Real Time Memory Editor Development

Copyright PureIso 2013

Table of Contents

[1. Getting Started 2](#_Toc348200375)

[2. Peek Poker 2](#_Toc348200376)

[2.1 Dumping Memory 3](#_Toc348200377)

[2.1.1 Xbox 360 Memory Map 3](#_Toc348200378)

[2.1.2 Dumping 3](#_Toc348200379)

[2.2 Peeking and Poking 4](#_Toc348200380)

[2.3 Searching 5](#_Toc348200381)

[3. Using Peek Poker Interface 6](#_Toc348200382)

[3.1 API 6](#_Toc348200383)

[3.2 Form Setup 7](#_Toc348200384)

[3.3 Coding – Entry Point 8](#_Toc348200385)

[3.4 Coding - MainForm - Without Pointer 10](#_Toc348200386)

[3.5 Coding – MainForm – Using Pointer 12](#_Toc348200387)

[3.6 Coding - Application Optional Form Plugin 16](#_Toc348200388)

[4 Links 18](#_Toc348200389)

[5 Thanks 18](#_Toc348200390)

# Getting Started

A real time editor is an application that allows you to edit memory in real time.

# Peek Poker

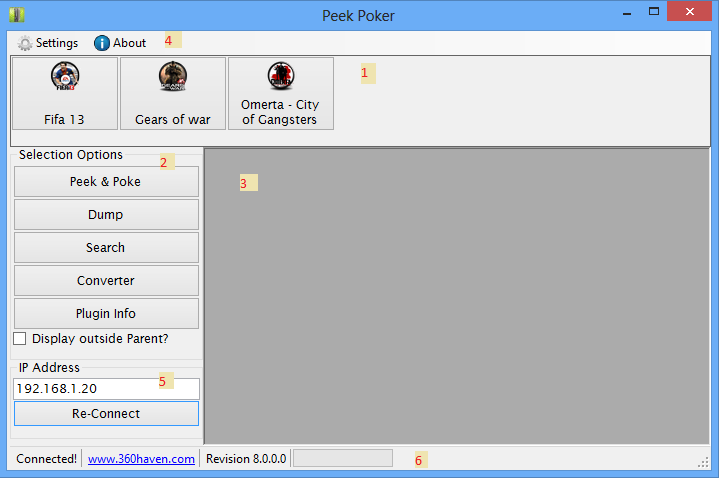


Figure 1: The user interface for Peek Poker

1. Plugin Editor Location – Simply click on the button with the game you want to load.
2. Option Panel – Contains Forms to aid with memory editing.

* Peek & Poke – Peek and Poke memory addresses.
* Dump – Dump memory
* Search – Search Memory
* Converter – Convert various values to Hex
* Plugin Info – Get information about all plugins

1. Display area – all forms will be displayed here within the parent unless the Display outside is checked.
2. Menu Strip Area – You can hide panels in the setting and you can video the about, credit going to major helpers.
3. IP Address Area – Connect to an IP address or Type “DEBUG” to unlock the panels for testers.

## Dumping Memory

### Xbox 360 Memory Map

* Virtual 4-KB range: 0x00000000 – 0x3FFFFFFF
* Virtual 64-KB range: 0x40000000 - 7FFFFFFF
* Image/Code 64-KB range: 0x80000000 - 8BFFFFFF
* Image/Code 4-KB range: 0x90000000 - 0x9FFFFFFF
* Physical 64-KB Full Ram range: 0xA0000000 - 0xBFFFFFFF
* Physical 16-MB Full Ram range: 0xC0000000 - 0xDFFFFFFF
* Physical 4-KB Full Ram range: 0xE0000000 - 0xFFFFFFFF

### Dumping

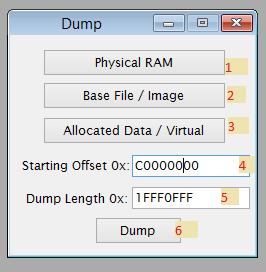


Figure 2: The user interface for the dump

1. Physical Ram – Dump the physical ram.
2. Base File – Dump the base file.
3. Allocated Data – Dump the allocated data.
4. Starting Offset – User can customise the dump offset
5. Dump Length – Users can customise the dump length
6. Dump – Commence the dumping.

## Peeking and Poking

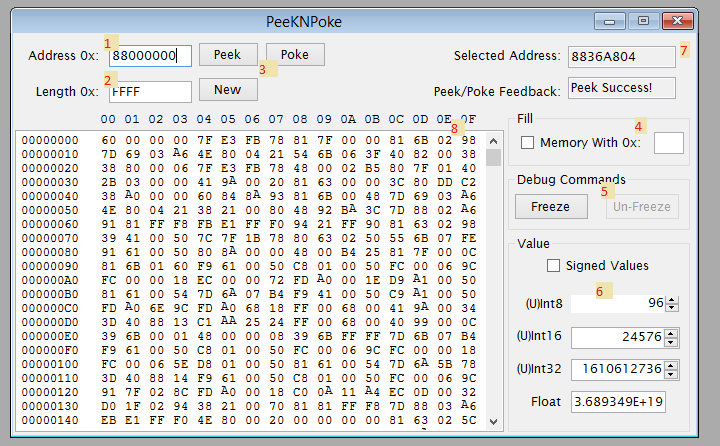


Figure 3: The user interface for peekNpoke form

1. Address – Poke or Peek the address using Hexadecimal values.
2. Length – This is the amount of bytes you want to display in the display area.
3. PeekNPoke Buttons – Allows you to Peek, Poke or reset values using the new button.
4. Fill – The purpose of this area is to fill an area in memory with the specific value, if checked it will overwrite (length) amount of bytes starting from the address.
5. Freeze / Un-Freeze – Freeze or Un-Freeze the game console.
6. Values – Convert highlighted values to decimal.
7. Information Area – Shows your current selected offset or last selected offset in the display box.
8. Display Box – Peeked Hex values will be displayed here.

## Searching

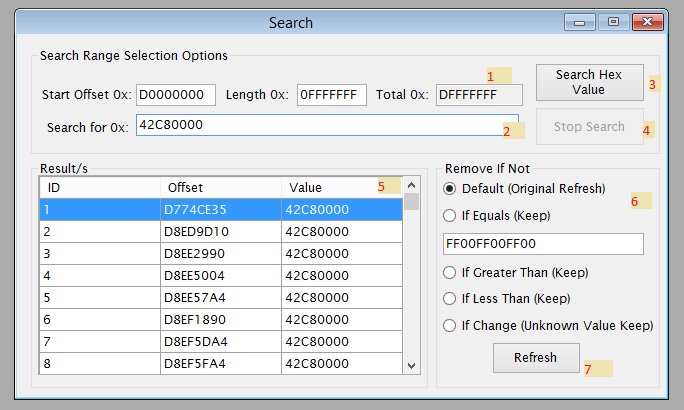


Figure 4: The user interface for the Search form

1. Search Range – User can enter the search offset, the search length and the total is also displayed.
2. Search Pattern – The pattern we are searching for.
3. Search Button - Commence the search.
4. Stop Button – Stop the search and display the results we currently have.
5. Result Panel – Search results are displayed here, the offset and the value. You can click on an id and (CTRL+C) to copy the offset.
6. Update Search – This group are allows you to update the search results.
7. Refresh Button – Refresh or update your results.

# Using Peek Poker Interface

## API

The Application Programming interface contains details of the object classes available to use in PeekPoker Interface library.



## 3.2 Form Setup

* Open Visual Studios and Start a new Windows Form Application Project

(As a convention)

* Rename your Form1 to MainForm
* Change MainForm Text -> \*\*\* - RTE (\*\*\* = Name of the game)
* FormBorderStyle -> FixedDialog
* MaximizeBox -> False
* Icon -> Browse for your Icon
* Font -> Lucida Sans, 9pt
* Add Reference -> PeekPoker.Interface
* Properties -> Build -> Platform target -> x86
* Properties -> Target Framework -> .Net Framework 2.0 (Better compatibility)
* Properties -> Output Type -> Class Library (Set this when done .dll instead of .exe)
* Properties -> Icon (Browse for your Icon)

(Checkout: http://www.coolutils.com/online/image-converter/)

* Add New Class
* Rename Class1 -> EntryPoint

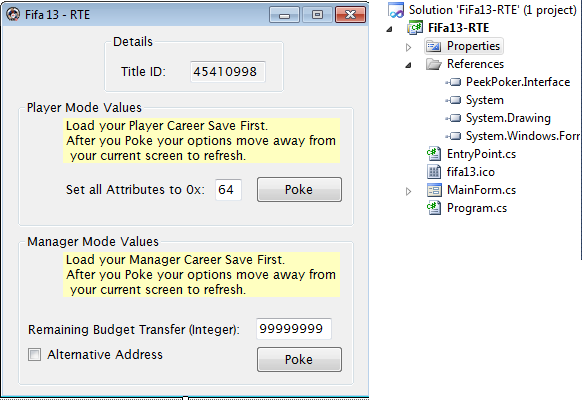


Figure 5: Sample Form and Solution View

## Coding – Entry Point

Below is the full EntryPoint code with comments:

using PeekPoker.Interface;

using System.Windows.Forms;

//===============================================

//Author: PureIso

//Description: FIFA 13 RTE EntryPoint

//===============================================

namespace FiFa13\_RTE

{

//Entry Point inherits from PeekPoker.Interface.AbstractPlugin

public class EntryPoint : AbstractPlugin

{

/// <summary>

/// A Thread Safe way to show messagebox using the codes in

/// PeekPoker

/// </summary>

internal static ShowMessageBoxHandler ShowMessageBox;

/// <summary>

/// A Thread Safe way to update the progressbar using the codes in

/// PeekPoker

/// </summary>

internal static UpdateProgressBarHandler UpdateProgressBar;

/// <summary>

/// A Thread Safe way to set text in a textbox using the codes in

/// PeekPoker

/// </summary>

internal static SetTextBoxTextDelegateHandler SetText;

/// <summary>

/// A Thread Safe way to get text from a textbox using the codes in

/// PeekPoker

/// </summary>

internal static GetTextBoxTextHandler GetText;

/// <summary>

/// A Thread Safe way to enable or disable controls using the codes in

/// PeekPoker

/// </summary>

internal static EnableControlHandler EnableControl;

/// <summary>

/// The real time memory accessor

/// </summary>

internal static RealTimeMemory Rtm;

/// <summary>

/// Constructor

/// </summary>

public EntryPoint()

{

//Application Information

//If you don't fill out any of the information AbstractPlugin will use

//the default which is "Unavailable"

base.ApplicationName = "Fifa 13";

base.Author = "PureIso";

base.Description = "Fifa 13 RealTime Editor";

base.Version = "1.0.0.0";

//Get the proper Icon

MainForm form = new MainForm();

base.Icon = form.Icon;

//Clean up

form.Dispose();

form.Close();

}

/// <summary>

/// Overrides the abstract Display method - if not called an empty form

/// will be loaded

/// </summary>

/// <param name="parent">The PeekPoker will be the parent</param>

public override void Display(Form parent)

{

//Handlers

ShowMessageBox = base.APShowMessageBox;

UpdateProgressBar = base.APUpdateProgressBar;

GetText = base.APGetTextBoxText;

SetText = base.APSetTextBoxText;

EnableControl = base.APEnableControl;

//You can get the RTM being using in PeekPoker or

//you can initialize you own

Rtm = base.APRtm;

MainForm form = base.IsMdiChild

? new MainForm { MdiParent = parent }

: new MainForm();

form.Show();

}

}

}

## Coding - MainForm - Without Pointer

Below is the full Main Form code with comments:

using System;

using System.Windows.Forms;

//===============================================

//Author: PureIso

//Description: FIFA 13 RTE MainForm

//===============================================

namespace FiFa13\_RTE

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

}

//When you click the attribute Poke button

private void attributePokeButton\_Click(object sender, EventArgs e)

{

try

{

//Initialise the hexString

string hexString = "";

//Get the byte String

string byteHex = EntryPoint.GetText(attributeValueTextBox);

//We need 34 bytes of the same value as Hex String

for (int i = 0; i < 34; i++)

{

//Concat each byte string to the Hex String

//It's faster than Getting the textbox text

hexString += byteHex;

}

//Poke that Address C53D7729 with the hexString Value - New Save

//This address is only useful after loading your save - Attributes

EntryPoint.Rtm.Poke("C53D7729", hexString);

//This address is for Skill Moves Star Rating

EntryPoint.Rtm.Poke("C53D77A8", "04");

EntryPoint.ShowMessageBox("Poke Successful!", "FIFA 13 - RTE", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

EntryPoint.ShowMessageBox(ex.Message,"FIFA 13 - RTE",MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

//When you click the Budget Allocation Poke button

private void budgetPokeButton\_Click(object sender, EventArgs e)

{

try

{

string remaianingBudgetTransfer = uint.Parse(EntryPoint.GetText(remainingBudgetTextBox)).ToString("X");

if ((remaianingBudgetTransfer.Length/2)%2 != 0)

remaianingBudgetTransfer = "0" + remaianingBudgetTransfer;

//Poke that Address CD296230 with the hexString Value - New Save

EntryPoint.Rtm.Poke("CD296230", remaianingBudgetTransfer);

//Alternative - In case the first doesn't work

if (alternateCheckBox.Checked)

{

EntryPoint.Rtm.Poke("CD1C8F90", remaianingBudgetTransfer);

}

EntryPoint.ShowMessageBox("Poke Successful!", "FIFA 13 - RTE", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

EntryPoint.ShowMessageBox(ex.Message, "FIFA 13 - RTE", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

## Coding – MainForm – Using Pointer

Below is the full Main Form code with comments this is as complicated as it gets:

using System;

using System.ComponentModel;

using System.Globalization;

using System.Threading;

using System.Windows.Forms;

using PeekPoker.Interface;

//======================================================

//Author: PureIso

//Description: Gears of War RTE Main Code using Pointer

// This is as complicated as it gets.

//======================================================

namespace GearsOfWar\_RTE

{

public partial class MainForm : Form

{

#region Variables

//This value / hex / pointer will lead us to the values we want to poke

private const string PointerHexString = "576172436865636B706F696E74";

private const string AssultRifle = "1E5761726661726547616D652E576561705F41737361756C745269666C6500";

private const string LocustAssultRifle =

"245761726661726547616D652E576561705F4C6F6375737441737361756C745269666C6500";

private const string Grenade = "1B5761726661726547616D652E576561705F434F47506973746F6C00";

private const string FragGrenade = "1D5761726661726547616D652E576561705F467261674772656E61646500";

private const string CogPistol = "1B5761726661726547616D652E576561705F434F47506973746F6C00";

private const string Shotgun = "195761726661726547616D652E576561705F53686F7467756E00";

private BindingList<SearchResults> \_result;

#endregion

public MainForm()

{

InitializeComponent();

}

private void PokeThread(object a)

{

try

{

//Disable the poke button so user won't click while working

EntryPoint.EnableControl(ammoPokeButton, false);

//Turn the value into a valid Hex String

//X8 0 padding = 00000000

//FF = 000000FF <-- Padding with 00

string value = uint.Parse(valueTextBox.Text).ToString("X8");

//Have a trigger - if you are doing multiple search

//This will prevent you from searching twice

bool found = false;

#region Alternative

if (alternateCheckBox.Checked)

{

//Set Dump offset

EntryPoint.Rtm.DumpOffset = Convert.ToUInt32("C4000000", 16);

//Set Dump Length

EntryPoint.Rtm.DumpLength = Convert.ToUInt32("07FFFFFF", 16);

//Find the Hex from the Offset + The Length

\_result = EntryPoint.Rtm.FindHexOffset(PointerHexString);

//Set teh trigger to true to prevent another search

found = \_result.Count > 0;

Search(value);

}

#endregion

#region Main

if (found)

{

EntryPoint.ShowMessageBox("Poke Successful.", "Gears Of War", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

EntryPoint.Rtm.DumpOffset = Convert.ToUInt32("C8000000", 16);

EntryPoint.Rtm.DumpLength = Convert.ToUInt32("03FFFFFF", 16);

\_result = EntryPoint.Rtm.FindHexOffset(PointerHexString);

found = \_result.Count > 0;

Search(value);

if(found)

EntryPoint.ShowMessageBox("Poke Successful.", "Gears Of War", MessageBoxButtons.OK, MessageBoxIcon.Information);

else

EntryPoint.ShowMessageBox("No Offset Found!", "Gears Of War", MessageBoxButtons.OK, MessageBoxIcon.Warning);

#endregion

}

catch (Exception ex)

{

EntryPoint.ShowMessageBox(ex.Message, "Gears Of War", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

//Reset the progressbar

EntryPoint.UpdateProgressBar(0, 100, 0);

//Enable the Button

EntryPoint.EnableControl(ammoPokeButton, true);

}

}

/// <summary>

/// This is the search function

/// </summary>

/// <param name="value">The value that will be set</param>

private void Search(string value)

{

//If we have a result

if (\_result.Count <= 0) return;

//If all the results we have

foreach (SearchResults results in \_result)

{

//This is specific for gears of war

//We want to set another Dump offset and a small dump length

//We will be modifying the game's checkpoint on the fly as soon as it's stored in memory

EntryPoint.Rtm.DumpOffset = Convert.ToUInt32(results.Offset, 16);

EntryPoint.Rtm.DumpLength = Convert.ToUInt32("0FFF", 16);

//Find the Assult rifle offset

BindingList<SearchResults> weapons = EntryPoint.Rtm.FindHexOffset(AssultRifle);

foreach (SearchResults item in weapons)

{

//We want to Poke the location of the ammo value

//The Ammo value is located at the end of the offset so we:

//item.Offset (turn it to uint) + weapons string length (looking at it a a byte array) / 2)

//Now we have the end of the offset so we poke it with the value

EntryPoint.Rtm.Poke(

uint.Parse(item.Offset, NumberStyles.HexNumber) + (uint) AssultRifle.Length/2, value);

}

weapons = EntryPoint.Rtm.FindHexOffset(LocustAssultRifle);

foreach (SearchResults item in weapons)

{

EntryPoint.Rtm.Poke(

uint.Parse(item.Offset, NumberStyles.HexNumber) + (uint) LocustAssultRifle.Length/2,

value);

}

weapons = EntryPoint.Rtm.FindHexOffset(Grenade);

foreach (SearchResults item in weapons)

{

EntryPoint.Rtm.Poke(

uint.Parse(item.Offset, NumberStyles.HexNumber) + (uint) Grenade.Length/2, value);

}

weapons = EntryPoint.Rtm.FindHexOffset(FragGrenade);

foreach (SearchResults item in weapons)

{

EntryPoint.Rtm.Poke(

uint.Parse(item.Offset, NumberStyles.HexNumber) + (uint) FragGrenade.Length/2, value);

}

weapons = EntryPoint.Rtm.FindHexOffset(CogPistol);

foreach (SearchResults item in weapons)

{

EntryPoint.Rtm.Poke(

uint.Parse(item.Offset, NumberStyles.HexNumber) + (uint) CogPistol.Length/2, value);

}

weapons = EntryPoint.Rtm.FindHexOffset(Shotgun);

foreach (SearchResults item in weapons)

{

EntryPoint.Rtm.Poke(

uint.Parse(item.Offset, NumberStyles.HexNumber) + (uint)Shotgun.Length / 2, value);

}

}

}

private void ammoPokeButton\_Click(object sender, EventArgs e)

{

//Create a thread Pool so application keeps running smoothly

ThreadPool.QueueUserWorkItem(PokeThread);

}

}

}

## Coding - Application Optional Form Plugin

This function is for developers who want to add a function to the Peek Poker application. Below is a code to the entry point and 2 sample screenshot.

using System.Windows.Forms;

using PeekPoker.Interface;

//===============================================

//Author: PureIso

//Description: Converter - Peek Poker Form Addon

//===============================================

namespace ConverterForm

{

//Entry Point inherits from PeekPoker.Interface.AbstractPlugin

public class EntryPoint : AbstractPlugin

{

/// <summary>

/// A Thread Safe way to show messagebox using the codes in

/// PeekPoker

/// </summary>

internal static ShowMessageBoxHandler ShowMessageBox;

/// <summary>

/// A Thread Safe way to update the progressbar using the codes in

/// PeekPoker

/// </summary>

internal static UpdateProgressBarHandler UpdateProgressBar;

/// <summary>

/// A Thread Safe way to set text in a textbox using the codes in

/// PeekPoker

/// </summary>

internal static SetTextBoxTextDelegateHandler SetText;

/// <summary>

/// A Thread Safe way to get text from a textbox using the codes in

/// PeekPoker

/// </summary>

internal static GetTextBoxTextHandler GetText;

/// <summary>

/// A Thread Safe way to enable or disable controls using the codes in

/// PeekPoker

/// </summary>

internal static EnableControlHandler EnableControl;

/// <summary>

/// The real time memory accessor

/// </summary>

internal static RealTimeMemory Rtm;

/// <summary>

/// Constructor

/// </summary>

public EntryPoint()

{

//Application Information

//If you don't fill out any of the information AbstractPlugin will use

//the default which is "Unavailable"

base.ApplicationName = "Converter";

base.Author = "PureIso";

base.Description = "Convert Values";

base.Version = "1.0.0.0";

//Add this for Option Plugin

base.PluginType = PluginType.SelectionOption;

//Get the proper Icon

Converter form = new Converter();

base.Icon = form.Icon;

//Clean up

form.Dispose();

form.Close();

}

/// <summary>

/// Overrides the abstract Display method - if not called an empty form

/// will be loaded

/// </summary>

/// <param name="parent">The PeekPoker will be the parent</param>

public override void Display(Form parent)

{

//Handlers

ShowMessageBox = base.APShowMessageBox;

UpdateProgressBar = base.APUpdateProgressBar;

GetText = base.APGetTextBoxText;

SetText = base.APSetTextBoxText;

EnableControl = base.APEnableControl;

//You can get the RTM being using in PeekPoker or

//you can initialize you own

Rtm = base.APRtm;

Converter form = base.IsMdiChild

? new Converter { MdiParent = parent }

: new Converter();

form.Show();

}

}

}

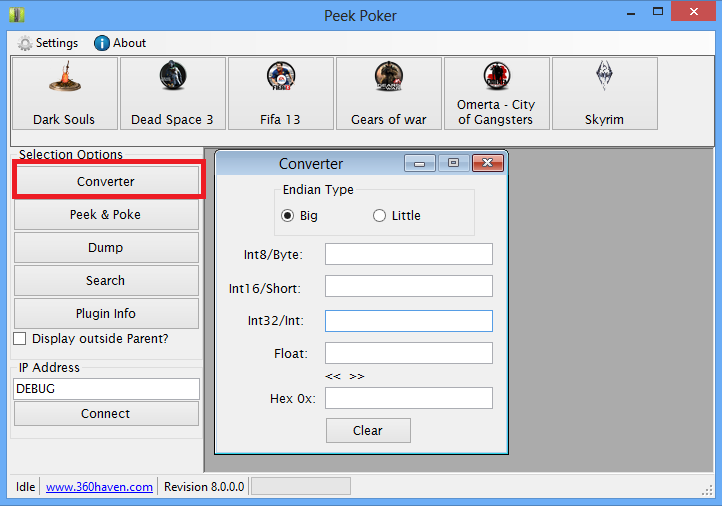


Figure 6: The Converter Plugin (Remember PluginType = PluginType.SelectionOption

# Links

* [www.360haven.com](http://www.360haven.com)
* <http://code.google.com/p/peek-poker/>

# Thanks

* 360Haven
* 8Ball
* ActualmanX
* Cybersam
* Fairchild
* Feudalnate
* Ioritree
* Jappi88
* Magnus Hydra
* Mojobojo
* Nateleroux
* Optantic
* Renegade

Everyone else I cannot remember at this time!