T2TimerControl
00069         {
00070             get { return T2Object.AutoReset; }
00071             set { T2Object.AutoReset = value; }
00072         }
00073
00074 /// <summary>
00075 /// Enable or check whether timer 1 is enabled or not.
00076 /// </summary>
00077         public bool T1IsEnabled
00078         {
00079             get { return T1Object.Enabled; }
00080             set { T1Object.Enabled = value; }
00081         }
00082
00083 /// <summary>
00084 /// Enable or check whether timer 2 is enabled or not.
00085 /// </summary>
00086         public bool T2IsEnabled
00087         {
00088             get { return T2Object.Enabled; }
00089             set { T2Object.Enabled = value; }
00090         }
00091
00092 /// <summary>
00093 /// Set or check the timer 1 interval.
00094 /// </summary>
00095         public double T1Interval { get { return (int)(Read(T1CL) | (Read(T1CH) << 8)); } }
00096
00097 /// <summary>
00098 /// Set or check the timer 2 interval.
00099 /// </summary>
00100         public double T2Interval
00101         {
00102             get { return (int)(Read(T2CL) | (Read(T2CH) << 8)); }

```

```

00103     }
00104
00105     /// <summary>
00106     /// Set or get the timer 1 object.
00107     /// </summary>
00108     public Timer T1Object { get; set; }
00109
00110     /// <summary>
00111     /// Set or get the timer 2 object.
00112     /// </summary>
00113     public Timer T2Object { get; set; }
00114
00115     /// <summary>
00116     /// Local reference to the processor object.
00117     /// </summary>
00118     private W65C02 Processor { get; set; }
00119 #endregion
00120
00121 #region Public Methods
00122     public W65C22(W65C02 processor, byte offset, int length)
00123     {
00124         if (offset > MemoryMap.DeviceArea.Length)
00125             throw new ArgumentException(String.Format("The offset: {0} is greater than the device
area: {1}", offset, MemoryMap.DeviceArea.Length));
00126         T1Init(1000);
00127         T2Init(1000);
00128
00129         Offset = MemoryMap.DeviceArea.Offset | offset;
00130         Memory = new byte[length + 1];
00131         Length = length;
00132         Processor = processor;
00133     }
00134
00135     /// <summary>
00136     /// Reset routine called whenever the emulated computer is reset.
00137     /// </summary>
00138     public void Reset()
00139     {
00140         T1TimerControl = false;
00141         T1IsEnabled = false;
00142         T2TimerControl = false;
00143         T2IsEnabled = false;
00144     }
00145
00146     /// <summary>
00147     /// Initialization routine for the VIA.
00148     /// </summary>
00149     /// <param name="timer">Amount of time to set timers for.</param>
00150     public void Init(double timer)
00151     {
00152         T1Init(timer);
00153         T2Init(timer);
00154     }
00155
00156     /// <summary>
00157     /// T1 counter initialization routine.
00158     /// </summary>
00159     ///
00160     /// <param name="value">Timer initialization value in milliseconds.</param>
00161     public void T1Init(double value)
00162     {
00163         T1Object = new Timer(value);
00164         T1Object.Start();
00165         T1Object.Elapsed += OnT1Timeout;
00166         T1TimerControl = true;
00167         T1IsEnabled = false;
00168     }
00169
00170     /// <summary>
00171     /// T2 counter initialization routine.
00172     /// </summary>
00173     ///
00174     /// <param name="value">Timer initialization value in milliseconds.</param>
00175     public void T2Init(double value)
00176     {
00177         T2Object = new Timer(value);
00178         T2Object.Start();
00179         T2Object.Elapsed += OnT2Timeout;
00180         T2TimerControl = true;
00181         T2IsEnabled = false;
00182     }
00183
00184     /// <summary>
00185     /// Routine to read from local memory.
00186     /// </summary>
00187     ///
00188     /// <param name="address">Address to read from.</param>

```

```

00189 ///
00190 /// <returns>Byte value stored in the local memory.</returns>
00191 public byte Read(int address)
00192 {
00193     if ((Offset <= address) && (address <= End))
00194     {
00195         byte data = 0x00;
00196         if (T1TimerControl)
00197         {
00198             data = (byte)(data | ACR_T1TC);
00199         }
00200         else if (T2TimerControl)
00201         {
00202             data = (byte)(data | ACR_T2TC);
00203         }
00204         return data;
00205     }
00206     else
00207     {
00208         return Memory[address - Offset];
00209     }
00210 }
00211
00212 /// <summary>
00213 /// Writes data to the specified address in local memory.
00214 /// </summary>
00215 ///
00216 /// <param name="address">The address to write data to.</param>
00217 /// <param name="data">The data to be written.</param>
00218 public void Write(int address, byte data)
00219 {
00220     if ((address == Offset + ACR) && ((data | ACR_T1TC) == ACR_T1TC))
00221     {
00222         T1TimerControl = true;
00223     }
00224     else if ((address == Offset + ACR) && ((data | ACR_T2TC) == ACR_T2TC))
00225     {
00226         T2TimerControl = true;
00227     }
00228     else if ((address == Offset + IER) && ((data | IER_T1) == IER_T1) && ((data | IER_EN) ==
IER_EN))
00229     {
00230         T1Init(T1Interval);
00231     }
00232     else if ((address == Offset + IER) && ((data | IER_T2) == IER_T2) && ((data | IER_EN) ==
IER_EN))
00233     {
00234         T2Init(T2Interval);
00235     }
00236     Memory[address - Offset] = data;
00237 }
00238 #endregion
00239
00240 #region Private Methods
00241 /// <summary>
00242 /// Called whenever System.Timers.Timer event elapses.
00243 /// </summary>
00244 ///
00245 /// <param name="sender"></param>
00246 /// <param name="e"></param>
00247 private void OnT1Timeout(object sender, ElapsedEventArgs e)
00248 {
00249     if (Processor.isRunning)
00250     {
00251         if (T1IsEnabled)
00252         {
00253             Write(IFR, (byte)(IFR_T1 & IFR_INT));
00254             if (T1IsIRQ)
00255             {
00256                 Processor.InterruptRequest();
00257             }
00258             else
00259             {
00260                 Processor.TriggerNmi = true;
00261             }
00262         }
00263     }
00264 }
00265
00266 /// <summary>
00267 /// Called whenever System.Timers.Timer event elapses
00268 /// </summary>
00269 ///
00270 /// <param name="sender"></param>
00271 /// <param name="e"></param>
00272 private void OnT2Timeout(object sender, ElapsedEventArgs e)
00273 {

```

```

00274         if (Processor.isRunning)
00275         {
00276             if (T2IsEnabled)
00277             {
00278                 Write(IFR, (byte)(IFR_T2 & IFR_INT));
00279                 if (T2IsIRQ)
00280                 {
00281                     Processor.InterruptRequest();
00282                 }
00283                 else
00284                 {
00285                     Processor.TriggerNmi = true;
00286                 }
00287             }
00288         }
00289     }
00290 #endregion
00291 }
00292 }

```

7.167 Hardware/Hardware/W65C51.cs File Reference

Classes

- class [Hardware.W65C51](#)
An implementation of a [W65C51](#) ACIA.

Namespaces

- namespace [Hardware](#)

7.168 W65C51.cs

[Go to the documentation of this file.](#)

```

00001 using System;
00002 using System.ComponentModel;
00003 using System.IO;
00004 using System.IO.Ports;
00005
00006 namespace Hardware
00007 {
00008     /// <summary>
00009     /// An implementation of a W65C51 ACIA.
00010     /// </summary>
00011     [Serializable]
00012     public class W65C51
00013     {
00014         #region Fields
00015         public readonly int defaultBaudRate = 115200;
00016         public byte byteIn;
00017     #endregion
00018
00019     #region Properties
00020         public byte[] Memory { get; set; }
00021         public bool IsEnabled { get; set; }
00022         public SerialPort Object { get; set; }
00023         public string ObjectName { get; set; }
00024         private W65C02 Processor { get; set; }
00025         private BackgroundWorker _backgroundWorker { get; set; }
00026         public int Offset { get; set; }
00027         public int Length { get; set; }
00028
00029         private bool DataRead { get; set; }
00030         private bool EchoMode { get; set; }
00031         private bool InterruptDisabled { get; set; }
00032         private bool Interrupted { get; set; }
00033         private bool Overrun { get; set; }
00034         private bool ParityEnabled { get; set; }
00035         private bool ReceiverFull { get; set; }
00036         private byte RtsControl { get; set; }
00037     #endregion
00038 }

```

```

00039 #region Public Methods
00040     public W65C51(W65C02 processor, byte offset)
00041     {
00042         if (offset > MemoryMap.DeviceArea.Length)
00043             throw new ArgumentException(String.Format("The offset: {0} is greater than the device
area: {1}", offset, MemoryMap.DeviceArea.Length));
00044
00045         Processor = processor;
00046
00047         Offset = MemoryMap.DeviceArea.Offset | offset;
00048         Length = 0x04;
00049         Memory = new byte[Length + 1];
00050
00051         _backgroundWorker = new BackgroundWorker
00052         {
00053             WorkerSupportsCancellation = true
00054         };
00055         _backgroundWorker.DoWork += BackgroundWorkerDoWork;
00056         _backgroundWorker.RunWorkerAsync();
00057     }
00058
00059     public void Reset()
00060     {
00061         IsEnabled = false;
00062     }
00063
00064     /// <summary>
00065     /// Default Constructor, Instantiates a new instance of COM Port I/O.
00066     /// </summary>
00067     ///
00068     /// <param name="port"> COM Port to use for I/O</param>
00069     public void Init(string port)
00070     {
00071         Object = new SerialPort(port, defaultBaudRate, Parity.None, 8, StopBits.One);
00072         ObjectName = port;
00073
00074         ComInit(Object);
00075     }
00076
00077     /// <summary>
00078     /// Default Constructor, Instantiates a new instance of COM Port I/O.
00079     /// </summary>
00080     ///
00081     /// <param name="port">COM Port to use for I/O</param>
00082     /// <param name="baudRate">Baud Rate to use for I/O</param>
00083     public void Init(string port, int baudRate)
00084     {
00085         Object = new SerialPort(port, baudRate, Parity.None, 8, StopBits.One);
00086         ObjectName = port;
00087
00088         ComInit(Object);
00089     }
00090
00091     /// <summary>
00092     /// Called when the window is closed.
00093     /// </summary>
00094     public void Fini()
00095     {
00096         ComFini(Object);
00097     }
00098
00099     /// <summary>
00100     /// Returns the byte at a given address.
00101     /// </summary>
00102     ///
00103     /// <param name="address"></param>
00104     ///
00105     /// <returns>the byte being returned</returns>
00106     public byte Read(int address)
00107     {
00108         HardwarePreRead(address);
00109         byte data = Memory[address - Offset];
00110         DataRead = true;
00111         return data;
00112     }
00113
00114     /// <summary>
00115     /// Writes data to the given address.
00116     /// </summary>
00117     ///
00118     /// <param name="address">The address to write data to</param>
00119     /// <param name="data">The data to write</param>
00120     public void Write(int address, byte data)
00121     {
00122         HardwarePreWrite(address, data);
00123         if (!(address == Offset) || (address == Offset + 1))
00124     {

```

```

00125         Memory[address - Offset] = data;
00126     }
00127 }
00128
00129 /// <summary>
00130 /// Called in order to write to the serial port.
00131 /// </summary>
00132 ///
00133 /// <param name="data">Byte of data to send</param>
00134 public void WriteCOM(byte data)
00135 {
00136     byte[] writeByte = new byte[] { data };
00137     Object.Write(writeByte, 0, 1);
00138 }
00139 #endregion
00140
00141 #region Private Methods
00142 /// <summary>
00143 /// Called whenever the ACIA is initialized.
00144 /// </summary>
00145 ///
00146 /// <param name="serialPort">SerialPort object to initialize.</param>
00147 private void ComInit(SerialPort serialPort)
00148 {
00149     try
00150     {
00151         serialPort.Open();
00152     }
00153     catch (UnauthorizedAccessException w)
00154     {
00155         FileStream file = new FileStream(FileLocations.ErrorFile, FileMode.OpenOrCreate,
00156 FileAccess.ReadWrite);
00157         StreamWriter stream = new StreamWriter(file);
00158         stream.WriteLine(w.Message);
00159         stream.WriteLine(w.Source);
00160         stream.Flush();
00161         file.Flush();
00162         stream.Close();
00163         file.Close();
00164         return;
00165     }
00166     serialPort.ReadTimeout = 50;
00167     serialPort.WriteTimeout = 50;
00168     serialPort.DataReceived += new SerialDataReceivedEventHandler(SerialDataReceived);
00169     try
00170     {
00171         serialPort.Write("-----\r\n");
00172         serialPort.Write(" WolfNet 6502 WBC Emulator\r\n");
00173         serialPort.Write("-----\r\n");
00174         serialPort.Write("\r\n");
00175     }
00176     catch (TimeoutException t)
00177     {
00178         _ = t;
00179         FileStream file = new FileStream(FileLocations.ErrorFile, FileMode.OpenOrCreate,
00180 FileAccess.ReadWrite);
00181         StreamWriter stream = new StreamWriter(file);
00182         stream.WriteLine("Read/Write error: Port timed out!");
00183         stream.WriteLine("Please ensure all cables are connected properly!");
00184         stream.Flush();
00185         file.Flush();
00186         stream.Close();
00187         file.Close();
00188         return;
00189     }
00190 }
00191 /// <summary>
00192 /// Called when the window is closed.
00193 /// </summary>
00194 ///
00195 /// <param name="serialPort">SerialPort Object to close</param>
00196 private void ComFini(SerialPort serialPort)
00197 {
00198     if (serialPort != null)
00199     {
00200         serialPort.Close();
00201     }
00202     _backgroundWorker.CancelAsync();
00203     _backgroundWorker.DoWork -= BackgroundWorkerDoWork;
00204 }
00205
00206 /// <summary>
00207 /// Called whenever SerialDataReceivedEventHandler event occurs.
00208 /// </summary>
00209 ///

```

```

00210 /// <param name="sender"></param>
00211 /// <param name="e"></param>
00212 private void SerialDataReceived(object sender, SerialDataReceivedEventArgs e)
00213 {
00214     try
00215     {
00216         if (EchoMode)
00217         {
00218             WriteCOM(Convert.ToByte(Object.ReadByte()));
00219         }
00220         else
00221         {
00222             if (!ReceiverFull)
00223             {
00224                 ReceiverFull = true;
00225             }
00226             else
00227             {
00228                 Overrun = true;
00229             }
00230             Memory[0] = Convert.ToByte(Object.ReadByte());
00231         }
00232     }
00233     if (!InterruptDisabled)
00234     {
00235         Interrupted = true;
00236         Processor.InterruptRequest();
00237     }
00238 }
00239 catch (Win32Exception w)
00240 {
00241     FileStream file = new FileStream(FileLocations.ErrorFile, FileMode.OpenOrCreate,
00242     FileAccess.ReadWrite);
00243     StreamWriter stream = new StreamWriter(file);
00244     stream.WriteLine(w.Message);
00245     stream.WriteLine(w.ErrorCode.ToString());
00246     stream.WriteLine(w.Source);
00247     stream.Flush();
00248     stream.Close();
00249     file.Flush();
00250     file.Close();
00251 }
00252 }
00253 private void HardwarePreWrite(int address, byte data)
00254 {
00255     if (address == Offset)
00256     {
00257         WriteCOM(data);
00258     }
00259     else if (address == Offset + 1)
00260     {
00261         Reset();
00262     }
00263     else if (address == Offset + 2)
00264     {
00265         CommandRegister(data);
00266     }
00267     else if (address == Offset + 3)
00268     {
00269         ControlRegister(data);
00270     }
00271 }
00272 }
00273 private void HardwarePreRead(int address)
00274 {
00275     if (address == Offset)
00276     {
00277         Interrupted = false;
00278         Overrun = false;
00279         ReceiverFull = false;
00280     }
00281     else if (address == Offset + 1)
00282     {
00283         StatusRegisterUpdate();
00284     }
00285     else if (address == Offset + 2)
00286     {
00287         CommandRegisterUpdate();
00288     }
00289     else if (address == Offset + 3)
00290     {
00291         ControlRegisterUpdate();
00292     }
00293 }
00294 }
00295

```



```

00296     private void CommandRegister(byte data)
00297     {
00298         byte test = (byte)(data & 0x20);
00299         if (test == 0x20)
00300         {
00301             throw new ArgumentException("Parity must NEVER be enabled!");
00302         }
00303
00304         test = (byte)(data & 0x10);
00305         if (test == 0x10)
00306         {
00307             EchoMode = true;
00308         }
00309         else
00310         {
00311             EchoMode = false;
00312         }
00313
00314         test = (byte)(data & 0x0C);
00315         if (test == 0x00)
00316         {
00317             Object.Handshake = Handshake.None;
00318             Object.RtsEnable = true;
00319             Object.Handshake = Handshake.RequestToSend;
00320         }
00321         else if (test == 0x04)
00322         {
00323             Object.Handshake = Handshake.None;
00324             Object.RtsEnable = false;
00325         }
00326         else if ((test == 0x08) || (test == 0x0C))
00327         {
00328             throw new NotImplementedException("This cannot be emulated on windows!");
00329         }
00330         else
00331         {
00332             throw new ArgumentOutOfRangeException("RtsControl is invalid!");
00333         }
00334
00335         test = (byte)(data & 0x02);
00336         if (test == 0x02)
00337         {
00338             InterruptDisabled = true;
00339         }
00340         else
00341         {
00342             InterruptDisabled = false;
00343         }
00344
00345         test = (byte)(data & 0x01);
00346         if (test == 0x01)
00347         {
00348             Object.DtrEnable = true;
00349         }
00350         else
00351         {
00352             Object.DtrEnable = false;
00353         }
00354     }
00355
00356     private void CommandRegisterUpdate()
00357     {
00358         byte data = Memory[Offset + 2];
00359
00360         if (ParityEnabled)
00361         {
00362             data |= 0x20;
00363         }
00364         else
00365         {
00366             data &= 0xD0;
00367         }
00368
00369         if (EchoMode)
00370         {
00371             data |= 0x10;
00372         }
00373         else
00374         {
00375             data &= 0xE0;
00376         }
00377
00378         data &= RtsControl;
00379
00380         if (InterruptDisabled)
00381         {
00382             data |= 0x02;

```

```

00383     }
00384     else
00385     {
00386         data &= 0x0D;
00387     }
00388     if (Object.DtrEnable)
00389     {
00390         data |= 0x01;
00391     }
00392     else
00393     {
00394         data &= 0x0E;
00395     }
00396
00397     Memory[Offset + 2] = data;
00398 }
00399
00400 private void ControlRegister(byte data)
00401 {
00402     byte test = (byte)(data & 0x80);
00403     if (test == 0x80)
00404     {
00405         test = (byte)(data & 0x60);
00406         if (test == 0x60)
00407         {
00408             Object.StopBits = StopBits.OnePointFive;
00409         }
00410         else
00411         {
00412             Object.StopBits = StopBits.Two;
00413         }
00414     }
00415     else
00416     {
00417         Object.StopBits = StopBits.One;
00418     }
00419
00420     test = (byte)(data & 0x60);
00421     if (test == 0x20)
00422     {
00423         Object.DataBits = 7;
00424     }
00425     else if (test == 0x40)
00426     {
00427         Object.DataBits = 6;
00428     }
00429     else if (test == 0x60)
00430     {
00431         Object.DataBits = 5;
00432     }
00433     else
00434     {
00435         Object.DataBits = 8;
00436     }
00437
00438     test = (byte)(data & 0x10);
00439     if (!(test == 0x10))
00440     {
00441         throw new ArgumentException("External clock rate not available on the WolfNet 65C02
WBC!");
00442     }
00443
00444     test = (byte)(data & 0x0F);
00445     if (test == 0x00)
00446     {
00447         Object.BaudRate = 115200;
00448     }
00449     else if (test == 0x01)
00450     {
00451         Object.BaudRate = 50;
00452     }
00453     else if (test == 0x02)
00454     {
00455         Object.BaudRate = 75;
00456     }
00457     else if (test == 0x03)
00458     {
00459         Object.BaudRate = 110;
00460     }
00461     else if (test == 0x04)
00462     {
00463         Object.BaudRate = 135;
00464     }
00465     else if (test == 0x05)
00466     {
00467         Object.BaudRate = 150;
00468     }

```

```

00469         else if (test == 0x06)
00470         {
00471             Object.BaudRate = 300;
00472         }
00473         else if (test == 0x07)
00474         {
00475             Object.BaudRate = 600;
00476         }
00477         else if (test == 0x08)
00478         {
00479             Object.BaudRate = 1200;
00480         }
00481         else if (test == 0x09)
00482         {
00483             Object.BaudRate = 1800;
00484         }
00485         else if (test == 0x0A)
00486         {
00487             Object.BaudRate = 2400;
00488         }
00489         else if (test == 0x0B)
00490         {
00491             Object.BaudRate = 3600;
00492         }
00493         else if (test == 0x0C)
00494         {
00495             Object.BaudRate = 4800;
00496         }
00497         else if (test == 0x0D)
00498         {
00499             Object.BaudRate = 7200;
00500         }
00501         else if (test == 0x0E)
00502         {
00503             Object.BaudRate = 9600;
00504         }
00505         else
00506         {
00507             Object.BaudRate = 19200;
00508         }
00509     }
00510
00511     private void ControlRegisterUpdate()
00512     {
00513         byte controlRegister = Memory[Offset + 3];
00514
00515         if (Object.StopBits == StopBits.Two)
00516         {
00517             controlRegister |= 0x80;
00518         }
00519         else if ((Object.StopBits == StopBits.OnePointFive) && (Object.DataBits == 5) ||
00520 (Object.StopBits == StopBits.One))
00521         {
00522             controlRegister &= 0x7F;
00523         }
00524         else
00525         {
00526             throw new ArgumentOutOfRangeException("StopBits or combination of StopBits and
DataBits is invalid!");
00527         }
00528
00529         if (Object.DataBits == 8)
00530         {
00531             controlRegister &= 0x9F;
00532         }
00533         else if (Object.DataBits == 7)
00534         {
00535             controlRegister |= 0x20;
00536         }
00537         else if (Object.DataBits == 6)
00538         {
00539             controlRegister |= 0x40;
00540         }
00541         else if (Object.DataBits == 5)
00542         {
00543             controlRegister |= 0x60;
00544         }
00545         else
00546         {
00547             throw new ArgumentOutOfRangeException("DataBits is out of range!");
00548         }
00549
00550         if (Object.BaudRate == 115200)
00551         {
00552             controlRegister &= 0xF0;
00553         }
00554         else if (Object.BaudRate == 50)

```

```

00554         {
00555             controlRegister |= 0x01;
00556         }
00557         else if (Object.BaudRate == 75)
00558         {
00559             controlRegister |= 0x02;
00560         }
00561         else if (Object.BaudRate == 110)
00562         {
00563             controlRegister |= 0x03;
00564         }
00565         else if (Object.BaudRate == 135)
00566         {
00567             controlRegister |= 0x04;
00568         }
00569         else if (Object.BaudRate == 150)
00570         {
00571             controlRegister |= 0x05;
00572         }
00573         else if (Object.BaudRate == 300)
00574         {
00575             controlRegister |= 0x06;
00576         }
00577         else if (Object.BaudRate == 600)
00578         {
00579             controlRegister |= 0x07;
00580         }
00581         else if (Object.BaudRate == 1200)
00582         {
00583             controlRegister |= 0x08;
00584         }
00585         else if (Object.BaudRate == 1800)
00586         {
00587             controlRegister |= 0x09;
00588         }
00589         else if (Object.BaudRate == 2400)
00590         {
00591             controlRegister |= 0x0A;
00592         }
00593         else if (Object.BaudRate == 3600)
00594         {
00595             controlRegister |= 0x0B;
00596         }
00597         else if (Object.BaudRate == 4800)
00598         {
00599             controlRegister |= 0x0C;
00600         }
00601         else if (Object.BaudRate == 7200)
00602         {
00603             controlRegister |= 0x0D;
00604         }
00605         else if (Object.BaudRate == 9600)
00606         {
00607             controlRegister |= 0x0E;
00608         }
00609         else if (Object.BaudRate == 19200)
00610         {
00611             controlRegister |= 0x0F;
00612         }
00613         else
00614         {
00615             throw new ArgumentOutOfRangeException("BaudRate is outside the range of Baud Rates
supported by the W65C51!");
00616         }
00617
00618         Memory[Offset + 3] = controlRegister;
00619     }
00620
00621     private void StatusRegisterUpdate()
00622     {
00623         byte statusRegister = Memory[Offset + 1];
00624
00625         if (Interrupted)
00626         {
00627             statusRegister |= 0x80;
00628         }
00629         else
00630         {
00631             statusRegister &= 0x7F;
00632         }
00633
00634         if (Object.DsrHolding == false)
00635         {
00636             statusRegister |= 0x40;
00637         }
00638         else
00639         {

```

```

00640         statusRegister &= 0xBF;
00641     }
00642
00643     if (Object.CDHolding)
00644     {
00645         statusRegister |= 0x20;
00646     }
00647     else
00648     {
00649         statusRegister &= 0xDF;
00650     }
00651
00652     statusRegister |= 0x10;
00653
00654     if (ReceiverFull)
00655     {
00656         statusRegister |= 0x08;
00657     }
00658     else
00659     {
00660         statusRegister &= 0xF7;
00661     }
00662
00663     if (Overflow)
00664     {
00665         statusRegister |= 0x04;
00666     }
00667     else
00668     {
00669         statusRegister &= 0xFB;
00670     }
00671
00672     statusRegister &= 0xFC;
00673
00674     Memory[Offset + 1] = statusRegister;
00675 }
00676
00677 private void BackgroundWorkerDoWork(object sender, DoWorkEventArgs e)
00678 {
00679     var worker = sender as BackgroundWorker;
00680
00681     while (true)
00682     {
00683         if (worker != null && worker.CancellationPending)
00684         {
00685             e.Cancel = true;
00686             return;
00687         }
00688
00689         if (Processor.isRunning)
00690         {
00691             if (ReceiverFull || Overflow)
00692             {
00693                 Memory[Offset + 1] = (byte)(Memory[Offset + 1] | 0x80);
00694                 Interrupted = true;
00695                 Processor.InterruptRequest();
00696             }
00697
00698             if (DataRead)
00699             {
00700                 ReceiverFull = false;
00701                 Interrupted = false;
00702                 Overflow = false;
00703                 DataRead = false;
00704             }
00705         }
00706     }
00707 }
00708 #endregion
00709 }
00710 }

```

Index

- Length
 - Hardware.MemoryMap.BankedRam, [21](#)
 - Hardware.MemoryMap.BankedRom, [23](#)
 - Hardware.MemoryMap.DeviceArea, [26](#)
 - Hardware.MemoryMap.SharedRom, [134](#)
 - Offset
 - Hardware.MemoryMap.BankedRam, [21](#)
 - Hardware.MemoryMap.BankedRom, [23](#)
 - Hardware.MemoryMap.DeviceArea, [26](#)
 - Hardware.MemoryMap.SharedRom, [134](#)
 - PortList
 - Emulator.ViewModel.SettingsViewModel, [132](#)
 - backgroundWorker
 - Emulator.ViewModel.MainViewModel, [58](#)
 - Hardware.W65C51, [208](#)
 - breakpointTriggered
 - Emulator.ViewModel.MainViewModel, [58](#)
 - contentLoaded
 - Emulator.MainWindow, [81](#)
 - Emulator.MemoryVisual, [94](#)
 - Emulator.SaveFile, [115](#)
 - Emulator.Settings, [126](#)
 - Emulator.Window1, [212](#)
 - XamlGeneratedNamespace.GeneratedApplication, [36](#)
 - cycleCount
 - Hardware.W65C02, [181](#)
 - interrupt
 - Hardware.W65C02, [181](#)
 - logger
 - Hardware.W65C02, [181](#)
 - memoryPageOffset
 - Emulator.ViewModel.MemoryVisualViewModel, [97](#)
 - previousInterrupt
 - Hardware.W65C02, [182](#)
 - programCounter
 - Hardware.W65C02, [182](#)
 - stackPointer
 - Hardware.W65C02, [182](#)
 - stateFileModel
 - Emulator.ViewModel.SaveFileViewModel, [117](#)
- About
 - Emulator.ViewModel.MainViewModel, [50](#)
- AboutCommand
 - Emulator.ViewModel.MainViewModel, [59](#)
- Accumulator
 - Emulator.Model.OutputLog, [102](#)
 - Hardware.W65C02, [182](#)
- ACIA
 - Hardware.MemoryMap, [86](#)
- ACR
 - Hardware.W65C22, [191](#)
- ACR_T1TC
 - Hardware.W65C22, [191](#)
- ACR_T2TC
 - Hardware.W65C22, [191](#)
- AddBreakPoint
 - Emulator.ViewModel.MainViewModel, [50](#)
- AddBreakPointCommand
 - Emulator.ViewModel.MainViewModel, [59](#)
- AddEventHandler
 - XamlGeneratedNamespace.GeneratedInternalTypeHelper, [37](#)
- AddressingMode
 - Hardware, [12](#)
- AddWithCarryOperation
 - Hardware.W65C02, [147](#)
- AllTypes
 - Emulator.Model.Breakpoint, [24](#)
 - Emulator.Model.BreakpointType, [25](#)
- AndOperation
 - Hardware.W65C02, [148](#)
- Apply
 - Emulator.ViewModel.SettingsViewModel, [132](#)
- ApplyCommand
 - Emulator.ViewModel.SettingsViewModel, [133](#)
- ApplyEnabled
 - Emulator.ViewModel.SettingsViewModel, [133](#)
- AslOperation
 - Hardware.W65C02, [148](#)
- AT28C010
 - Emulator.Model.StateFileModel, [136](#)
 - Emulator.ViewModel.MainViewModel, [59](#)
- AT28C64
 - Emulator.Model.StateFileModel, [136](#)
 - Emulator.ViewModel.MainViewModel, [59](#)
- AT28CXX
 - Hardware.AT28CXX, [15](#)
- BackgroundWorkerDoWork
 - Emulator.ViewModel.MainViewModel, [51](#)
 - Hardware.W65C51, [197](#)
- BankedRAM
 - Hardware.MemoryMap, [86](#)
- BankedROM
 - Hardware.MemoryMap, [86](#)
- Banks
 - Hardware.AT28CXX, [19](#)
 - Hardware.HM62256, [45](#)
- BankSize
 - Hardware.MemoryMap.BankedRam, [21](#)
- BinaryLoadedNotification
 - Emulator.ViewModel.MainViewModel, [51](#)
- BIOS_LOADPROGRAM_ERROR
 - Emulator.ExitCodes, [29](#)
- BitOperation
 - Hardware.W65C02, [149](#)
- BranchOperation
 - Hardware.W65C02, [150](#)

- BreakOperation
 - Hardware.W65C02, [150](#)
- Breakpoints
 - Emulator.ViewModel.MainViewModel, [59](#)
- Build
 - Emulator.Versioning.Product, [104](#)
 - Emulator.Versioning.SettingsFile, [127](#)
- byteIn
 - Hardware.W65C51, [207](#)
- CarryFlag
 - Hardware.W65C02, [182](#)
- ChangeMemoryByOne
 - Hardware.W65C02, [150](#)
- ChangeRegisterByOne
 - Hardware.W65C02, [151](#)
- Cleanup
 - Emulator.ViewModel.ViewModelLocator, [143](#)
- Clear
 - Hardware.AT28CXX, [15](#)
 - Hardware.HM62256, [43](#)
- Close
 - Emulator.IClosable, [46](#)
 - Emulator.ViewModel.MainViewModel, [51](#)
 - Emulator.ViewModel.SaveFileViewModel, [117](#)
 - Emulator.ViewModel.SettingsViewModel, [132](#)
- CloseCommand
 - Emulator.ViewModel.MainViewModel, [59](#)
 - Emulator.ViewModel.SaveFileViewModel, [117](#)
 - Emulator.ViewModel.SettingsViewModel, [133](#)
- CloseFile
 - Emulator.MainWindow, [66](#)
- CollectionChanged
 - Emulator.MultiThreadedObservableCollection< T>, [101](#)
- ComFini
 - Hardware.W65C51, [197](#)
- ComInit
 - Hardware.W65C51, [198](#)
- CommandRegister
 - Hardware.W65C51, [199](#)
- CommandRegisterUpdate
 - Hardware.W65C51, [199](#)
- Company
 - Emulator.Versioning.Product, [104](#)
 - Hardware.Versioning.Product, [106](#)
- CompareOperation
 - Hardware.W65C02, [151](#)
- ComPortName
 - Emulator.Model.SettingsModel, [128](#)
- ComPortSelection
 - Emulator.ViewModel.SettingsViewModel, [133](#)
- Connect
 - Emulator.MainWindow, [66](#), [68](#), [70](#), [72](#), [74](#), [76](#)
 - Emulator.MemoryVisual, [92](#), [93](#)
 - Emulator.SaveFile, [110–112](#)
 - Emulator.Settings, [120–123](#)
 - Emulator.Window1, [211](#)
- ControlRegister
 - Hardware.W65C51, [200](#)
- ControlRegisterUpdate
 - Hardware.W65C51, [201](#)
- ConvertFlagsToByte
 - Hardware.W65C02, [153](#)
- ConvertOpCodeIntoString
 - Hardware.Utility, [137](#)
- Copyright
 - Emulator.Versioning.Product, [104](#)
 - Hardware.Versioning.Product, [106](#)
- CpuSpeed
 - Emulator.ViewModel.MainViewModel, [60](#)
- CreateDelegate
 - XamlGeneratedNamespace.GeneratedInternalTypeHelper, [38](#)
- CreateInstance
 - XamlGeneratedNamespace.GeneratedInternalTypeHelper, [39](#)
- CreateNew
 - Emulator.SettingsFile, [126](#)
- CurrentBank
 - Hardware.AT28CXX, [19](#)
 - Hardware.HM62256, [45](#)
- CurrentDisassembly
 - Emulator.ViewModel.MainViewModel, [60](#)
 - Hardware.W65C02, [183](#)
- CurrentOpCode
 - Emulator.Model.OutputLog, [103](#)
 - Hardware.W65C02, [183](#)
- CurrentSerialPort
 - Emulator.ViewModel.MainViewModel, [60](#)
- CycleCountIncrementedAction
 - Hardware.W65C02, [183](#)
- DataRead
 - Hardware.W65C51, [208](#)
- DecimalFlag
 - Hardware.W65C02, [183](#)
- defaultBaudRate
 - Hardware.W65C51, [207](#)
- Description
 - Emulator.Versioning.Product, [105](#)
 - Hardware.Versioning.Product, [106](#)
- DisableInterruptFlag
 - Hardware.W65C02, [183](#)
- DisassemblyOutput
 - Hardware.Disassembly, [28](#)
- DumpMemory
 - Hardware.AT28CXX, [16](#)
 - Hardware.HM62256, [44](#)
- EchoMode
 - Hardware.W65C51, [208](#)
- Emulator, [10](#)
- Emulator.App, [14](#)
- Emulator.ExitCodes, [29](#)
 - BIOS_LOADPROGRAM_ERROR, [29](#)
 - LOAD_BIOS_FILE_ERROR, [29](#)
 - LOAD_ROM_FILE_ERROR, [30](#)

- LOAD_STATE_ERROR, 30
- NO_BIOS, 30
- NO_ERROR, 30
- ROM_LOADPROGRAM_ERROR, 30
- USER_ERROR, 30
- Emulator.IClosable, 46
 - Close, 46
- Emulator.MainWindow, 64
 - _contentLoaded, 81
 - CloseFile, 66
 - Connect, 66, 68, 70, 72, 74, 76
 - InitializeComponent, 78, 79
 - LoadFile, 80
 - MainWindow, 65
 - NotificationMessageReceived, 80
 - SaveFile, 80
 - ToClose, 81
- Emulator.MemoryVisual, 91
 - _contentLoaded, 94
 - Connect, 92, 93
 - InitializeComponent, 93, 94
 - MemoryVisual, 92
- Emulator.Model, 11
- Emulator.Model.Breakpoint, 23
 - AllTypes, 24
 - IsEnabled, 24
 - Type, 24
 - Value, 24
- Emulator.Model.BreakpointType, 25
 - AllTypes, 25
 - NumberOfCycleType, 25
 - ProgramCounterType, 26
- Emulator.Model.MemoryRowModel, 87
 - Location00, 88
 - Location01, 88
 - Location02, 88
 - Location03, 88
 - Location04, 88
 - Location05, 88
 - Location06, 89
 - Location07, 89
 - Location08, 89
 - Location09, 89
 - Location0A, 89
 - Location0B, 89
 - Location0C, 90
 - Location0D, 90
 - Location0E, 90
 - Location0F, 90
 - Offset, 90
- Emulator.Model.OutputLog, 101
 - Accumulator, 102
 - CurrentOpCode, 103
 - NumberOfCycles, 103
 - OutputLog, 102
 - ProgramCounter, 103
 - StackPointer, 103
 - XRegister, 103
 - YRegister, 103
- Emulator.Model.RomFileModel, 107
 - Rom, 107
 - RomBanks, 107
 - RomBankSize, 108
 - RomFileName, 108
 - RomFilePath, 108
- Emulator.Model.SettingsModel, 128
 - ComPortName, 128
 - SettingsVersionBuild, 128
 - SettingsVersionMajor, 129
 - SettingsVersionMinor, 129
 - SettingsVersionRevision, 129
- Emulator.Model.StateFileModel, 135
 - AT28C010, 136
 - AT28C64, 136
 - MM65SIB, 136
 - NumberOfCycles, 136
 - OutputLog, 136
 - W65C02, 136
 - W65C22, 137
 - W65C51, 137
- Emulator.MultiThreadedObservableCollection< T >, 99
 - CollectionChanged, 101
 - MultiThreadedObservableCollection, 100
 - OnCollectionChanged, 101
- Emulator.SaveFile, 109
 - _contentLoaded, 115
 - Connect, 110–112
 - InitializeComponent, 112–114
 - NotificationMessageReceived, 114
 - SaveFile, 110
- Emulator.Settings, 119
 - _contentLoaded, 126
 - Connect, 120–123
 - InitializeComponent, 123–125
 - NotificationMessageReceived, 125
 - PortSelectionDropDownClosed, 125
 - Settings, 120
- Emulator.SettingsFile, 126
 - CreateNew, 126
- Emulator.Versioning, 142
- Emulator.Versioning.Product, 104
 - Build, 104
 - Company, 104
 - Copyright, 104
 - Description, 105
 - Major, 105
 - Minor, 105
 - Name, 105
 - Revision, 105
 - Title, 105
 - VersionString, 105
- Emulator.Versioning.SettingsFile, 127
 - Build, 127
 - Major, 127
 - Minor, 127
 - Revision, 128

- Emulator.ViewModel, 11
- Emulator.ViewModel.MainViewModel, 47
 - _backgroundWorker, 58
 - _breakpointTriggered, 58
 - About, 50
 - AboutCommand, 59
 - AddBreakPoint, 50
 - AddBreakPointCommand, 59
 - AT28C010, 59
 - AT28C64, 59
 - BackgroundWorkerDoWork, 51
 - BinaryLoadedNotification, 51
 - Breakpoints, 59
 - Close, 51
 - CloseCommand, 59
 - CpuSpeed, 60
 - CurrentDisassembly, 60
 - CurrentSerialPort, 60
 - GenericNotification, 52
 - GetLogModValue, 53
 - GetOutputLog, 53
 - GetSleepValue, 54
 - HM62256, 60
 - IsBreakPointTriggered, 54
 - IsRomLoaded, 60
 - IsRunning, 61
 - MainViewModel, 49
 - MemoryPage, 61
 - MemoryView, 54
 - MemoryVisualCommand, 61
 - MM65SIB, 61
 - NumberOfCycles, 61
 - OnClose, 55
 - OnLoad, 55
 - OutputLog, 62
 - RemoveBreakPoint, 55
 - RemoveBreakPointCommand, 62
 - Reset, 56
 - ResetCommand, 62
 - RomFile, 62
 - RunPause, 56
 - RunPauseCommand, 62
 - SelectedBreakpoint, 62
 - Settings, 56
 - SettingsAppliedNotification, 57
 - SettingsCommand, 63
 - SettingsModel, 63
 - StateLoadedNotification, 57
 - Step, 57
 - StepCommand, 63
 - StepProcessor, 58
 - UpdateMemoryMapCommand, 63
 - UpdateMemoryPage, 58
 - UpdateUi, 58
 - W65C02, 63
 - W65C22, 63
 - W65C51, 64
 - WindowTitle, 64
- Emulator.ViewModel.MemoryVisualViewModel, 95
 - _memoryPageOffset, 97
 - GenericNotification, 96
 - MemoryPage, 97
 - MemoryPageOffset, 97
 - MemoryVisualViewModel, 96
 - UpdateMemoryMapCommand, 98
 - UpdateMemoryPage, 96
 - UpdateUi, 97
- Emulator.ViewModel.SaveFileViewModel, 115
 - _stateFileModel, 117
 - Close, 117
 - CloseCommand, 117
 - Filename, 118
 - Save, 117
 - SaveEnabled, 118
 - SaveFileCommand, 118
 - SaveFileViewModel, 116
 - Select, 117
 - SelectFileCommand, 118
- Emulator.ViewModel.SettingsViewModel, 129
 - _PortList, 132
 - Apply, 132
 - ApplyCommand, 133
 - ApplyEnabled, 133
 - Close, 132
 - CloseCommand, 133
 - ComPortSelection, 133
 - PortList, 133
 - SettingsModel, 133
 - SettingsViewModel, 130, 131
 - UpdatePortList, 132
- Emulator.ViewModel.ViewModelLocator, 142
 - Cleanup, 143
 - Main, 143
 - MemoryVisual, 143
 - Settings, 144
 - ViewModelLocator, 143
- Emulator.Window1, 210
 - _contentLoaded, 212
 - Connect, 211
 - InitializeComponent, 211
- Emulator/App.xaml.cs, 212
- Emulator/Classes/ExitCodes.cs, 213
- Emulator/Classes/FileLocations.cs, 213
- Emulator/Classes/SettingsFile.cs, 214
- Emulator/Classes/Versioning.cs, 214, 215
- Emulator/Interfaces/IClosable.cs, 216
- Emulator/MainWindow.xaml.cs, 216
- Emulator/MemoryVisual.xaml.cs, 217
- Emulator/Model/Breakpoint.cs, 218
- Emulator/Model/BreakpointType.cs, 218, 219
- Emulator/Model/MemoryRowModel.cs, 219
- Emulator/Model/OutputLog.cs, 220, 221
- Emulator/Model/RomFileModel.cs, 221, 222
- Emulator/Model/SettingsModel.cs, 222
- Emulator/Model/StateFileModel.cs, 223
- Emulator/MultiThreadedCollection.cs, 224

- Emulator/obj/x86/Debug/.NETFramework,Version=v4.8.AssemblyAttributes.cs, 225
- Emulator/obj/x86/Debug/App.g.cs, 226, 227
- Emulator/obj/x86/Debug/App.g.i.cs, 230, 231
- Emulator/obj/x86/Debug/Emulator_Content.g.cs, 234
- Emulator/obj/x86/Debug/Emulator_Content.g.i.cs, 235
- Emulator/obj/x86/Debug/GeneratedInternalTypeHelper.g.cs, 236
- Emulator/obj/x86/Debug/GeneratedInternalTypeHelper.g.i.cs, 237
- Emulator/obj/x86/Debug/MainWindow.g.cs, 240
- Emulator/obj/x86/Debug/MainWindow.g.i.cs, 261
- Emulator/obj/x86/Debug/MemoryVisual.g.cs, 282
- Emulator/obj/x86/Debug/MemoryVisual.g.i.cs, 285, 286
- Emulator/obj/x86/Debug/SaveFile.g.cs, 289
- Emulator/obj/x86/Debug/SaveFile.g.i.cs, 295
- Emulator/obj/x86/Debug/Settings.g.cs, 301
- Emulator/obj/x86/Debug/Settings.g.i.cs, 307
- Emulator/obj/x86/Publish/.NETFramework,Version=v4.8.AssemblyAttributes.cs, 225
- Emulator/obj/x86/Publish/App.g.cs, 228
- Emulator/obj/x86/Publish/App.g.i.cs, 232
- Emulator/obj/x86/Publish/Emulator_Content.g.cs, 235
- Emulator/obj/x86/Publish/Emulator_Content.g.i.cs, 236
- Emulator/obj/x86/Publish/GeneratedInternalTypeHelper.g.cs, 236
- Emulator/obj/x86/Publish/GeneratedInternalTypeHelper.g.i.cs, 238
- Emulator/obj/x86/Publish/MainWindow.g.cs, 247
- Emulator/obj/x86/Publish/MainWindow.g.i.cs, 268
- Emulator/obj/x86/Publish/SaveFile.g.cs, 291
- Emulator/obj/x86/Publish/SaveFile.g.i.cs, 297
- Emulator/obj/x86/Publish/Settings.g.cs, 303
- Emulator/obj/x86/Publish/Settings.g.i.cs, 308, 309
- Emulator/obj/x86/Release/.NETFramework,Version=v4.8.AssemblyAttributes.cs, 226
- Emulator/obj/x86/Release/App.g.cs, 229
- Emulator/obj/x86/Release/App.g.i.cs, 233
- Emulator/obj/x86/Release/Emulator_Content.g.cs, 235
- Emulator/obj/x86/Release/Emulator_Content.g.i.cs, 236
- Emulator/obj/x86/Release/GeneratedInternalTypeHelper.g.cs, 236
- Emulator/obj/x86/Release/GeneratedInternalTypeHelper.g.i.cs, 239
- Emulator/obj/x86/Release/MainWindow.g.cs, 254
- Emulator/obj/x86/Release/MainWindow.g.i.cs, 275
- Emulator/obj/x86/Release/MemoryMap.g.i.cs, 312, 313
- Emulator/obj/x86/Release/MemoryVisual.g.cs, 284
- Emulator/obj/x86/Release/MemoryVisual.g.i.cs, 287
- Emulator/obj/x86/Release/SaveFile.g.cs, 293
- Emulator/obj/x86/Release/SaveFile.g.i.cs, 299
- Emulator/obj/x86/Release/Settings.g.cs, 305
- Emulator/obj/x86/Release/Settings.g.i.cs, 310, 311
- Emulator/obj/x86/Release/Window1.g.i.cs, 314
- Emulator/Properties/AssemblyInfo.cs, 315
- Emulator/SaveFile.xaml.cs, 316, 317
- Emulator/Settings.xaml.cs, 317
- Emulator/ViewModel/MainViewModel.cs, 318, 319
- Emulator/ViewModel/MemoryVisualViewModel.cs, 328
- Emulator/ViewModel/SaveFileViewModel.cs, 329, 330
- Emulator/ViewModel/SettingsViewModel.cs, 331
- Emulator/ViewModel/ViewModelLocator.cs, 332, 333
- End
 - Hardware.AT28CXX, 20
 - Hardware.HM62256, 45
 - Hardware.MemoryMap.DeviceArea, 27
 - Hardware.W65C22, 193
- EorOperation
 - Hardware.W65C02, 153
- ExecuteOpCode
 - Hardware.W65C02, 154
- Filename
 - Emulator.ViewModel.SaveFileViewModel, 118
- Fin
 - Hardware.W65C51, 203
- GenericNotification
 - Emulator.ViewModel.MainViewModel, 52
- GenericNotification
 - Emulator.ViewModel.MemoryVisualViewModel, 96
- GetAddressByAddressingMode
 - Hardware.W65C02, 166
- GetAddressingMode
 - Hardware.W65C02, 168
- GetCycleCount
 - Hardware.W65C02, 171
- GetLogModValue
 - Emulator.ViewModel.MainViewModel, 53
- GetOutputLog
 - Emulator.ViewModel.MainViewModel, 53
- GetPropertyValue
 - XamlGeneratedNamespace.GeneratedInternalTypeHelper, 40
- GetSleepValue
 - Emulator.ViewModel.MainViewModel, 54
- GPIO
 - Hardware.MemoryMap, 86
- Hardware, 11
 - AddressingMode, 12
- Hardware.AT28CXX, 14
 - AT28CXX, 15
 - Banks, 19
 - Clear, 15
 - CurrentBank, 19
 - DumpMemory, 16
 - End, 20
 - Length, 20
 - Load, 16, 18
 - Memory, 20
 - Offset, 20
 - Processor, 20
 - Read, 18
 - ReadFile, 18
 - Write, 19
- Hardware.Disassembly, 28

- DisassemblyOutput, 28
- HighAddress, 28
- LowAddress, 28
- OpCodeString, 29
- Hardware.HM62256, 42
 - Banks, 45
 - Clear, 43
 - CurrentBank, 45
 - DumpMemory, 44
 - End, 45
 - HM62256, 43
 - Length, 45
 - Memory, 46
 - Offset, 46
 - Read, 44
 - Reset, 44
 - Write, 45
- Hardware.MemoryMap, 81
 - ACIA, 86
 - BankedRAM, 86
 - BankedROM, 86
 - GPIO, 86
 - Init, 82
 - Length, 86
 - MM65SIB, 86
 - Processor, 86
 - Read, 82
 - ReadWithoutCycle, 84
 - SharedROM, 87
 - Write, 84
 - WriteWithoutCycle, 85
- Hardware.MemoryMap.BankedRam, 21
 - _Length, 21
 - _Offset, 21
 - BankSize, 21
 - Length, 22
 - Offset, 22
 - TotalBanks, 21
 - TotalLength, 22
- Hardware.MemoryMap.BankedRom, 22
 - _Length, 23
 - _Offset, 23
 - Length, 23
 - Offset, 23
 - TotalBanks, 23
- Hardware.MemoryMap.DeviceArea, 26
 - _Length, 26
 - _Offset, 26
 - End, 27
 - Length, 27
 - Offset, 27
- Hardware.MemoryMap.Devices, 27
- Hardware.MemoryMap.Devices.ACIA, 13
 - Length, 13
 - Offset, 13
- Hardware.MemoryMap.Devices.GPIO, 42
 - Length, 42
 - Offset, 42
- Hardware.MemoryMap.Devices.MM65SIB, 98
 - Length, 98
 - Offset, 98
- Hardware.MemoryMap.SharedRom, 134
 - _Length, 134
 - _Offset, 134
 - Length, 135
 - Offset, 135
 - TotalBanks, 134
- Hardware.Utility, 137
 - ConvertOpCodeIntoString, 137
- Hardware.Versioning.Product, 106
 - Company, 106
 - Copyright, 106
 - Description, 106
 - Name, 106
 - Title, 106
 - Version, 107
- Hardware.W65C02, 144
 - _cycleCount, 181
 - _interrupt, 181
 - _logger, 181
 - _previousInterrupt, 182
 - _programCounter, 182
 - _stackPointer, 182
 - Accumulator, 182
 - AddWithCarryOperation, 147
 - AndOperation, 148
 - AslOperation, 148
 - BitOperation, 149
 - BranchOperation, 150
 - BreakOperation, 150
 - CarryFlag, 182
 - ChangeMemoryByOne, 150
 - ChangeRegisterByOne, 151
 - CompareOperation, 151
 - ConvertFlagsToByte, 153
 - CurrentDisassembly, 183
 - CurrentOpCode, 183
 - CycleCountIncrementedAction, 183
 - DecimalFlag, 183
 - DisableInterruptFlag, 183
 - EorOperation, 153
 - ExecuteOpCode, 154
 - GetAddressByAddressingMode, 166
 - GetAddressingMode, 168
 - GetCycleCount, 171
 - IncrementCycleCount, 171
 - InterruptRequest, 171
 - isRunning, 182
 - JumpToSubRoutineOperation, 171
 - LsrOperation, 172
 - MoveProgramCounterByRelativeValue, 172
 - NegativeFlag, 183
 - NextStep, 173
 - OrOperation, 173
 - OverflowFlag, 184
 - PeekStack, 174

- PokeStack, [174](#)
- ProcessIRQ, [174](#)
- ProcessNMI, [174](#)
- ProgramCounter, [184](#)
- PullFlagsOperation, [175](#)
- PushFlagsOperation, [175](#)
- Reset, [175](#)
- ResetCycleCount, [175](#)
- ReturnFromInterruptOperation, [176](#)
- ReturnFromSubRoutineOperation, [176](#)
- RolOperation, [176](#)
- RorOperation, [177](#)
- SetDisassembly, [178](#)
- SetNegativeFlag, [180](#)
- SetZeroFlag, [180](#)
- StackPointer, [184](#)
- SubtractWithBorrowOperation, [180](#)
- TriggerIRQ, [184](#)
- TriggerNmi, [184](#)
- W65C02, [147](#)
- WrapProgramCounter, [181](#)
- XRegister, [185](#)
- YRegister, [185](#)
- ZeroFlag, [185](#)
- Hardware.W65C22, [185](#)
 - ACR, [191](#)
 - ACR_T1TC, [191](#)
 - ACR_T2TC, [191](#)
 - End, [193](#)
 - IER, [191](#)
 - IER_EN, [191](#)
 - IER_T1, [191](#)
 - IER_T2, [191](#)
 - IFR, [192](#)
 - IFR_INT, [192](#)
 - IFR_T1, [192](#)
 - IFR_T2, [192](#)
 - Init, [187](#)
 - Length, [193](#)
 - Memory, [193](#)
 - Offset, [193](#)
 - OnT1Timeout, [188](#)
 - OnT2Timeout, [188](#)
 - Processor, [194](#)
 - Read, [189](#)
 - Reset, [189](#)
 - T1CH, [192](#)
 - T1CL, [192](#)
 - T1Init, [189](#)
 - T1Interval, [194](#)
 - T1IsEnabled, [194](#)
 - T1IsIRQ, [192](#)
 - T1Object, [194](#)
 - T1TimerControl, [194](#)
 - T2CH, [192](#)
 - T2CL, [193](#)
 - T2Init, [190](#)
 - T2Interval, [194](#)
 - T2IsEnabled, [195](#)
 - T2IsIRQ, [193](#)
 - T2Object, [195](#)
 - T2TimerControl, [195](#)
 - W65C22, [187](#)
 - Write, [190](#)
- Hardware.W65C51, [195](#)
 - _backgroundWorker, [208](#)
 - BackgroundWorkerDoWork, [197](#)
 - byteIn, [207](#)
 - ComFini, [197](#)
 - ComInit, [198](#)
 - CommandRegister, [199](#)
 - CommandRegisterUpdate, [199](#)
 - ControlRegister, [200](#)
 - ControlRegisterUpdate, [201](#)
 - DataRead, [208](#)
 - defaultBaudRate, [207](#)
 - EchoMode, [208](#)
 - Fini, [203](#)
 - HardwarePreRead, [203](#)
 - HardwarePreWrite, [203](#)
 - Init, [204](#)
 - InterruptDisabled, [208](#)
 - Interrupted, [208](#)
 - IsEnabled, [208](#)
 - Length, [208](#)
 - Memory, [209](#)
 - Object, [209](#)
 - ObjectName, [209](#)
 - Offset, [209](#)
 - Overrun, [209](#)
 - ParityEnabled, [209](#)
 - Processor, [209](#)
 - Read, [205](#)
 - ReceiverFull, [210](#)
 - Reset, [205](#)
 - RtsControl, [210](#)
 - SerialDataReceived, [205](#)
 - StatusRegisterUpdate, [206](#)
 - W65C51, [197](#)
 - Write, [207](#)
 - WriteCOM, [207](#)
- Hardware/Classes/AddressingMode.cs, [334](#)
- Hardware/Classes/Disassembly.cs, [335](#)
- Hardware/Classes/FileLocations.cs, [213](#), [214](#)
- Hardware/Classes/MemoryMap.cs, [336](#)
- Hardware/Classes/Utility.cs, [339](#)
- Hardware/Classes/Versioning.cs, [215](#)
- Hardware/Hardware/AT28CXX.cs, [343](#)
- Hardware/Hardware/HM62256.cs, [345](#)
- Hardware/Hardware/W65C02.cs, [347](#)
- Hardware/Hardware/W65C22.cs, [376](#)
- Hardware/Hardware/W65C51.cs, [380](#)
- Hardware/obj/Debug/.NETFramework,Version=v4.8.AssemblyAttributes.cs, [226](#)
- Hardware/obj/Publish/.NETFramework,Version=v4.8.AssemblyAttributes.cs, [226](#)

- Hardware/obj/Release/.NETFramework,Version=v4.8.AssemblyAttributes.cs, [226](#)
- Hardware/Properties/AssemblyInfo.cs, [316](#)
- HardwarePreRead
 - Hardware.W65C51, [203](#)
- HardwarePreWrite
 - Hardware.W65C51, [203](#)
- HighAddress
 - Hardware.Disassembly, [28](#)
- HM62256
 - Emulator.ViewModel.MainViewModel, [60](#)
 - Hardware.HM62256, [43](#)
- IER
 - Hardware.W65C22, [191](#)
- IER_EN
 - Hardware.W65C22, [191](#)
- IER_T1
 - Hardware.W65C22, [191](#)
- IER_T2
 - Hardware.W65C22, [191](#)
- IFR
 - Hardware.W65C22, [192](#)
- IFR_INT
 - Hardware.W65C22, [192](#)
- IFR_T1
 - Hardware.W65C22, [192](#)
- IFR_T2
 - Hardware.W65C22, [192](#)
- IncrementCycleCount
 - Hardware.W65C02, [171](#)
- Init
 - Hardware.MemoryMap, [82](#)
 - Hardware.W65C22, [187](#)
 - Hardware.W65C51, [204](#)
- InitializeComponent
 - Emulator.MainWindow, [78](#), [79](#)
 - Emulator.MemoryVisual, [93](#), [94](#)
 - Emulator.SaveFile, [112–114](#)
 - Emulator.Settings, [123–125](#)
 - Emulator.Window1, [211](#)
 - XamlGeneratedNamespace.GeneratedApplication, [32–34](#)
- InterruptDisabled
 - Hardware.W65C51, [208](#)
- Interrupted
 - Hardware.W65C51, [208](#)
- InterruptRequest
 - Hardware.W65C02, [171](#)
- IsBreakPointTriggered
 - Emulator.ViewModel.MainViewModel, [54](#)
- IsEnabled
 - Emulator.Model.Breakpoint, [24](#)
 - Hardware.W65C51, [208](#)
- IsRomLoaded
 - Emulator.ViewModel.MainViewModel, [60](#)
- IsRunning
 - Emulator.ViewModel.MainViewModel, [61](#)
- isRunning
- Hardware.W65C02, [182](#)
- JumpToSubRoutineOperation
 - Hardware.W65C02, [171](#)
- Length
 - Hardware.AT28CXX, [20](#)
 - Hardware.HM62256, [45](#)
 - Hardware.MemoryMap, [86](#)
 - Hardware.MemoryMap.BankedRam, [22](#)
 - Hardware.MemoryMap.BankedRom, [23](#)
 - Hardware.MemoryMap.DeviceArea, [27](#)
 - Hardware.MemoryMap.Devices.ACIA, [13](#)
 - Hardware.MemoryMap.Devices.GPIO, [42](#)
 - Hardware.MemoryMap.Devices.MM65SIB, [98](#)
 - Hardware.MemoryMap.SharedRom, [135](#)
 - Hardware.W65C22, [193](#)
 - Hardware.W65C51, [208](#)
- Load
 - Hardware.AT28CXX, [16](#), [18](#)
- LOAD_BIOS_FILE_ERROR
 - Emulator.ExitCodes, [29](#)
- LOAD_ROM_FILE_ERROR
 - Emulator.ExitCodes, [30](#)
- LOAD_STATE_ERROR
 - Emulator.ExitCodes, [30](#)
- LoadFile
 - Emulator.MainWindow, [80](#)
- Location00
 - Emulator.Model.MemoryRowModel, [88](#)
- Location01
 - Emulator.Model.MemoryRowModel, [88](#)
- Location02
 - Emulator.Model.MemoryRowModel, [88](#)
- Location03
 - Emulator.Model.MemoryRowModel, [88](#)
- Location04
 - Emulator.Model.MemoryRowModel, [88](#)
- Location05
 - Emulator.Model.MemoryRowModel, [88](#)
- Location06
 - Emulator.Model.MemoryRowModel, [89](#)
- Location07
 - Emulator.Model.MemoryRowModel, [89](#)
- Location08
 - Emulator.Model.MemoryRowModel, [89](#)
- Location09
 - Emulator.Model.MemoryRowModel, [89](#)
- Location0A
 - Emulator.Model.MemoryRowModel, [89](#)
- Location0B
 - Emulator.Model.MemoryRowModel, [89](#)
- Location0C
 - Emulator.Model.MemoryRowModel, [90](#)
- Location0D
 - Emulator.Model.MemoryRowModel, [90](#)
- Location0E
 - Emulator.Model.MemoryRowModel, [90](#)
- Location0F

- Emulator.Model.MemoryRowModel, 90
- LowAddress
 - Hardware.Disassembly, 28
- LsrOperation
 - Hardware.W65C02, 172
- Main
 - Emulator.ViewModel.ViewModelLocator, 143
 - XamlGeneratedNamespace.GeneratedApplication, 34, 35
- MainViewModel
 - Emulator.ViewModel.MainViewModel, 49
- MainViewModel.cs
 - W65C02, 318
 - W65C22, 318
 - W65C51, 318
- MainWindow
 - Emulator.MainWindow, 65
- Major
 - Emulator.Versioning.Product, 105
 - Emulator.Versioning.SettingsFile, 127
- Memory
 - Hardware.AT28CXX, 20
 - Hardware.HM62256, 46
 - Hardware.W65C22, 193
 - Hardware.W65C51, 209
- MemoryPage
 - Emulator.ViewModel.MainViewModel, 61
 - Emulator.ViewModel.MemoryVisualViewModel, 97
- MemoryPageOffset
 - Emulator.ViewModel.MemoryVisualViewModel, 97
- MemoryView
 - Emulator.ViewModel.MainViewModel, 54
- MemoryVisual
 - Emulator.MemoryVisual, 92
 - Emulator.ViewModel.ViewModelLocator, 143
- MemoryVisualCommand
 - Emulator.ViewModel.MainViewModel, 61
- MemoryVisualViewModel
 - Emulator.ViewModel.MemoryVisualViewModel, 96
- Minor
 - Emulator.Versioning.Product, 105
 - Emulator.Versioning.SettingsFile, 127
- MM65SIB
 - Emulator.Model.StateFileModel, 136
 - Emulator.ViewModel.MainViewModel, 61
 - Hardware.MemoryMap, 86
- MoveProgramCounterByRelativeValue
 - Hardware.W65C02, 172
- MultiThreadedObservableCollection
 - Emulator.MultiThreadedObservableCollection< T >, 100
- Name
 - Emulator.Versioning.Product, 105
 - Hardware.Versioning.Product, 106
- NegativeFlag
 - Hardware.W65C02, 183
- NextStep
 - Hardware.W65C02, 173
- NO_BIOS
 - Emulator.ExitCodes, 30
- NO_ERROR
 - Emulator.ExitCodes, 30
- NotificationMessageReceived
 - Emulator.MainWindow, 80
 - Emulator.SaveFile, 114
 - Emulator.Settings, 125
- NumberOfCycles
 - Emulator.Model.OutputLog, 103
 - Emulator.Model.StateFileModel, 136
 - Emulator.ViewModel.MainViewModel, 61
- NumberOfCycleType
 - Emulator.Model.BreakpointType, 25
- Object
 - Hardware.W65C51, 209
- ObjectName
 - Hardware.W65C51, 209
- Offset
 - Emulator.Model.MemoryRowModel, 90
 - Hardware.AT28CXX, 20
 - Hardware.HM62256, 46
 - Hardware.MemoryMap.BankedRam, 22
 - Hardware.MemoryMap.BankedRom, 23
 - Hardware.MemoryMap.DeviceArea, 27
 - Hardware.MemoryMap.Devices.ACIA, 13
 - Hardware.MemoryMap.Devices.GPIO, 42
 - Hardware.MemoryMap.Devices.MM65SIB, 98
 - Hardware.MemoryMap.SharedRom, 135
 - Hardware.W65C22, 193
 - Hardware.W65C51, 209
- OnClose
 - Emulator.ViewModel.MainViewModel, 55
- OnCollectionChanged
 - Emulator.MultiThreadedObservableCollection< T >, 101
- OnLoad
 - Emulator.ViewModel.MainViewModel, 55
- OnT1Timeout
 - Hardware.W65C22, 188
- OnT2Timeout
 - Hardware.W65C22, 188
- OpCodeString
 - Hardware.Disassembly, 29
- OrOperation
 - Hardware.W65C02, 173
- OutputLog
 - Emulator.Model.OutputLog, 102
 - Emulator.Model.StateFileModel, 136
 - Emulator.ViewModel.MainViewModel, 62
- OverflowFlag
 - Hardware.W65C02, 184
- Overrun
 - Hardware.W65C51, 209
- ParityEnabled
 - Hardware.W65C51, 209

- PeekStack
 - Hardware.W65C02, [174](#)
- PokeStack
 - Hardware.W65C02, [174](#)
- PortList
 - Emulator.ViewModel.SettingsViewModel, [133](#)
- PortSelectionDropDownClosed
 - Emulator.Settings, [125](#)
- ProcessIRQ
 - Hardware.W65C02, [174](#)
- ProcessNMI
 - Hardware.W65C02, [174](#)
- Processor
 - Hardware.AT28CXX, [20](#)
 - Hardware.MemoryMap, [86](#)
 - Hardware.W65C22, [194](#)
 - Hardware.W65C51, [209](#)
- ProgramCounter
 - Emulator.Model.OutputLog, [103](#)
 - Hardware.W65C02, [184](#)
- ProgramCounterType
 - Emulator.Model.BreakpointType, [26](#)
- PullFlagsOperation
 - Hardware.W65C02, [175](#)
- PushFlagsOperation
 - Hardware.W65C02, [175](#)
- Read
 - Hardware.AT28CXX, [18](#)
 - Hardware.HM62256, [44](#)
 - Hardware.MemoryMap, [82](#)
 - Hardware.W65C22, [189](#)
 - Hardware.W65C51, [205](#)
- ReadFile
 - Hardware.AT28CXX, [18](#)
- ReadWithoutCycle
 - Hardware.MemoryMap, [84](#)
- ReceiverFull
 - Hardware.W65C51, [210](#)
- RemoveBreakPoint
 - Emulator.ViewModel.MainViewModel, [55](#)
- RemoveBreakPointCommand
 - Emulator.ViewModel.MainViewModel, [62](#)
- Reset
 - Emulator.ViewModel.MainViewModel, [56](#)
 - Hardware.HM62256, [44](#)
 - Hardware.W65C02, [175](#)
 - Hardware.W65C22, [189](#)
 - Hardware.W65C51, [205](#)
- ResetCommand
 - Emulator.ViewModel.MainViewModel, [62](#)
- ResetCycleCount
 - Hardware.W65C02, [175](#)
- ReturnFromInterruptOperation
 - Hardware.W65C02, [176](#)
- ReturnFromSubRoutineOperation
 - Hardware.W65C02, [176](#)
- Revision
 - Emulator.Versioning.Product, [105](#)
- Emulator.Versioning.SettingsFile, [128](#)
- RolOperation
 - Hardware.W65C02, [176](#)
- Rom
 - Emulator.Model.RomFileModel, [107](#)
- ROM_LOADPROGRAM_ERROR
 - Emulator.ExitCodes, [30](#)
- RomBanks
 - Emulator.Model.RomFileModel, [107](#)
- RomBankSize
 - Emulator.Model.RomFileModel, [108](#)
- RomFile
 - Emulator.ViewModel.MainViewModel, [62](#)
- RomFileName
 - Emulator.Model.RomFileModel, [108](#)
- RomFilePath
 - Emulator.Model.RomFileModel, [108](#)
- RorOperation
 - Hardware.W65C02, [177](#)
- RtsControl
 - Hardware.W65C51, [210](#)
- RunPause
 - Emulator.ViewModel.MainViewModel, [56](#)
- RunPauseCommand
 - Emulator.ViewModel.MainViewModel, [62](#)
- Save
 - Emulator.ViewModel.SaveFileViewModel, [117](#)
- SaveEnabled
 - Emulator.ViewModel.SaveFileViewModel, [118](#)
- SaveFile
 - Emulator.MainWindow, [80](#)
 - Emulator.SaveFile, [110](#)
- SaveFileCommand
 - Emulator.ViewModel.SaveFileViewModel, [118](#)
- SaveFileViewModel
 - Emulator.ViewModel.SaveFileViewModel, [116](#)
- Select
 - Emulator.ViewModel.SaveFileViewModel, [117](#)
- SelectedBreakpoint
 - Emulator.ViewModel.MainViewModel, [62](#)
- SelectFileCommand
 - Emulator.ViewModel.SaveFileViewModel, [118](#)
- SerialDataReceived
 - Hardware.W65C51, [205](#)
- SetDisassembly
 - Hardware.W65C02, [178](#)
- SetNegativeFlag
 - Hardware.W65C02, [180](#)
- SetPropertyValue
 - XamlGeneratedNamespace.GeneratedInternalTypeHelper, [40, 41](#)
- Settings
 - Emulator.Settings, [120](#)
 - Emulator.ViewModel.MainViewModel, [56](#)
 - Emulator.ViewModel.ViewModelLocator, [144](#)
- SettingsAppliedNotifcation
 - Emulator.ViewModel.MainViewModel, [57](#)
- SettingsCommand

- Emulator.ViewModel.MainViewModel, 63
- SettingsModel
 - Emulator.ViewModel.MainViewModel, 63
 - Emulator.ViewModel.SettingsViewModel, 133
- SettingsVersionBuild
 - Emulator.Model.SettingsModel, 128
- SettingsVersionMajor
 - Emulator.Model.SettingsModel, 129
- SettingsVersionMinor
 - Emulator.Model.SettingsModel, 129
- SettingsVersionRevision
 - Emulator.Model.SettingsModel, 129
- SettingsViewModel
 - Emulator.ViewModel.SettingsViewModel, 130, 131
- SetZeroFlag
 - Hardware.W65C02, 180
- SharedROM
 - Hardware.MemoryMap, 87
- StackPointer
 - Emulator.Model.OutputLog, 103
 - Hardware.W65C02, 184
- StateLoadedNotification
 - Emulator.ViewModel.MainViewModel, 57
- StatusRegisterUpdate
 - Hardware.W65C51, 206
- Step
 - Emulator.ViewModel.MainViewModel, 57
- StepCommand
 - Emulator.ViewModel.MainViewModel, 63
- StepProcessor
 - Emulator.ViewModel.MainViewModel, 58
- SubtractWithBorrowOperation
 - Hardware.W65C02, 180
- T1CH
 - Hardware.W65C22, 192
- T1CL
 - Hardware.W65C22, 192
- T1Init
 - Hardware.W65C22, 189
- T1Interval
 - Hardware.W65C22, 194
- T1IsEnabled
 - Hardware.W65C22, 194
- T1IsIRQ
 - Hardware.W65C22, 192
- T1Object
 - Hardware.W65C22, 194
- T1TimerControl
 - Hardware.W65C22, 194
- T2CH
 - Hardware.W65C22, 192
- T2CL
 - Hardware.W65C22, 193
- T2Init
 - Hardware.W65C22, 190
- T2Interval
 - Hardware.W65C22, 194
- T2IsEnabled
 - Hardware.W65C22, 195
- T2IsIRQ
 - Hardware.W65C22, 193
- T2Object
 - Hardware.W65C22, 195
- T2TimerControl
 - Hardware.W65C22, 195
- Title
 - Emulator.Versioning.Product, 105
 - Hardware.Versioning.Product, 106
- ToClose
 - Emulator.MainWindow, 81
- TotalBanks
 - Hardware.MemoryMap.BankedRam, 21
 - Hardware.MemoryMap.BankedRom, 23
 - Hardware.MemoryMap.SharedRom, 134
- TotalLength
 - Hardware.MemoryMap.BankedRam, 22
- TriggerIRQ
 - Hardware.W65C02, 184
- TriggerNmi
 - Hardware.W65C02, 184
- Type
 - Emulator.Model.Breakpoint, 24
- UpdateMemoryMapCommand
 - Emulator.ViewModel.MainViewModel, 63
 - Emulator.ViewModel.MemoryVisualViewModel, 98
- UpdateMemoryPage
 - Emulator.ViewModel.MainViewModel, 58
 - Emulator.ViewModel.MemoryVisualViewModel, 96
- UpdatePortList
 - Emulator.ViewModel.SettingsViewModel, 132
- UpdateUi
 - Emulator.ViewModel.MainViewModel, 58
 - Emulator.ViewModel.MemoryVisualViewModel, 97
- USER_ERROR
 - Emulator.ExitCodes, 30
- Value
 - Emulator.Model.Breakpoint, 24
- Version
 - Hardware.Versioning.Product, 107
- VersionString
 - Emulator.Versioning.Product, 105
- ViewModelLocator
 - Emulator.ViewModel.ViewModelLocator, 143
- W65C02
 - Emulator.Model.StateFileModel, 136
 - Emulator.ViewModel.MainViewModel, 63
 - Hardware.W65C02, 147
 - MainViewModel.cs, 318
- W65C22
 - Emulator.Model.StateFileModel, 137
 - Emulator.ViewModel.MainViewModel, 63
 - Hardware.W65C22, 187
 - MainViewModel.cs, 318
- W65C51

- Emulator.Model.StateFileModel, [137](#)
- Emulator.ViewModel.MainViewModel, [64](#)
- Hardware.W65C51, [197](#)
- MainViewModel.cs, [318](#)
- WindowTitle
 - Emulator.ViewModel.MainViewModel, [64](#)
- WrapProgramCounter
 - Hardware.W65C02, [181](#)
- Write
 - Hardware.AT28CXX, [19](#)
 - Hardware.HM62256, [45](#)
 - Hardware.MemoryMap, [84](#)
 - Hardware.W65C22, [190](#)
 - Hardware.W65C51, [207](#)
- WriteCOM
 - Hardware.W65C51, [207](#)
- WriteWithoutCycle
 - Hardware.MemoryMap, [85](#)
- XamlGeneratedNamespace, [13](#)
- XamlGeneratedNamespace.GeneratedApplication, [31](#)
 - _contentLoaded, [36](#)
 - InitializeComponent, [32–34](#)
 - Main, [34, 35](#)
- XamlGeneratedNamespace.GeneratedInternalType-Helper, [36](#)
 - AddEventHandler, [37](#)
 - CreateDelegate, [38](#)
 - CreateInstance, [39](#)
 - GetPropertyValue, [40](#)
 - SetPropertyValue, [40, 41](#)
- XRegister
 - Emulator.Model.OutputLog, [103](#)
 - Hardware.W65C02, [185](#)
- YRegister
 - Emulator.Model.OutputLog, [103](#)
 - Hardware.W65C02, [185](#)
- ZeroFlag
 - Hardware.W65C02, [185](#)