WolfNet 6502 WorkBench Computer Emulator

beta

Generated by WolfNet Computing using Doxygen 1.9.5

1 Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

2 Hierarchical Index

2.1 Class Hierarchy

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

3 Class Index

3.1 Class List

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

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

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
      full value would then be added to the X Register.
00016 /// If the X register was 0x01 then the address would be 0xFF01.  and the value stored there would
      have an ADC operation performed on it and the value would
00017 /// be added to the accumulator.
00018 /// </summary>
00019         AbsoluteX = 2,
00020 /// <summary>
00021 /// In this mode a full address is given to operation on IE: Memory byte[] { 0x79, 0x00, 0xFF } The
      full value would then be added to the Y Register.
00022 /// If the Y register was 0x01 then the address would be 0xFF01.  and the value stored there would
      have an ADC operation performed on it and the value would
00023 /// be added to the accumulator
00024 /// </summary>
00025         AbsoluteY = 3,
00026 /// <summary>
00027 /// In this mode the instruction operates on the accumulator.  No operands are needed.
00028 /// </summary>
00029         Accumulator = 4,
00030 /// <summary>
00031 /// In this mode, the value to operate on immediately follows the instruction.  IE: Memory byte[] {
      0x69, 0x01 }
00032 /// would perform an ADC operation and Add 0x01 directly to the accumulator
00033 /// </summary>
00034         Immediate = 5,
00035 /// <summary>
00036 /// No address is needed for this mode.  EX: BRK (Break), CLC (Clear Carry Flag) etc
00037 /// </summary>
00038         Implied = 6,
00039 /// <summary>
00040 /// In this mode assume the following
00041 /// Memory = { 0x61, 0x02, 0x04, 0x00, 0x03 }
00042 /// RegisterX = 0x01
00043 /// 1.  Take the sum of the X Register and the value after the opcode 0x01 + 0x01 = 0x02.
00044 /// 2.  Starting at position 0x02 get an address (0x04,0x00) = 0x0004
00045 /// 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 addres 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

bank	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

bank	The bank to load data to.
data	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

data	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

bank	The bank to read data from.
address	

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

bank	The bank to load data to.
address	The address to write data to
data	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 ProgamCounter 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 ProgamCounter 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.

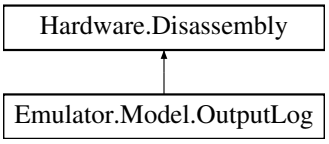
The documentation for this class was generated from the following file:

- 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.Initialize↩ Component ( ) [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 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         }
```

6.12.2.2 InitializeComponent() [2/6] void XamlGeneratedNamespace.GeneratedApplication.Initialize↩ Component ( ) [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 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         }
```

6.12.2.3 InitializeComponent() [3/6] void XamlGeneratedNamespace.GeneratedApplication.Initialize↩ Component ( ) [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 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         }
```

6.12.2.4 InitializeComponent() [4/6] void XamlGeneratedNamespace.GeneratedApplication.Initialize↩ Component ( ) [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 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         }
```

6.12.2.5 InitializeComponent() [5/6] void XamlGeneratedNamespace.GeneratedApplication.Initialize↩ Component ( ) [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 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         }
```

6.12.2.6 InitializeComponent() [6/6] void XamlGeneratedNamespace.GeneratedApplication.Initialize↩ Component ( ) [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 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         }
```

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.

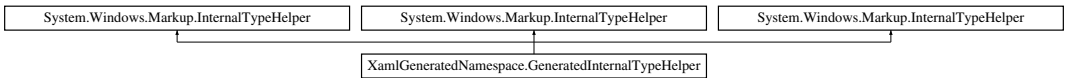
The documentation for this class was generated from the following files:

- 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.GeneratedInternalType↩ Helper.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.GeneratedInternalType↩ Helper.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.GeneratedInternalType↩ Helper.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.Generated↩ InternalTypeHelper.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",
      (System.Reflection.BindingFlags.InvokeMethod
00049                             | (System.Reflection.BindingFlags.NonPublic |
      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.Generated↩ InternalTypeHelper.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",
      (System.Reflection.BindingFlags.InvokeMethod
00049                             | (System.Reflection.BindingFlags.NonPublic |
      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",
      (System.Reflection.BindingFlags.InvokeMethod
00049                             | (System.Reflection.BindingFlags.NonPublic |
      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 |
      System.Reflection.BindingFlags.NonPublic)
00027                             | (System.Reflection.BindingFlags.Instance |
      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 |
      System.Reflection.BindingFlags.NonPublic)
00027                             | (System.Reflection.BindingFlags.Instance |
      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.GeneratedInternalType↩ Helper.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,
      culture);
00042         }
```

6.13.2.14 SetPropertyValue() [2/3] override void XamlGeneratedNamespace.GeneratedInternalType↩ Helper.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,
      culture);
00042         }
```

6.13.2.15 SetPropertyValue() [3/3] override void XamlGeneratedNamespace.GeneratedInternalType↩ Helper.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,
      culture);
00042         }
```

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

banks	Number of banks the new memory will have.
offset	Offset of the new memory in the address space.
length	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

bank	The bank to read data from.
address	

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

bank	The bank to load data to.
address	The address to write data to
data	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.

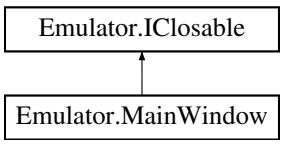
Definition at line 13 of file HM62256.cs.
```
00013 { get; set; }
```

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 ( )`

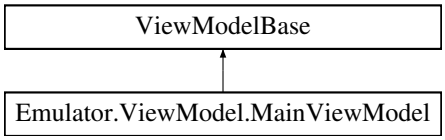
The documentation for this interface was generated from the following file:

- 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 Prorgam 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 progam 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 StateLoadedNotifcation (NotificationMessage< StateFileModel > notificationMessage)
- void GenericNotifcation (NotificationMessage notificationMessage)
- void SettingsAppliedNotifcation (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,
      MessageBoxImage.Warning,
00230                                     MessageBoxResult.OK);
00231                     // Close the file, then delete it.
00232                     _stream.Close();
00233                     File.Delete(FileLocations.SettingsFile);
00234                     SettingsModel = SettingsFile.CreateNew();
00235                 }
00236             }
00237             else
00238             {
00239                 MessageBox.Show("Creating new settings file...");
00240                 SettingsModel = SettingsFile.CreateNew();
00241             }
00242             _stream.Close();
00243
```

```
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, GenericNotifcation);
00276             Messenger.Default.Register<NotificationMessage<RomFileModel»(this,
      BinaryLoadedNotification);
00277             Messenger.Default.Register<NotificationMessage<SettingsModel»(this,
      SettingsAppliedNotifcation);
00278             Messenger.Default.Register<NotificationMessage<StateFileModel»(this,
      StateLoadedNotifcation);
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 GenericNotifcation() void Emulator.ViewModel.MainViewModel.GenericNotifcation (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
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         }
```

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,
      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.",
      Versioning.Product.Name),
00303                                                            Versioning.Product.Title,
      MessageBoxButton.OKCancel,
00304                                                            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         }
```

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,
      "SettingsWindow"));
00736         }
```

6.17.3.18 SettingsAppliedNotifcation() void Emulator.ViewModel.MainViewModel.SettingsApplied↩ Notifcation (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 StateLoadedNotifcation() void Emulator.ViewModel.MainViewModel.StateLoadedNotifcation (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
      MultiThreadedObservableCollection<OutputLog>(notificationMessage.Content.OutputLog);
00381             RaisePropertyChanged("OutputLog");
00382 
00383             NumberOfCycles = notificationMessage.Content.NumberOfCycles;
00384 
00385             W65C02 = notificationMessage.Content.W65C02;
00386             W65C22 = notificationMessage.Content.W65C22;
00387             MM65SIB = notificationMessage.Content.MM65SIB;
00388             W65C51 = notificationMessage.Content.W65C51;
00389             AT28C010 = notificationMessage.Content.AT28C010;
00390             AT28C64 = notificationMessage.Content.AT28C64;
00391             UpdateMemoryPage();
00392             UpdateUi();
00393 
00394             IsRomLoaded = true;
00395             RaisePropertyChanged("IsRomLoaded");
00396         }
```

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.AddBreakPoint↩ Command [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.Main↩ ViewModel.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 Prorgam 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 progam 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.UpdateMemoryMapCommand [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; } }
```

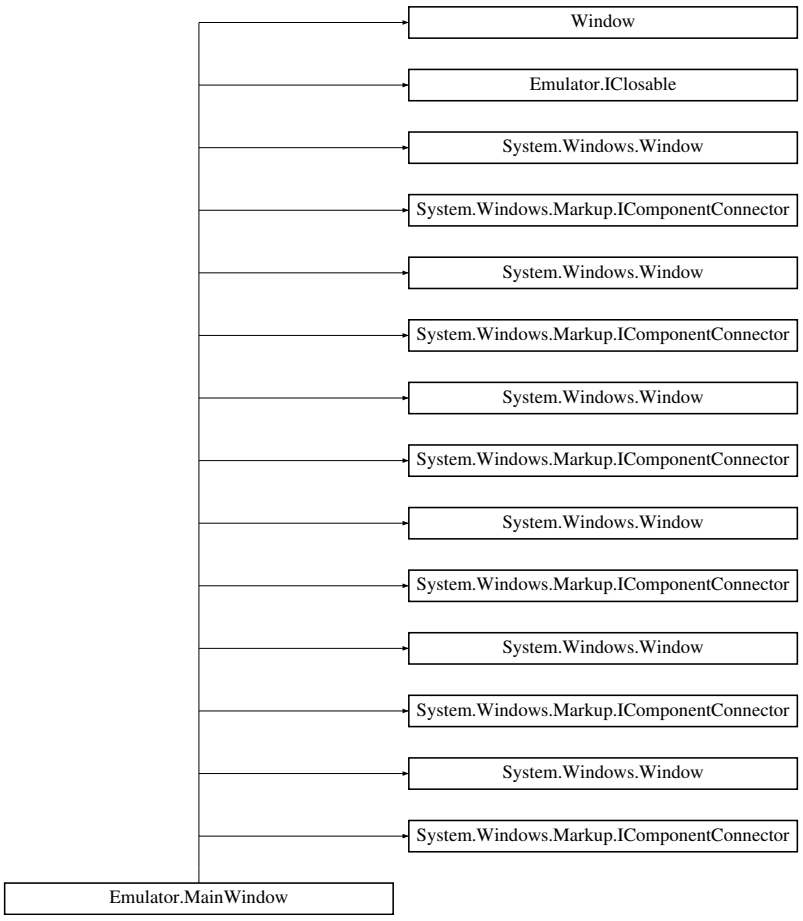
The documentation for this class was generated from the following file:

- 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 "..\..\..\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.InterrupFlag = ((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 "..\..\..\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.InterrupFlag = ((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.InterrupFlag = ((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 "..\..\..\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.InterrupFlag = ((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 "..\..\..\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.InterrupFlag = ((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 "..\..\..\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.InterrupFlag = ((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 "..\..\..\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 "..\..\..\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 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         }
```

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 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         }
```

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 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         }
```

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 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         }
```

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.NotificationMessageReceived (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.NotificationMessageReceived (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

address	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

address	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

address	The address to write data to.
data	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

address	The address to write data to.
data	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; }
```

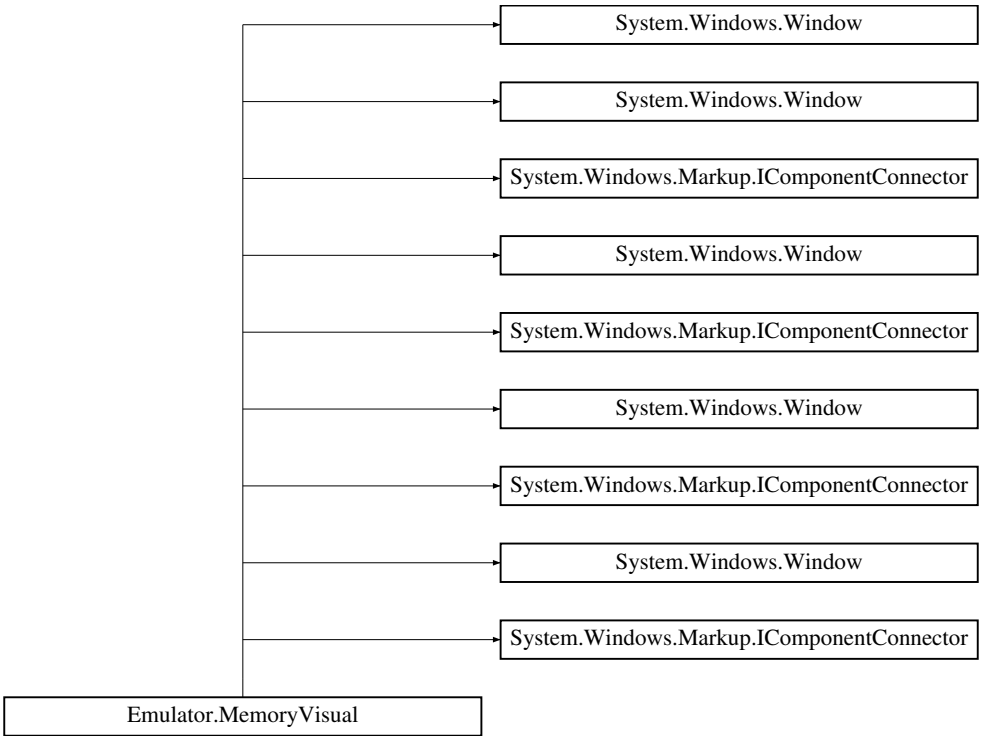
The documentation for this class was generated from the following file:

- 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.MemoryVisual.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.MemoryVisual.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.MemoryVisual.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.MemoryVisual.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 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         }
```

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 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         }
```

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 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         }
```

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 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         }
```

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.

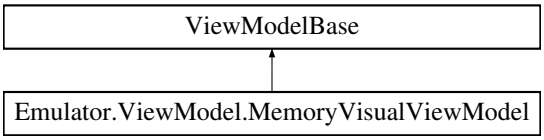
The documentation for this class was generated from the following files:

- 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 multiplyer = 0;
00080             for (ushort i = (ushort)offset; i < 256 * (_memoryPageOffset + 1); i++)
00081             {
00082                 MemoryPage.Add(new MemoryRowModel
00083                 {
00084                     Offset = ((16 * multiplyer) + 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                 multiplyer++;
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.

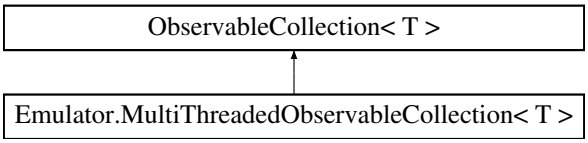
The documentation for this class was generated from the following file:

- Hardware/Classes/MemoryMap.cs

6.24 Emulator.MultiThreadedObservableCollection< T > Class Template Reference

A MultiThreaedObservableCollection. 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 MultiThreaedObservableCollection. This allows multiple threads to access the same observable collection in a safe manner.

Template Parameters

T	

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

collection	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

list	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.MultiThreadedObservableColle< T >.CollectionChanged

The NotifyCollectionChangedEventHandler, Sends a notification anytime the collection has been modified.

Definition at line 47 of file MultiThreadedCollection.cs.

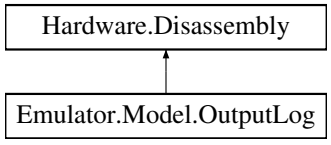
The documentation for this class was generated from the following file:

- 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 Accummulator
- 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 Accummulator

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 Ⓒ 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 l' 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 Ĩ 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 l' 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; }
```

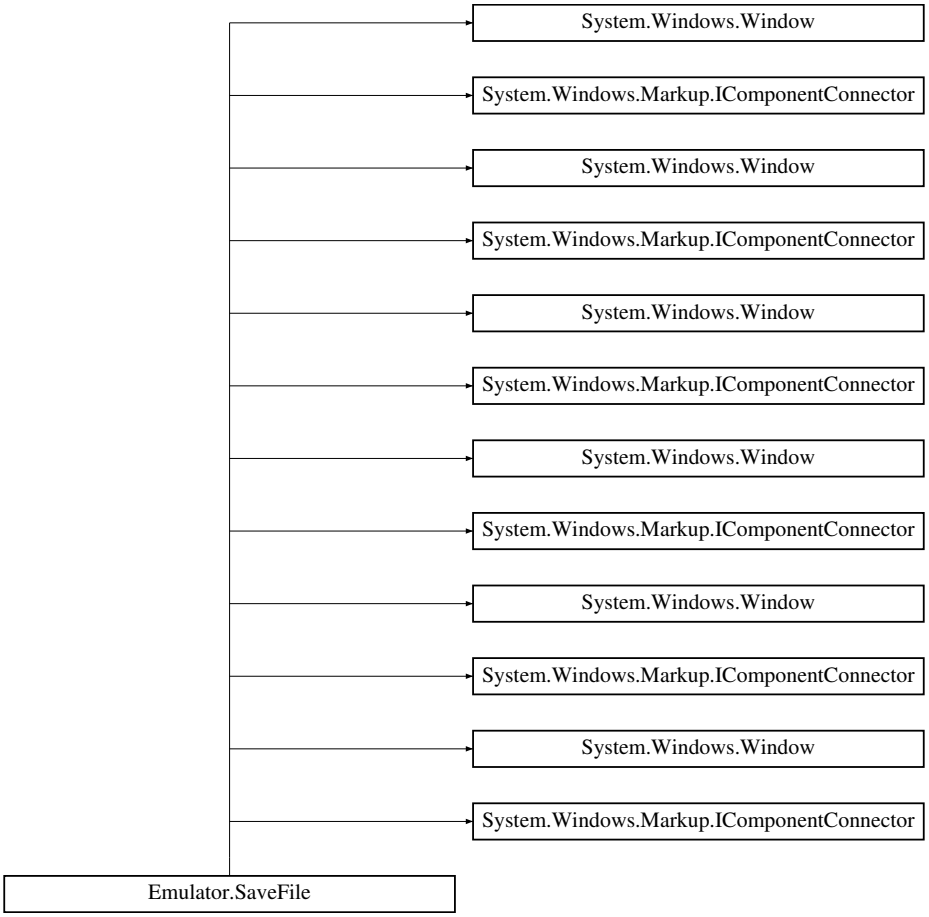
The documentation for this class was generated from the following file:

- 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 resourceLocater = new System.Uri("/Emulator;component/savefile.xaml",
      System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097             System.Windows.Application.LoadComponent(this, resourceLocater);
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 resourceLocater = new System.Uri("/Emulator;component/savefile.xaml",
      System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097             System.Windows.Application.LoadComponent(this, resourceLocater);
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 resourceLocater = new System.Uri("/Emulator;component/savefile.xaml",
      System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097             System.Windows.Application.LoadComponent(this, resourceLocater);
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 resourceLocater = new System.Uri("/Emulator;component/savefile.xaml",
      System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097             System.Windows.Application.LoadComponent(this, resourceLocater);
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 resourceLocater = new System.Uri("/Emulator;component/savefile.xaml",
      System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097             System.Windows.Application.LoadComponent(this, resourceLocater);
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 resourceLocater = new System.Uri("/Emulator;component/savefile.xaml",
      System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097             System.Windows.Application.LoadComponent(this, resourceLocater);
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.

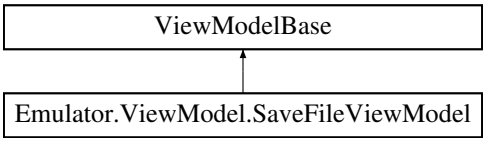
The documentation for this class was generated from the following files:

- 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

stateFileModel	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.SelectFileCommand [get], [set]`

The Relay Command called when Selecting a file

Definition at line 33 of file SaveFileViewModel.cs.
```
00033 { get; set; }
```

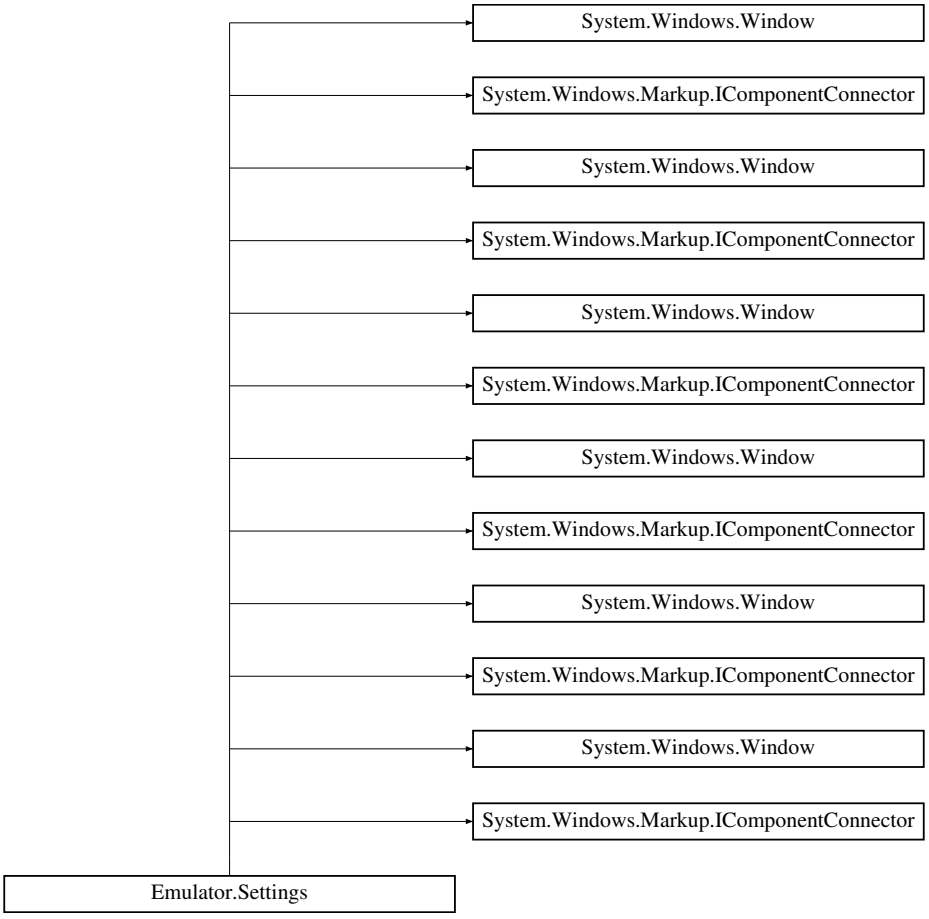
The documentation for this class was generated from the following file:

- 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,
      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 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         }
```

6.31.3.8 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 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         }
```

6.31.3.9 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 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         }
```

6.31.3.10 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 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         }
```

6.31.3.11 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; }
```

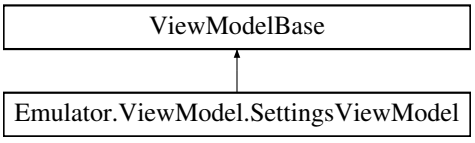
The documentation for this class was generated from the following file:

- 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

settingsModel	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:    //ăADCăImmediate
00012                 case 0x65:    //ăADCăZeroăPage
00013                 case 0x75:    //ăADCăZeroăPageăX
00014                 case 0x6D:  //ăADCăAbsolute
00015                 case 0x7D:  //ăADCăAbsoluteăX
00016                 case 0x79:    //ăADCăAbsoluteăY
00017                 case 0x61:    //ăADCăIndrectăX
00018                 case 0x71:    //ăADCăIndirectăY
00019                     {
00020                         return "ADC";
00021                     }
00022                 case 0x29:    //ăANDăImmediate
00023                 case 0x25:    //ăANDăZeroăPage
00024                 case 0x35:    //ăANDăZeroăPageăX
00025                 case 0x2D:  //ăANDăAbsolute
00026                 case 0x3D:  //ăANDăAbsoluteăX
00027                 case 0x39:    //ăANDăAbsoluteăY
00028                 case 0x21:    //ăANDăIndirectăX
00029                 case 0x31:    //ăANDăIndirectăY
00030                     {
00031                         return "AND";
00032                     }
00033                 case 0x0A:  //ăASLăAccumulator
00034                 case 0x06:    //ăASLăZeroăPage
00035                 case 0x16:    //ăASLăZeroăPageăX
00036                 case 0x0E:  //ăASLăAbsolute
00037                 case 0x1E:  //ăASLăAbsoluteăX
00038                     {
00039                         return "ASL";
00040                     }
00041                 case 0x90:    //ăBCCăRelative
00042                     {
00043                         return "BCC";
00044                     }
00045                 case 0xB0:    //ăBCSăRelative
00046                     {
00047                         return "BCS";
00048                     }
00049                 case 0xF0:    //ăBEQăRelative
00050                     {
00051                         return "BEQ";
00052                     }
00053                 case 0x24:    //ăBITăZeroăPage
00054                 case 0x2C:  //ăBITăAbsolute
00055                     {
00056                         return "BIT";
00057                     }
00058                 case 0x30:    //ăBMIăRelative
00059                     {
00060                         return "BMI";
00061                     }
00062                 case 0xD0:    //ăBNEăRelative
00063                     {
00064                         return "BNE";
00065                     }
00066                 case 0x10:    //ăBPLăRelative
00067                     {
00068                         return "BPL";
00069                     }
00070                 case 0x00:    //ăBRKăImplied
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:    //ăCLCăImplied
00083                     {
00084                         return "CLC";
00085                     }
00086                 case 0xD8:    //ăCLDăImplied
00087                     {
00088                         return "CLD";
00089                     }
```

```
00090                 case 0x58:    //ăCLIăImplied
00091                     {
00092                         return "CLI";
00093                     }
00094                 case 0xB8:    //ăCLVăImplied
00095                     {
00096                         return "CLV";
00097                     }
00098                 case 0xC9:    //ăCMPăImmediate
00099                 case 0xC5:    //ăCMPăZeroPage
00100                 case 0xD5:    //ăCMPăZeroăPageăX
00101                 case 0xCD:  //ăCMPăAbsolute
00102                 case 0xDD:  //ăCMPăAbsoluteăX
00103                 case 0xD9:    //ăCMPăAbsoluteăY
00104                 case 0xC1:    //ăCMPăIndirectăX
00105                 case 0xD1:    //ăCMPăIndirectăY
00106                     {
00107                         return "CMP";
00108                     }
00109                 case 0xE0:    //ăCPXăImmediate
00110                 case 0xE4:    //ăCPXăZeroPage
00111                 case 0xEC:  //ăCPXăAbsolute
00112                     {
00113                         return "CPX";
00114                     }
00115                 case 0xC0:    //ăCPYăImmediate
00116                 case 0xC4:    //ăCPYăZeroPage
00117                 case 0xCC:  //ăCPYăAbsolute
00118                     {
00119                         return "CPY";
00120                     }
00121                 case 0xC6:    //ăDECăZeroăPage
00122                 case 0xD6:    //ăDECăZeroăPageăX
00123                 case 0xCE:  //ăDECăAbsolute
00124                 case 0xDE:  //ăDECăAbsoluteăX
00125                     {
00126                         return "DEC";
00127                     }
00128                 case 0xCA:  //ăDEXăImplied
00129                     {
00130                         return "DEX";
00131                     }
00132                 case 0x88:    //ăDEYăImplied
00133                     {
00134                         return "DEY";
00135                     }
00136                 case 0x49:    //ăEORăImmediate
00137                 case 0x45:    //ăEORăZeroăPage
00138                 case 0x55:    //ăEORăZeroăPageăX
00139                 case 0x4D:  //ăEORăAbsolute
00140                 case 0x5D:  //ăEORăAbsoluteăX
00141                 case 0x59:    //ăEORăAbsoluteăY
00142                 case 0x41:    //ăEORăIndrectăX
00143                 case 0x51:    //ăEORăIndirectăY
00144                     {
00145                         return "EOR";
00146                     }
00147                 case 0xE6:    //ăINCăZeroăPage
00148                 case 0xF6:    //ăINCăZeroăPageăX
00149                     {
00150                         return "INC";
00151                     }
00152                 case 0xE8:    //ăINXăImplied
00153                     {
00154                         return "INX";
00155                     }
00156                 case 0xC8:    //ăINYăImplied
00157                     {
00158                         return "INY";
00159                     }
00160                 case 0xEE:  //ăINCăAbsolute
00161                 case 0xFE:  //ăINCăAbsoluteăX
00162                     {
00163                         return "INC";
00164                     }
00165                 case 0x4C:  //ăJMPăAbsolute
00166                 case 0x6C:  //ăJMPăIndirect
00167                     {
00168                         return "JMP";
00169                     }
00170                 case 0x20:    //ăJSRăAbsolute
00171                     {
00172                         return "JSR";
00173                     }
00174                 case 0xA9:    //ăLDAăImmediate
00175                 case 0xA5:    //ăLDAăZeroăPage
00176                 case 0xB5:    //ăLDAăZeroăPageăX
```

```
00177                 case 0xAD:  //ăLDAăAbsolute
00178                 case 0xBD:  //ăLDAăAbsoluteăX
00179                 case 0xB9:    //ăLDAăAbsoluteăY
00180                 case 0xA1:    //ăLDAăIndirectăX
00181                 case 0xB1:    //ăLDAăIndirectăY
00182                     {
00183                         return "LDA";
00184                     }
00185                 case 0xA2:    //ăLDXăImmediate
00186                 case 0xA6:    //ăLDXăZeroăPage
00187                 case 0xB6:    //ăLDXăZeroăPageăY
00188                 case 0xAE:  //ăLDXăAbsolute
00189                 case 0xBE:  //ăLDXăAbsoluteăY
00190                     {
00191                         return "LDX";
00192                     }
00193                 case 0xA0:    //ăLDYăImmediate
00194                 case 0xA4:    //ăLDYăZeroăPage
00195                 case 0xB4:    //ăLDYăZeroăPageăY
00196                 case 0xAC:  //ăLDYăAbsolute
00197                 case 0xBC:  //ăLDYăAbsoluteăY
00198                     {
00199                         return "LDY";
00200                     }
00201                 case 0x4A:  //ăLSRăAccumulator
00202                 case 0x46:    //ăLSRăZeroăPage
00203                 case 0x56:    //ăLSRăZeroăPageăX
00204                 case 0x4E:  //ăLSRăAbsolute
00205                 case 0x5E:  //ăLSRăAbsoluteăX
00206                     {
00207                         return "LSR";
00208                     }
00209                 case 0xEA:  //ăNOPăImplied
00210                     {
00211                         return "NOP";
00212                     }
00213                 case 0x09:    //ăORAăImmediate
00214                 case 0x05:    //ăORAăZeroăPage
00215                 case 0x15:    //ăORAăZeroăPageăX
00216                 case 0x0D:  //ăORAăAbsolute
00217                 case 0x1D:  //ăORAăAbsoluteăX
00218                 case 0x19:    //ăORAăAbsoluteăY
00219                 case 0x01:    //ăORAăIndirectăX
00220                 case 0x11:    //ăORAăIndirectăY
00221                     {
00222                         return "ORA";
00223                     }
00224                 case 0x48:    //ăPHAăImplied
00225                     {
00226                         return "PHA";
00227                     }
00228                 case 0x08:    //ăPHPăImplied
00229                     {
00230                         return "PHP";
00231                     }
00232                 case 0x68:    //ăPLAăImplied
00233                     {
00234                         return "PLA";
00235                     }
00236                 case 0x28:    //ăPLPăImplied
00237                     {
00238                         return "PLP";
00239                     }
00240                 case 0x2A:  //ăROLăAccumulator
00241                 case 0x26:    //ăROLăZeroăPage
00242                 case 0x36:    //ăROLăZeroăPageăX
00243                 case 0x2E:  //ăROLăAbsolute
00244                 case 0x3E:  //ăROLăAbsoluteăX
00245                     {
00246                         return "ROL";
00247                     }
00248                 case 0x6A:  //ăRORăAccumulator
00249                 case 0x66:    //ăRORăZeroăPage
00250                 case 0x76:    //ăRORăZeroăPageăX
00251                 case 0x6E:  //ăRORăAbsolute
00252                 case 0x7E:  //ăRORăAbsoluteăX
00253                     {
00254                         return "ROR";
00255                     }
00256                 case 0x40:    //ăRTIăImplied
00257                     {
00258                         return "RTI";
00259                     }
00260                 case 0x60:    //ăRTSăImplied
00261                     {
00262                         return "RTS";
00263                     }
```

```
00264                 case 0xE9:    //ăSBCăImmediate
00265                 case 0xE5:    //ăSBCăZeroăPage
00266                 case 0xF5:    //ăSBCăZeroăPageăX
00267                 case 0xED:  //ăSBCăAbsolute
00268                 case 0xFD:  //ăSBCăAbsoluteăX
00269                 case 0xF9:    //ăSBCăAbsoluteăY
00270                 case 0xE1:    //ăSBCăIndrectăX
00271                 case 0xF1:    //ăSBCăIndirectăY
00272                     {
00273                         return "SBC";
00274                     }
00275                 case 0x38:    //ăSECăImplied
00276                     {
00277                         return "SEC";
00278                     }
00279                 case 0xF8:    //ăSEDăImplied
00280                     {
00281                         return "SED";
00282                     }
00283                 case 0x78:    //ăSEIăImplied
00284                     {
00285                         return "SEI";
00286                     }
00287                 case 0x85:    //ăSTAăZeroPage
00288                 case 0x95:    //ăSTAăZeroăPageăX
00289                 case 0x8D:  //ăSTAăAbsolute
00290                 case 0x9D:  //ăSTAăAbsoluteăX
00291                 case 0x99:    //ăSTAăAbsoluteăY
00292                 case 0x81:    //ăSTAăIndirectăX
00293                 case 0x91:    //ăSTAăIndirectăY
00294                     {
00295                         return "STA";
00296                     }
00297                 case 0x86:    //ăSTXăZeroăPage
00298                 case 0x96:    //ăSTXăZeroăPageăY
00299                 case 0x8E:  //ăSTXăAbsolute
00300                     {
00301                         return "STX";
00302                     }
00303                 case 0x84:    //ăSTYăZeroăPage
00304                 case 0x94:    //ăSTYăZeroăPageăX
00305                 case 0x8C:  //ăSTYăAbsolute
00306                     {
00307                         return "STY";
00308                     }
00309                 case 0xAA:  //ăTAXăImplied
00310                     {
00311                         return "TAX";
00312                     }
00313                 case 0xA8:    //ăTAYăImplied
00314                     {
00315                         return "TAY";
00316                     }
00317                 case 0xBA:  //ăTSXăImplied
00318                     {
00319                         return "TSX";
00320                     }
00321                 case 0x8A:  //ăTXAăImplied
00322                     {
00323                         return "TXA";
00324                     }
00325                 case 0x9A:  //ăTXSăImplied
00326                     {
00327                         return "TXS";
00328                     }
00329                 case 0x98:    //ăTYAăImplied
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 intead 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 occuring 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 opertion. 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 occurrs

- void ProcessIRQ ()

  This is ran anytime an IRQ occurrs

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

addressingMode	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

addressingMode	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

addressingMode	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

addressingMode	

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

performBranch	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

addressingMode	The addressing mode to use
decrement	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

useXRegister	If the operation is using the X or Y register
decrement	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

addressingMode	The addressing mode to use
comparisonValue	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]`

Coverts the Flags into its byte representation.

Parameters

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

Returns

Definition at line 1521 of file W65C02.cs.

```
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         }
```

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

addressingMode	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                     }
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, AbsolueteX 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, AbsolueteX 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
      it reads from the wrong location!
00772                             address += 256 * MemoryMap.Read(ProgramCounter - 255);
00773                             ProgramCounter = address;
00774                         }
00775                         else
00776                         {
00777                             ProgramCounter = GetAddressByAddressingMode(AddressingMode.Absolute);
00778                         }
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, Absolue 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),
      (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),
      (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),
      (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),
      (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),
      (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),
      (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),
      (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),
      (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),
      (byte)XRegister);
01230                       break;
01231                   }
01232               //STX Store Index X, Absolute, 3 Bytes, 4 Cycles
01233               case 0x8E:
01234                   {
01235                       MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.Absolute),
```

```
      (byte)XRegister);
01236                         break;
01237                     }
01238                 //STY Store Index Y, Zero Page, 2 Bytes, 3 Cycles
01239                 case 0x84:
01240                     {
01241                         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.ZeroPage),
      (byte)YRegister);
01242                         break;
01243                     }
01244                 //STY Store Index Y, Zero Page X, 2 Bytes, 4 Cycles
01245                 case 0x94:
01246                     {
01247                         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.ZeroPageX),
      (byte)YRegister);
01248                         break;
01249                     }
01250                 //STY Store Index Y, Absolute, 2 Bytes, 4 Cycles
01251                 case 0x8C:
01252                     {
01253                         MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.Absolute),
      (byte)YRegister);
01254                         break;
01255                     }
01256 #endregion
01257
01258 #region Transfer Operations
01259                 //TAX Transfer Accumulator to X Register, Implied, 1 Bytes, 2 Cycles
01260                 case 0xAA:
01261                     {
01262                         IncrementCycleCount();
01263                         XRegister = Accumulator;
01264
01265                         SetNegativeFlag(XRegister);
01266                         SetZeroFlag(XRegister);
01267                         break;
01268                     }
01269                 //TAY Transfer Accumulator to Y Register, 1 Bytes, 2 Cycles
01270                 case 0xA8:
01271                     {
01272                         IncrementCycleCount();
01273                         YRegister = Accumulator;
01274
01275                         SetNegativeFlag(YRegister);
01276                         SetZeroFlag(YRegister);
01277                         break;
01278                     }
01279                 //TXA Transfer X Register to Accumulator, Implied, 1 Bytes, 2 Cycles
01280                 case 0x8A:
01281                     {
01282                         IncrementCycleCount();
01283                         Accumulator = XRegister;
01284
01285                         SetNegativeFlag(Accumulator);
01286                         SetZeroFlag(Accumulator);
01287                         break;
01288                     }
01289                 //TYA Transfer Y Register to Accumulator, Implied, 1 Bytes, 2 Cycles
01290                 case 0x98:
01291                     {
01292                         IncrementCycleCount();
01293                         Accumulator = YRegister;
01294
01295                         SetNegativeFlag(Accumulator);
01296                         SetZeroFlag(Accumulator);
01297                         break;
01298                     }
01299 #endregion
01300
01301                 //NOP Operation, Implied, 1 Byte, 2 Cycles
01302                 case 0xEA:
01303                     {
01304                         IncrementCycleCount();
01305                         break;
01306                     }
01307
01308                 default:
01309                     throw new NotSupportedException(string.Format("The OpCode {0} is not supported",
      CurrentOpCode));
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

addressingMode	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++) « 8));
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 boundry, 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 boundry, 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           }
```

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 
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         }
```

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

addressingMode	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.MoveProgramCounterBy↩ RelativeValue (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
      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         }
```

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

addressingMode	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
      translated
01502             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

value	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
      translated
01513             MemoryMap.Write(StackPointer + 0x100, value);
01514         }
```

6.41.3.25 ProcessIRQ() void Hardware.W65C02.ProcessIRQ ( ) [inline], [private]

This is ran anytime an IRQ occurrs

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 
02429             SetDisassembly();
02430         }
```

6.41.3.26 ProcessNMI() void Hardware.W65C02.ProcessNMI ( ) [inline], [private]

This is ran anytime an NMI occurrs

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) <<
      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 opertion. 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

addressingMode	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

addressingMode	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("${0}{1}",
      address2.Value.ToString("X").PadLeft(2, '0'), address1.Value.ToString("X").PadLeft(2, '0'));
01546                         break;
01547                     }
01548                 case AddressingMode.AbsoluteX:
01549                     {
01550                         disassembledStep = string.Format("${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 
01616                         break;
01617                     }
01618                 case AddressingMode.ZeroPage:
01619                     {
01620                         address2 = null;
01621 
01622                         disassembledStep = string.Format("${0}",
      address1.Value.ToString("X").PadLeft(2, '0'));
01623                         break;
01624                     }
01625                 case AddressingMode.ZeroPageX:
01626                     {
01627                         address2 = null;
01628 
01629                         disassembledStep = string.Format("${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 
01645             CurrentDisassembly = new Disassembly
01646             {
```

```
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

value	

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

value	

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

addressingMode	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 
02275             SetNegativeFlag(newValue);
02276             SetZeroFlag(newValue);
02277 
02278             Accumulator = newValue;
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 intead 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 occuring 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

timer	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

sender	
e	

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

sender	
e	

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

address	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

value	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

value	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

address	The address to write data to.
data	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) ==
      IER_EN))
00233             {
00234                 T2Init(T2Interval);
00235             }
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 referemce 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

serialPort	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

serialPort	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                 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) ||
      (Object.StopBits == StopBits.One))
00520             {
00521                 controlRegister &= 0x7F;
00522             }
00523             else
00524             {
00525                 throw new ArgumentOutOfRangeException("StopBits or combination of StopBits and
      DataBits is invalid!");
00526             }
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 
00549             if (Object.BaudRate == 115200)
00550             {
00551                 controlRegister &= 0xF0;
00552             }
00553             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         }
```

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             }
00282             else if (address == Offset + 1)
00283             {
00284                 StatusRegisterUpdate();
00285             }
00286             else if (address == Offset + 2)
00287             {
00288                 CommandRegisterUpdate();
00289             }
00290             else if (address == Offset + 3)
00291             {
00292                 ControlRegisterUpdate();
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

port	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 
00074             ComInit(Object);
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

port	COM Port to use for I/O
baudRate	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 
00088             ComInit(Object);
00089         }
```

6.43.3.13 Read() byte Hardware.W65C51.Read (int *address*) [inline]

Returns the byte at a given address.

Parameters

address	

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

sender	
e	

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

address	The address to write data to
data	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

data	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; }
```

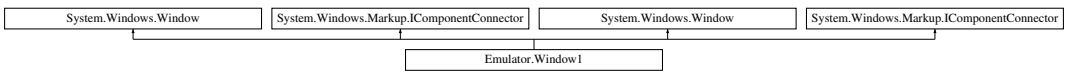
The documentation for this class was generated from the following file:

- 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 resourceLocater = new System.Uri("/Emulator;component/memorymap.xaml",
      System.UriKind.Relative);
00064
00065 #line 1 "..\..\..\MemoryMap.xaml"
00066             System.Windows.Application.LoadComponent(this, resourceLocater);
00067
00068 #line default
00069 #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 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         }
```

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         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 ProgamCounter 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 /// <summary>
```

```
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 Accummulator
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
      disk
00008 /// </summary>
00009     [Serializable]
00010     public class StateFileModel
00011     {
00012 /// <summary>
00013 /// The Number of Cycles the Program has Ran so Far
00014 /// </summary>
00015         public int NumberOfCycles { get; set; }
00016
00017 /// <summary>
00018 /// The output of the program
00019 /// </summary>
00020         public IList<OutputLog> OutputLog { get; set; }
00021
00022 /// <summary>
00023 /// The Processor Object that is being saved
00024 /// </summary>
00025         public Hardware.W65C02 W65C02 { get; set; }
00026
00027 /// <summary>
00028 /// The first VIA Object that is being saved
00029 /// </summary>
00030         public Hardware.W65C22 W65C22 { get; set; }
00031
00032 /// <summary>
00033 /// 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
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
```

7.53 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}"
      "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
```

7.55 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
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
```

7.59 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}"
      "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
```

7.61 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 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
```

7.83 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
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
```

7.85 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 "..\..\..\MainWindow.xaml" "{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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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 "..\..\..\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 "..\..\..\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.InterrupFlag = ((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 "..\..\..\MainWindow.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
      "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",
      "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",
      "CA1823:AvoidUnusedPrivateFields")]
00181         internal System.Windows.Controls.TextBlock StackPointerRegisterText;
00182
00183 #line default
00184 #line hidden
00185
00186
00187 #line 147 "..\..\..\MainWindow.xaml"
00188         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1823:AvoidUnusedPrivateFields")]
00189         internal System.Windows.Controls.TextBlock AText;
00190
00191 #line default
00192 #line hidden
00193
00194
00195 #line 148 "..\..\..\MainWindow.xaml"
00196         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1823:AvoidUnusedPrivateFields")]
00197         internal System.Windows.Controls.TextBlock CurrentInstructionText;
00198
00199 #line default
00200 #line hidden
00201
00202
00203 #line 149 "..\..\..\MainWindow.xaml"
00204         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1823:AvoidUnusedPrivateFields")]
00205         internal System.Windows.Controls.TextBlock ProgramCounterText;
00206
00207 #line default
00208 #line hidden
00209
00210
00211 #line 150 "..\..\..\MainWindow.xaml"
00212         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1823:AvoidUnusedPrivateFields")]
00213         internal System.Windows.Controls.TextBlock CycleCountText;
00214
00215 #line default
00216 #line hidden
00217
00218
00219 #line 151 "..\..\..\MainWindow.xaml"
00220         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1823:AvoidUnusedPrivateFields")]
00221         internal System.Windows.Controls.CheckBox CarryFlag;
00222
00223 #line default
00224 #line hidden
00225
00226
00227 #line 152 "..\..\..\MainWindow.xaml"
00228         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1823:AvoidUnusedPrivateFields")]
00229         internal System.Windows.Controls.TextBlock CarryFlagText;
00230
00231 #line default
00232 #line hidden
00233
00234
00235 #line 153 "..\..\..\MainWindow.xaml"
00236         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1823:AvoidUnusedPrivateFields")]
00237         internal System.Windows.Controls.CheckBox ZeroFlag;
00238
00239 #line default
00240 #line hidden
00241
00242
00243 #line 154 "..\..\..\MainWindow.xaml"
00244         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1823:AvoidUnusedPrivateFields")]
00245         internal System.Windows.Controls.TextBlock ZeroFlagText;
00246
00247 #line default
00248 #line hidden
```

```
00249
00250
00251 #line 155 "..\..\..\MainWindow.xaml"
00252         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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
      [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.InterrupFlag = ((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 "..\..\..\MainWindow.xaml" "{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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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 "..\..\..\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 "..\..\..\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.InterrupFlag = ((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 "..\..\..\MainWindow.xaml" "{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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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 "..\..\..\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 "..\..\..\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.InterrupFlag = ((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}"
      "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",
      "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",
      "CA1823:AvoidUnusedPrivateFields")]
00181         internal System.Windows.Controls.TextBlock StackPointerRegisterText;
00182
00183 #line default
00184 #line hidden
00185
00186
00187 #line 147 "..\..\..\MainWindow.xaml"
00188         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1823:AvoidUnusedPrivateFields")]
00189         internal System.Windows.Controls.TextBlock AText;
00190
00191 #line default
00192 #line hidden
00193
00194
00195 #line 148 "..\..\..\MainWindow.xaml"
00196         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1823:AvoidUnusedPrivateFields")]
00197         internal System.Windows.Controls.TextBlock CurrentInstructionText;
00198
00199 #line default
00200 #line hidden
00201
00202
00203 #line 149 "..\..\..\MainWindow.xaml"
00204         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1823:AvoidUnusedPrivateFields")]
00205         internal System.Windows.Controls.TextBlock ProgramCounterText;
00206
00207 #line default
00208 #line hidden
00209
00210
00211 #line 150 "..\..\..\MainWindow.xaml"
00212         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1823:AvoidUnusedPrivateFields")]
00213         internal System.Windows.Controls.TextBlock CycleCountText;
00214
00215 #line default
00216 #line hidden
00217
00218
00219 #line 151 "..\..\..\MainWindow.xaml"
00220         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1823:AvoidUnusedPrivateFields")]
00221         internal System.Windows.Controls.CheckBox CarryFlag;
00222
00223 #line default
00224 #line hidden
00225
00226
00227 #line 152 "..\..\..\MainWindow.xaml"
00228         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1823:AvoidUnusedPrivateFields")]
00229         internal System.Windows.Controls.TextBlock CarryFlagText;
00230
00231 #line default
00232 #line hidden
00233
00234
00235 #line 153 "..\..\..\MainWindow.xaml"
00236         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1823:AvoidUnusedPrivateFields")]
00237         internal System.Windows.Controls.CheckBox ZeroFlag;
00238
00239 #line default
00240 #line hidden
00241
00242
00243 #line 154 "..\..\..\MainWindow.xaml"
00244         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1823:AvoidUnusedPrivateFields")]
00245         internal System.Windows.Controls.TextBlock ZeroFlagText;
00246
00247 #line default
00248 #line hidden
00249
00250
00251 #line 155 "..\..\..\MainWindow.xaml"
00252         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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
      [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.InterrupFlag = ((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}"
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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",
      "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 "..\..\..\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 "..\..\..\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.InterrupFlag = ((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}"
      "7B2B3A560666411FAE151D1ACBF76FF18A3D6578E63FF227C3D020B54E19F5CB"
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 /// 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}"
      "7B2B3A560666411FAE151D1ACBF76FF18A3D6578E63FF227C3D020B54E19F5CB"
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 /// 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 
```

7.101 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}"
      "7B2B3A560666411FAE151D1ACBF76FF18A3D6578E63FF227C3D020B54E19F5CB"
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 /// 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
```

7.103 Emulator/obj/x86/Release/MemoryVisual.g.i.cs File Reference

Classes

- class Emulator.MemoryVisual

  Interaction logic for Window1.xaml

Namespaces

- namespace Emulator

7.104 MemoryVisual.g.i.cs

Go to the documentation of this file.

```
00001 #pragma checksum "..\..\..\MemoryVisual.xaml" "{8829d00f-11b8-4213-878b-770e8597ac16}"
      "7B2B3A560666411FAE151D1ACBF76FF18A3D6578E63FF227C3D020B54E19F5CB"
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 /// 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
```

7.105 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}"
      "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",
      "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 resourceLocater = new System.Uri("/Emulator;component/savefile.xaml",
      System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097             System.Windows.Application.LoadComponent(this, resourceLocater);
00098
00099 #line default
00100 #line hidden
00101         }
00102
00103         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00104         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00105
      [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00106         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
      "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00107         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
      "CA1502:AvoidExcessiveComplexity")]
00108         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1800:DoNotCastUnnecessarily")]
00109         void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
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         }
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}"
      "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",
      "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 resourceLocater = new System.Uri("/Emulator;component/savefile.xaml",
      System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097             System.Windows.Application.LoadComponent(this, resourceLocater);
00098
00099 #line default
00100 #line hidden
00101         }
00102
00103         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00104         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00105
      [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00106         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
      "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00107         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
      "CA1502:AvoidExcessiveComplexity")]
00108         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1800:DoNotCastUnnecessarily")]
00109         void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
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         }
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}"
      "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",
      "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 resourceLocater = new System.Uri("/Emulator;component/savefile.xaml",
      System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097             System.Windows.Application.LoadComponent(this, resourceLocater);
00098
00099 #line default
00100 #line hidden
00101         }
00102
00103         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00104         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00105
      [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00106         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
      "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00107         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
      "CA1502:AvoidExcessiveComplexity")]
00108         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1800:DoNotCastUnnecessarily")]
00109         void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
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         }
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}"
      "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",
      "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 resourceLocater = new System.Uri("/Emulator;component/savefile.xaml",
      System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097             System.Windows.Application.LoadComponent(this, resourceLocater);
00098
00099 #line default
00100 #line hidden
00101         }
00102
00103         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00104         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00105
      [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00106         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
      "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00107         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
      "CA1502:AvoidExcessiveComplexity")]
00108         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1800:DoNotCastUnnecessarily")]
00109         void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
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         }
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}"
      "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",
      "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 resourceLocater = new System.Uri("/Emulator;component/savefile.xaml",
      System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097             System.Windows.Application.LoadComponent(this, resourceLocater);
00098
00099 #line default
00100 #line hidden
00101         }
00102
00103         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00104         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00105
      [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00106         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
      "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00107         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
      "CA1502:AvoidExcessiveComplexity")]
00108         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1800:DoNotCastUnnecessarily")]
00109         void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
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         }
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}"
      "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",
      "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 resourceLocater = new System.Uri("/Emulator;component/savefile.xaml",
      System.UriKind.Relative);
00095
00096 #line 1 "..\..\..\SaveFile.xaml"
00097             System.Windows.Application.LoadComponent(this, resourceLocater);
00098
00099 #line default
00100 #line hidden
00101         }
00102
00103         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00104         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00105
      [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00106         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
      "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00107         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
      "CA1502:AvoidExcessiveComplexity")]
00108         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1800:DoNotCastUnnecessarily")]
00109         void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target) {
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         }
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}"
      "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
      [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
```

7.119 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.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",
      "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
      [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
```

7.121 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.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",
      "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
      [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
```

7.123 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",
      "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
      [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
```

7.125 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.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",
      "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
      [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",
      "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
      [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
```

7.129 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",
      "CA1823:AvoidUnusedPrivateFields")]
00046         internal System.Windows.Controls.DataGrid MemoryMap;
00047
00048 #line default
00049 #line hidden
00050
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 resourceLocater = new System.Uri("/Emulator;component/memorymap.xaml",
      System.UriKind.Relative);
00064
00065 #line 1 "..\..\..\MemoryMap.xaml"
00066             System.Windows.Application.LoadComponent(this, resourceLocater);
00067
00068 #line default
00069 #line hidden
00070         }
00071
00072         [System.Diagnostics.DebuggerNonUserCodeAttribute()]
00073         [System.CodeDom.Compiler.GeneratedCodeAttribute("PresentationBuildTasks", "4.0.0.0")]
00074
      [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00075         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
      "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00076         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
      "CA1502:AvoidExcessiveComplexity")]
```

```
00077         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "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}"
      "4BA73942B3E4CC642C22E491C94CD66BBC88A7F17A65FB594B61788FC9DFDB08"
00003 //------------------------------------------------------------------------------
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 
      [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Never)]
00072         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Design",
      "CA1033:InterfaceMethodsShouldBeCallableByChildTypes")]
00073         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Maintainability",
      "CA1502:AvoidExcessiveComplexity")]
00074         [System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
      "CA1800:DoNotCastUnnecessarily")]
00075         void System.Windows.Markup.IComponentConnector.Connect(int connectionId, object target)
00076         {
00077             this._contentLoaded = true;
00078         }
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 Prorgam 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 progam 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 /// Relay Command 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                     MessageBox.Show("Settings file contains old information...\nDeleting old settings
      file...",
00229                                     "Settings file stale!", MessageBoxButton.OKCancel,
      MessageBoxImage.Warning,
00230                                     MessageBoxResult.OK);
00231                     // Close the file, then delete it.
00232                     _stream.Close();
00233                     File.Delete(FileLocations.SettingsFile);
00234                     SettingsModel = SettingsFile.CreateNew();
00235                 }
00236             }
00237             else
00238             {
00239                 MessageBox.Show("Creating new settings file...");
00240                 SettingsModel = SettingsFile.CreateNew();
00241             }
00242             _stream.Close();
00243 
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, GenericNotifcation);
00276             Messenger.Default.Register<NotificationMessage<RomFileModel»(this,
      BinaryLoadedNotification);
00277             Messenger.Default.Register<NotificationMessage<SettingsModel»(this,
      SettingsAppliedNotifcation);
00278             Messenger.Default.Register<NotificationMessage<StateFileModel»(this,
      StateLoadedNotifcation);
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" +
00301                                                            "Be warned that this is a beta build.\n" +
00302                                                            "It may break or have bugs.",
      Versioning.Product.Name),
00303                                                            Versioning.Product.Title,
      MessageBoxButton.OKCancel,
00304                                                            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!",
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,
      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 StateLoadedNotifcation(NotificationMessage<StateFileModel> notificationMessage)
00372         {
00373             if (notificationMessage.Notification != "StateLoaded")
00374             {
00375                 return;
00376             }
00377
00378             Reset();
00379
00380             OutputLog = new
      MultiThreadedObservableCollection<OutputLog>(notificationMessage.Content.OutputLog);
00381             RaisePropertyChanged("OutputLog");
00382
00383             NumberOfCycles = notificationMessage.Content.NumberOfCycles;
00384
00385             W65C02 = notificationMessage.Content.W65C02;
00386             W65C22 = notificationMessage.Content.W65C22;
00387             MM65SIB = notificationMessage.Content.MM65SIB;
00388             W65C51 = notificationMessage.Content.W65C51;
00389             AT28C010 = notificationMessage.Content.AT28C010;
00390             AT28C64 = notificationMessage.Content.AT28C64;
00391             UpdateMemoryPage();
00392             UpdateUi();
00393
00394             IsRomLoaded = true;
00395             RaisePropertyChanged("IsRomLoaded");
00396         }
00397
00398         private void GenericNotifcation(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 /// <summary>
00044 /// Relay Command 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 multiplyer = 0;
00080             for (ushort i = (ushort)offset; i < 256 * (_memoryPageOffset + 1); i++)
00081             {
00082                 MemoryPage.Add(new MemoryRowModel
00083                 {
00084                     Offset = ((16 * multiplyer) + 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                 multiplyer++;
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
      default instance.
00049 /// </summary>
00050         [PreferredConstructor]
00051         public SaveFileViewModel()
00052         {
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 }
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
      full value would then be added to the X Register.
00016 /// If the X register was 0x01 then the address would be 0xFF01.  and the value stored there would
      have an ADC operation performed on it and the value would
00017 /// be added to the accumulator.
00018 /// </summary>
00019         AbsoluteX = 2,
00020 /// <summary>
00021 /// In this mode a full address is given to operation on IE: Memory byte[] { 0x79, 0x00, 0xFF } The
      full value would then be added to the Y Register.
00022 /// If the Y register was 0x01 then the address would be 0xFF01.  and the value stored there would
      have an ADC operation performed on it and the value would
00023 /// be added to the accumulator
00024 /// </summary>
00025         AbsoluteY = 3,
00026 /// <summary>
00027 /// In this mode the instruction operates on the accumulator.  No operands are needed.
00028 /// </summary>
00029         Accumulator = 4,
00030 /// <summary>
00031 /// In this mode, the value to operate on immediately follows the instruction.  IE: Memory byte[] {
      0x69, 0x01 }
00032 /// would perform an ADC operation and Add 0x01 directly to the accumulator
00033 /// </summary>
00034         Immediate = 5,
00035 /// <summary>
00036 /// No address is needed for this mode.  EX: BRK (Break), CLC (Clear Carry Flag) etc
00037 /// </summary>
00038         Implied = 6,
00039 /// <summary>
00040 /// In this mode assume the following
00041 /// Memory = { 0x61, 0x02, 0x04, 0x00, 0x03 }
00042 /// RegisterX = 0x01
00043 /// 1.  Take the sum of the X Register and the value after the opcode 0x01 + 0x01 = 0x02.
00044 /// 2.  Starting at position 0x02 get an address (0x04,0x00) = 0x0004
00045 /// 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 addres 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:    //ăADCăImmediate
00012                 case 0x65:    //ăADCăZeroăPage
00013                 case 0x75:    //ăADCăZeroăPageăX
00014                 case 0x6D:  //ăADCăAbsolute
00015                 case 0x7D:  //ăADCăAbsoluteăX
00016                 case 0x79:    //ăADCăAbsoluteăY
00017                 case 0x61:    //ăADCăIndrectăX
00018                 case 0x71:    //ăADCăIndirectăY
00019                     {
00020                         return "ADC";
00021                     }
00022                 case 0x29:    //ăANDăImmediate
00023                 case 0x25:    //ăANDăZeroăPage
00024                 case 0x35:    //ăANDăZeroăPageăX
00025                 case 0x2D:  //ăANDăAbsolute
00026                 case 0x3D:  //ăANDăAbsoluteăX
00027                 case 0x39:    //ăANDăAbsoluteăY
00028                 case 0x21:    //ăANDăIndirectăX
00029                 case 0x31:    //ăANDăIndirectăY
00030                     {
00031                         return "AND";
00032                     }
00033                 case 0x0A:  //ăASLăAccumulator
00034                 case 0x06:    //ăASLăZeroăPage
00035                 case 0x16:    //ăASLăZeroăPageăX
00036                 case 0x0E:  //ăASLăAbsolute
00037                 case 0x1E:  //ăASLăAbsoluteăX
00038                     {
00039                         return "ASL";
00040                     }
00041                 case 0x90:    //ăBCCăRelative
00042                     {
00043                         return "BCC";
00044                     }
00045                 case 0xB0:    //ăBCSăRelative
00046                     {
00047                         return "BCS";
00048                     }
00049                 case 0xF0:    //ăBEQăRelative
00050                     {
00051                         return "BEQ";
00052                     }
00053                 case 0x24:    //ăBITăZeroăPage
00054                 case 0x2C:  //ăBITăAbsolute
00055                     {
00056                         return "BIT";
00057                     }
00058                 case 0x30:    //ăBMIăRelative
```

```
00059                   {
00060                       return "BMI";
00061                   }
00062               case 0xD0:    //ăBNEăRelative
00063                   {
00064                       return "BNE";
00065                   }
00066               case 0x10:    //ăBPLăRelative
00067                   {
00068                       return "BPL";
00069                   }
00070               case 0x00:    //ăBRKăImplied
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:    //ăCLCăImplied
00083                   {
00084                       return "CLC";
00085                   }
00086               case 0xD8:    //ăCLDăImplied
00087                   {
00088                       return "CLD";
00089                   }
00090               case 0x58:    //ăCLIăImplied
00091                   {
00092                       return "CLI";
00093                   }
00094               case 0xB8:    //ăCLVăImplied
00095                   {
00096                       return "CLV";
00097                   }
00098               case 0xC9:    //ăCMPăImmediate
00099               case 0xC5:    //ăCMPăZeroPage
00100               case 0xD5:    //ăCMPăZeroăPageăX
00101               case 0xCD:  //ăCMPăAbsolute
00102               case 0xDD:  //ăCMPăAbsoluteăX
00103               case 0xD9:    //ăCMPăAbsoluteăY
00104               case 0xC1:    //ăCMPăIndirectăX
00105               case 0xD1:    //ăCMPăIndirectăY
00106                   {
00107                       return "CMP";
00108                   }
00109               case 0xE0:    //ăCPXăImmediate
00110               case 0xE4:    //ăCPXăZeroPage
00111               case 0xEC:  //ăCPXăAbsolute
00112                   {
00113                       return "CPX";
00114                   }
00115               case 0xC0:    //ăCPYăImmediate
00116               case 0xC4:    //ăCPYăZeroPage
00117               case 0xCC:  //ăCPYăAbsolute
00118                   {
00119                       return "CPY";
00120                   }
00121               case 0xC6:    //ăDECăZeroăPage
00122               case 0xD6:    //ăDECăZeroăPageăX
00123               case 0xCE:  //ăDECăAbsolute
00124               case 0xDE:  //ăDECăAbsoluteăX
00125                   {
00126                       return "DEC";
00127                   }
00128               case 0xCA:  //ăDEXăImplied
00129                   {
00130                       return "DEX";
00131                   }
00132               case 0x88:    //ăDEYăImplied
00133                   {
00134                       return "DEY";
00135                   }
00136               case 0x49:    //ăEORăImmediate
00137               case 0x45:    //ăEORăZeroăPage
00138               case 0x55:    //ăEORăZeroăPageăX
00139               case 0x4D:  //ăEORăAbsolute
00140               case 0x5D:  //ăEORăAbsoluteăX
00141               case 0x59:    //ăEORăAbsoluteăY
00142               case 0x41:    //ăEORăIndrectăX
00143               case 0x51:    //ăEORăIndirectăY
00144                   {
00145                       return "EOR";
```

```
00146                 }
00147             case 0xE6:    //ăINCăZeroăPage
00148             case 0xF6:    //ăINCăZeroăPageăX
00149                 {
00150                     return "INC";
00151                 }
00152             case 0xE8:    //ăINXăImplied
00153                 {
00154                     return "INX";
00155                 }
00156             case 0xC8:    //ăINYăImplied
00157                 {
00158                     return "INY";
00159                 }
00160             case 0xEE:  //ăINCăAbsolute
00161             case 0xFE:  //ăINCăAbsoluteăX
00162                 {
00163                     return "INC";
00164                 }
00165             case 0x4C:  //ăJMPăAbsolute
00166             case 0x6C:  //ăJMPăIndirect
00167                 {
00168                     return "JMP";
00169                 }
00170             case 0x20:    //ăJSRăAbsolute
00171                 {
00172                     return "JSR";
00173                 }
00174             case 0xA9:    //ăLDAăImmediate
00175             case 0xA5:    //ăLDAăZeroăPage
00176             case 0xB5:    //ăLDAăZeroăPageăX
00177             case 0xAD:  //ăLDAăAbsolute
00178             case 0xBD:  //ăLDAăAbsoluteăX
00179             case 0xB9:    //ăLDAăAbsoluteăY
00180             case 0xA1:    //ăLDAăIndirectăX
00181             case 0xB1:    //ăLDAăIndirectăY
00182                 {
00183                     return "LDA";
00184                 }
00185             case 0xA2:    //ăLDXăImmediate
00186             case 0xA6:    //ăLDXăZeroăPage
00187             case 0xB6:    //ăLDXăZeroăPageăY
00188             case 0xAE:  //ăLDXăAbsolute
00189             case 0xBE:  //ăLDXăAbsoluteăY
00190                 {
00191                     return "LDX";
00192                 }
00193             case 0xA0:    //ăLDYăImmediate
00194             case 0xA4:    //ăLDYăZeroăPage
00195             case 0xB4:    //ăLDYăZeroăPageăY
00196             case 0xAC:  //ăLDYăAbsolute
00197             case 0xBC:  //ăLDYăAbsoluteăY
00198                 {
00199                     return "LDY";
00200                 }
00201             case 0x4A:  //ăLSRăAccumulator
00202             case 0x46:    //ăLSRăZeroăPage
00203             case 0x56:    //ăLSRăZeroăPageăX
00204             case 0x4E:  //ăLSRăAbsolute
00205             case 0x5E:  //ăLSRăAbsoluteăX
00206                 {
00207                     return "LSR";
00208                 }
00209             case 0xEA:  //ăNOPăImplied
00210                 {
00211                     return "NOP";
00212                 }
00213             case 0x09:    //ăORAăImmediate
00214             case 0x05:    //ăORAăZeroăPage
00215             case 0x15:    //ăORAăZeroăPageăX
00216             case 0x0D:  //ăORAăAbsolute
00217             case 0x1D:  //ăORAăAbsoluteăX
00218             case 0x19:    //ăORAăAbsoluteăY
00219             case 0x01:    //ăORAăIndirectăX
00220             case 0x11:    //ăORAăIndirectăY
00221                 {
00222                     return "ORA";
00223                 }
00224             case 0x48:    //ăPHAăImplied
00225                 {
00226                     return "PHA";
00227                 }
00228             case 0x08:    //ăPHPăImplied
00229                 {
00230                     return "PHP";
00231                 }
00232             case 0x68:    //ăPLAăImplied
```

```
00233                     {
00234                         return "PLA";
00235                     }
00236                 case 0x28:    //ăPLPăImplied
00237                     {
00238                         return "PLP";
00239                     }
00240                 case 0x2A:  //ăROLăAccumulator
00241                 case 0x26:    //ăROLăZeroăPage
00242                 case 0x36:    //ăROLăZeroăPageăX
00243                 case 0x2E:  //ăROLăAbsolute
00244                 case 0x3E:  //ăROLăAbsoluteăX
00245                     {
00246                         return "ROL";
00247                     }
00248                 case 0x6A:  //ăRORăAccumulator
00249                 case 0x66:    //ăRORăZeroăPage
00250                 case 0x76:    //ăRORăZeroăPageăX
00251                 case 0x6E:  //ăRORăAbsolute
00252                 case 0x7E:  //ăRORăAbsoluteăX
00253                     {
00254                         return "ROR";
00255                     }
00256                 case 0x40:    //ăRTIăImplied
00257                     {
00258                         return "RTI";
00259                     }
00260                 case 0x60:    //ăRTSăImplied
00261                     {
00262                         return "RTS";
00263                     }
00264                 case 0xE9:    //ăSBCăImmediate
00265                 case 0xE5:    //ăSBCăZeroăPage
00266                 case 0xF5:    //ăSBCăZeroăPageăX
00267                 case 0xED:  //ăSBCăAbsolute
00268                 case 0xFD:  //ăSBCăAbsoluteăX
00269                 case 0xF9:    //ăSBCăAbsoluteăY
00270                 case 0xE1:    //ăSBCăIndrectăX
00271                 case 0xF1:    //ăSBCăIndirectăY
00272                     {
00273                         return "SBC";
00274                     }
00275                 case 0x38:    //ăSECăImplied
00276                     {
00277                         return "SEC";
00278                     }
00279                 case 0xF8:    //ăSEDăImplied
00280                     {
00281                         return "SED";
00282                     }
00283                 case 0x78:    //ăSEIăImplied
00284                     {
00285                         return "SEI";
00286                     }
00287                 case 0x85:    //ăSTAăZeroPage
00288                 case 0x95:    //ăSTAăZeroăPageăX
00289                 case 0x8D:  //ăSTAăAbsolute
00290                 case 0x9D:  //ăSTAăAbsoluteăX
00291                 case 0x99:    //ăSTAăAbsoluteăY
00292                 case 0x81:    //ăSTAăIndirectăX
00293                 case 0x91:    //ăSTAăIndirectăY
00294                     {
00295                         return "STA";
00296                     }
00297                 case 0x86:    //ăSTXăZeroăPage
00298                 case 0x96:    //ăSTXăZeroăPageăY
00299                 case 0x8E:  //ăSTXăAbsolute
00300                     {
00301                         return "STX";
00302                     }
00303                 case 0x84:    //ăSTYăZeroăPage
00304                 case 0x94:    //ăSTYăZeroăPageăX
00305                 case 0x8C:  //ăSTYăAbsolute
00306                     {
00307                         return "STY";
00308                     }
00309                 case 0xAA:  //ăTAXăImplied
00310                     {
00311                         return "TAX";
00312                     }
00313                 case 0xA8:    //ăTAYăImplied
00314                     {
00315                         return "TAY";
00316                     }
00317                 case 0xBA:  //ăTSXăImplied
00318                     {
00319                         return "TSX";
```

```
00320                     }
00321                 case 0x8A:  //ăTXAăImplied
00322                     {
00323                         return "TXA";
00324                     }
00325                 case 0x9A:  //ăTXSăImplied
00326                     {
00327                         return "TXS";
00328                     }
00329                 case 0x98:    //ăTYAăImplied
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 occuring 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) «
      8));
00159             CurrentOpCode = MemoryMap.Read(ProgramCounter);
00160             //SetDisassembly();
00161             DisableInterruptFlag = true;
00162             _previousInterrupt = false;
00163             TriggerNmi = false;
00164             TriggerIRQ = false;
00165         }
00166
00167 /// <summary>
00168 /// Performs the next step on the processor
00169 /// </summary>
00170         public void NextStep()
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         }
00195
00196 /// <summary>
00197 /// The InterruptRequest or IRQ
00198 /// </summary>
00199         public void InterruptRequest()
00200         {
00201             TriggerIRQ = true;
00202         }
00203
00204 /// <summary>
00205 /// Gets the Number of Cycles that have elapsed
00206 /// </summary>
00207 /// <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
      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                     }
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, AbsolueteX 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, AbsolueteX 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
      it reads from the wrong location!
00772                             address += 256 * MemoryMap.Read(ProgramCounter - 255);
00773                             ProgramCounter = address;
00774                         }
00775                         else
00776                         {
00777                             ProgramCounter = GetAddressByAddressingMode(AddressingMode.Absolute);
00778                         }
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, Absolue 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),
    (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),
    (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),
    (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),
    (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),
    (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),
    (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),
    (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),
    (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),
    (byte)XRegister);
01230                           break;
01231                       }
01232                   //STX Store Index X, Absolute, 3 Bytes, 4 Cycles
01233                   case 0x8E:
01234                       {
01235                           MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.Absolute),
    (byte)XRegister);
01236                           break;
01237                       }
01238                   //STY Store Index Y, Zero Page, 2 Bytes, 3 Cycles
01239                   case 0x84:
01240                       {
01241                           MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.ZeroPage),
    (byte)YRegister);
01242                           break;
01243                       }
01244                   //STY Store Index Y, Zero Page X, 2 Bytes, 4 Cycles
01245                   case 0x94:
01246                       {
01247                           MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.ZeroPageX),
    (byte)YRegister);
01248                           break;
01249                       }
01250                   //STY Store Index Y, Absolute, 2 Bytes, 4 Cycles
01251                   case 0x8C:
01252                       {
01253                           MemoryMap.Write(GetAddressByAddressingMode(AddressingMode.Absolute),
    (byte)YRegister);
01254                           break;
01255                       }
01256 #endregion
01257
01258 #region Transfer Operations
01259                   //TAX Transfer Accumulator to X Register, Implied, 1 Bytes, 2 Cycles
01260                   case 0xAA:
01261                       {
01262                           IncrementCycleCount();
01263                           XRegister = Accumulator;
01264
01265                           SetNegativeFlag(XRegister);
01266                           SetZeroFlag(XRegister);
01267                           break;
01268                       }
01269                   //TAY Transfer Accumulator to Y Register, 1 Bytes, 2 Cycles
01270                   case 0xA8:
01271                       {
01272                           IncrementCycleCount();
01273                           YRegister = Accumulator;
01274
01275                           SetNegativeFlag(YRegister);
01276                           SetZeroFlag(YRegister);
01277                           break;
01278                       }
01279                   //TXA Transfer X Register to Accumulator, Implied, 1 Bytes, 2 Cycles
01280                   case 0x8A:
01281                       {
01282                           IncrementCycleCount();
01283                           Accumulator = XRegister;
01284
01285                           SetNegativeFlag(Accumulator);
01286                           SetZeroFlag(Accumulator);
01287                           break;
01288                       }
01289                   //TYA Transfer Y Register to Accumulator, Implied, 1 Bytes, 2 Cycles
01290                   case 0x98:
01291                       {
01292                           IncrementCycleCount();
01293                           Accumulator = YRegister;
01294
01295                           SetNegativeFlag(Accumulator);
01296                           SetZeroFlag(Accumulator);
01297                           break;
01298                       }
01299 #endregion
01300
01301                   //NOP Operation, Implied, 1 Byte, 2 Cycles
01302                   case 0xEA:
01303                       {
01304                           IncrementCycleCount();
01305                           break;
```

```
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                     {
01347                         return (MemoryMap.Read(ProgramCounter++) | (MemoryMap.Read(ProgramCounter++) «
      8));
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 boundry, 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 boundry, 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 /// Coverts 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("${0}",
      newProgramCounter.ToString("X").PadLeft(4, '0'));
01615
01616                           break;
01617                       }
01618                   case AddressingMode.ZeroPage:
01619                       {
01620                           address2 = null;
01621
01622                           disassembledStep = string.Format("${0}",
      address1.Value.ToString("X").PadLeft(2, '0'));
01623                           break;
01624                       }
01625                   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 
01645             CurrentDisassembly = new Disassembly
01646             {
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         }
01674 
01675         private int WrapProgramCounter(int value)
01676         {
01677             return value & 0xFFFF;
01678         }
01679 
01680         private AddressingMode GetAddressingMode()
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
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) &
      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         }
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 vaulue 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
      memory
02089 /// </summary>
02090 /// <param name="addressingMode">The addressing mode to use</param>
02091         private void EorOperation(AddressingMode addressingMode)
02092         {
02093             Accumulator = Accumulator ^ MemoryMap.Read(GetAddressByAddressingMode(addressingMode));
02094
02095             SetNegativeFlag(Accumulator);
02096             SetZeroFlag(Accumulator);
02097         }
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
      memory.
02251 /// </summary>
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 
02346 /// <summary>
02347 /// The BRK routine.  Called when a BRK occurs.
02348 /// </summary>
02349         private void BreakOperation(bool isBrk, int vector)
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         }
02379 
02380 /// <summary>
02381 /// The RTI routine.  Called when returning from a BRK opertion.
02382 /// Note:  when called after a BRK operation the Program Counter is not set to the location after the
      BRK,
02383 /// it is set +1
02384 /// </summary>
02385         private void ReturnFromInterruptOperation()
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         }
02404
02405 /// <summary>
02406 /// This is ran anytime an NMI occurrs
02407 /// </summary>
02408         private void ProcessNMI()
02409         {
02410             ProgramCounter--;
02411             BreakOperation(false, 0xFFFA);
02412             CurrentOpCode = MemoryMap.Read(ProgramCounter);
02413
02414             SetDisassembly();
02415         }
02416
02417 /// <summary>
02418 /// This is ran anytime an IRQ occurrs
02419 /// </summary>
02420         private void ProcessIRQ()
02421         {
02422             if (DisableInterruptFlag)
02423                 return;
02424
02425             ProgramCounter--;
02426             BreakOperation(false, 0xFFFE);
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 referemce 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,
    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                 stream.Close();
00185                 file.Close();
00186                 return;
00187             }
00188         }
00189 
00190 /// <summary>
00191 /// Called when the window is closed.
00192 /// </summary>
00193 ///
00194 /// <param name="serialPort">SerialPort Object to close</param>
00195         private void ComFini(SerialPort serialPort)
00196         {
00197             if (serialPort != null)
00198             {
00199                 serialPort.Close();
00200             }
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,
      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         }
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             }
00282             else if (address == Offset + 1)
00283             {
00284                 StatusRegisterUpdate();
00285             }
00286             else if (address == Offset + 2)
00287             {
00288                 CommandRegisterUpdate();
00289             }
00290             else if (address == Offset + 3)
00291             {
00292                 ControlRegisterUpdate();
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) ||
    (Object.StopBits == StopBits.One))
00520             {
00521                 controlRegister &= 0x7F;
00522             }
00523             else
00524             {
00525                 throw new ArgumentOutOfRangeException("StopBits or combination of StopBits and
    DataBits is invalid!");
00526             }
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
00549             if (Object.BaudRate == 115200)
00550             {
00551                 controlRegister &= 0xF0;
00552             }
00553             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 (Overrun)
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 || 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         }
00708 #endregion
00709     }
00710 }
```

Index