EBOOK

Michael Andre Franiatte

C# Codes for Wiimote to Play PC Games

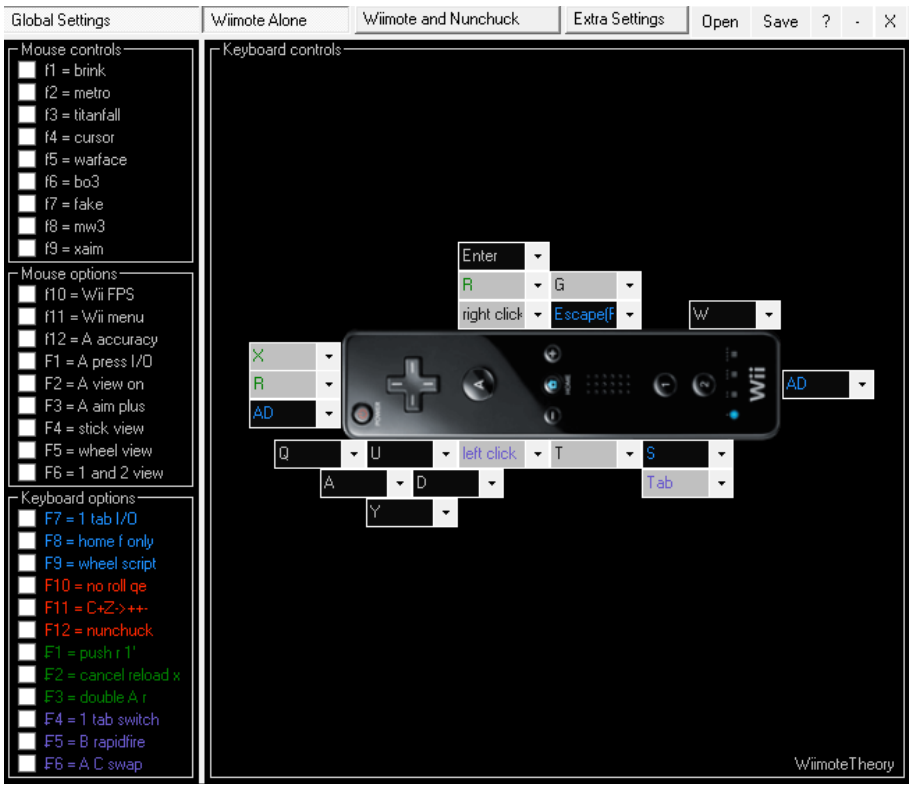
WiimoteTheory.exe

EBOOK

C# Codes for Wiimote to Play PC Games

WiimoteTheory.exe

Michael Franiatte
03/02/2019



The C# codes presented can simulate keyboard and mouse events to play very well PC games using Wiimote/Nunchuck/Sensor bar as a simple program and script. Information about license, EULA and contract for using these following works can be found at https://michaelfraniatte.wordpress.com.

C# Codes for Wiimote to Play PC Games

Michael Franiatte*

Abstract

With these C# codes, Wiimote on PC is the best solution to play games allowing to replace keyboard and mouse, with the same accuracy and more easy to use for a best comfortable experience of gameplay. The codes presented here allow simulating keyboard and mouse events in order to play PC games using Wiimote/Nunchuck/Sensor bar. This paper gives 10 years of works on coding Wiimote and coding keyboard and mouse events to have the best controls never reached by other works on it. This is the perfect solution to play PC games in a beauty manner with all codes to play in all different manner adapted to all games. Wiimote is very competitive with these codes which allow a perfect control without any flaw or lag for all game genres and settings. Some complementary explanations are available in other books of the same author.

Keywords: *gamepads, PC, gameplay, games, codes, Wiimote*

* Author correspondence: michael.franiatte@gmail.com

1. *Mouse Acceleration and Configuration on PC*

For better control with the plugin X-Aim, check for the mouse configuration and acceleration, change otherwise, if the Windows Registry Keys are like this:

```
[HKEY_CURRENT_USER\Control Panel\Mouse]
"SmoothMouseXCurve"=hex:
00,00,00,00,00,00,00,00,
15,6e,00,00,00,00,00,00,
00,40,01,00,00,00,00,00,
29,dc,03,00,00,00,00,00,
00,00,28,00,00,00,00,00
"SmoothMouseYCurve"=hex:
00,00,00,00,00,00,00,00,
b8,5e,01,00,00,00,00,00,
cd,4c,05,00,00,00,00,00,
cd,4c,18,00,00,00,00,00,
00,00,38,02,00,00,00,00
```

2. *Win32 C++ DLL WiimotePairing Codes*

```
using namespace std;
#include "stdafx.h"
#include <windows.h>
#include <bthsdpdef.h>
#include <bthdef.h>
#include <BluetoothAPIs.h>
#include <strsafe.h>
#include <iostream>
using namespace std;
#pragma comment(lib, "Bthprops.lib")
BLUETOOTH_DEVICE_INFO _btdi;
bool wiimotefound = false;
HBLUETOOTH_DEVICE_FIND hFind = NULL;
WCHAR pass[6];
DWORD pcServices = 16;
GUID guids[16];
#pragma warning(disable : 4995)
extern "C"
{
	__declspec(dllexport) bool connect()
	{
		int radio;
		int nRadios = 0;
		HANDLE hRadios[256];
		HBLUETOOTH_RADIO_FIND hFindRadio;
		BLUETOOTH_FIND_RADIO_PARAMS radioParam;
		radioParam.dwSize = sizeof(BLUETOOTH_FIND_RADIO_PARAMS);
		BLUETOOTH_RADIO_INFO radioInfo;
		BLUETOOTH_DEVICE_SEARCH_PARAMS srch;
		radioInfo.dwSize = sizeof(radioInfo);
		BLUETOOTH_DEVICE_INFO btdi;
		_btdi.dwSize = sizeof(_btdi);
		btdi.dwSize = sizeof(btdi);
		srch.dwSize = sizeof(BLUETOOTH_DEVICE_SEARCH_PARAMS);
		hFindRadio = BluetoothFindFirstRadio(&radioParam, &hRadios[nRadios++]);
		while (BluetoothFindNextRadio(hFindRadio, &hRadios[nRadios++]))
		{
			hFindRadio = BluetoothFindFirstRadio(&radioParam,
&hRadios[nRadios++]);
			BluetoothFindRadioClose(hFindRadio);
		}
		for (radio = 0; radio < nRadios; radio++)
		{
			BluetoothGetRadioInfo(hRadios[radio], &radioInfo);
			srch.fReturnAuthenticated = TRUE;
			srch.fReturnRemembered = TRUE;
			srch.fReturnConnected = TRUE;
			srch.fReturnUnknown = TRUE;
```

```
                        srch.fIssueInquiry = TRUE;
                        srch.cTimeoutMultiplier = 2;
                        srch.hRadio = hRadios[radio];
                        if (hFindRadio)
                        {
                                BluetoothGetRadioInfo(hRadios[1], &radioInfo);
                                srch.hRadio = hRadios[1];
                                int nPaired = 0;
                                int numberOfDevices = 2;
                                hFind = BluetoothFindFirstDevice(&srch, &btdi);
                                while (nPaired < numberOfDevices)
                                {
                                        do
                                        {
                                                if ((!wcscmp(btdi.szName, L"Nintendo RVL-WBC-
01") | !wcscmp(btdi.szName, L"Nintendo RVL-CNT-01")) & !wiimotefound)
                                                {
                                                        _btdi = btdi;
                                                        pass[0] = radioInfo.address.rgBytes[0];
                                                        pass[1] = radioInfo.address.rgBytes[1];
                                                        pass[2] = radioInfo.address.rgBytes[2];
                                                        pass[3] = radioInfo.address.rgBytes[3];
                                                        pass[4] = radioInfo.address.rgBytes[4];
                                                        pass[5] = radioInfo.address.rgBytes[5];
                                                        BluetoothAuthenticateDevice(NULL,
hRadios[1], &btdi, pass, 6);

        BluetoothEnumerateInstalledServices(hRadios[1], &btdi, &pcServices, guids);
                                                        BluetoothSetServiceState(hRadios[1],
&btdi, &HumanInterfaceDeviceServiceClass_UUID, BLUETOOTH_SERVICE_ENABLE);
                                                        wiimotefound = true;
                                                }
                                                nPaired++;
                                                Sleep(1);
                                        } while (BluetoothFindNextDevice(hFind, &btdi));
                                        Sleep(1);
                                }
                                if (wiimotefound)
                                        return true;
                        }
                }
                return false;
        }
        __declspec(dllexport) bool disconnect()
        {
                BluetoothRemoveDevice(&_btdi.Address);
                return true;
        }
}
```

3. *Win32 C++ DLL SendInputLibrary Codes*

```
#include "stdafx.h"
#include <windows.h>
INPUT down[1], up[1], downa[1], upa[1], MiceW3[1], Micek[1], Micel[1], Micelf[1],
Micerc[1], Micercf[1], Micemc[1], Micemcf[1], Micewd[1], Micewu[1];
bool downb, upb, downab, upab, MiceW3b, Micekb, Micelb, Micelfb, Micercb, Micercfb,
Micemcb, Micemcfb, Micewdb, Micewub;
int size = sizeof(INPUT);
extern "C"
{
        __declspec(dllexport) void SimulateKeyDown(UINT keyCode, UINT bScan)
        {
                if (!downb)
                {
                        down[0].type = 1;
                        down[0].ki.dwFlags = 0;
```

```
		downb = true;
	}
	down[0].ki.wVk = keyCode;
	down[0].ki.wScan = bScan;
	SendInput(1, down, size);
}
__declspec(dllexport) void SimulateKeyUp(UINT keyCode, UINT bScan)
{
	if (!upb)
	{
		up[0].type = 1;
		up[0].ki.dwFlags = 0x0002;
		upb = true;
	}
	up[0].ki.wVk = keyCode;
	up[0].ki.wScan = bScan;
	SendInput(1, up, size);
}
__declspec(dllexport) void SimulateKeyDownArrows(UINT keyCode, UINT bScan)
{
	if (!downab)
	{
		downa[0].type = 1;
		downab = true;
	}
	downa[0].ki.wVk = keyCode;
	downa[0].ki.wScan = bScan;
	downa[0].ki.dwFlags = 0x0001 | 0x0008;
	SendInput(1, downa, size);
	downa[0].ki.dwFlags = 0;
	SendInput(1, downa, size);
}
__declspec(dllexport) void SimulateKeyUpArrows(UINT keyCode, UINT bScan)
{
	if (!upab)
	{
		upa[0].type = 1;
		upab = true;
	}
	upa[0].ki.wVk = keyCode;
	upa[0].ki.wScan = bScan;
	upa[0].ki.dwFlags = 0x0002 | 0x0001 | 0x0008;
	SendInput(1, upa, size);
	upa[0].ki.dwFlags = 0x0002;
	SendInput(1, upa, size);
}
__declspec(dllexport) void MouseMW3(int x, int y)
{
	if (!MiceW3b)
	{
		MiceW3[0].type = 0;
		MiceW3[0].mi.dwFlags = 0x8001;
		MiceW3b = true;
	}
	MiceW3[0].mi.dx = x;
	MiceW3[0].mi.dy = y;
	SendInput(1, MiceW3, size);
}
__declspec(dllexport) void MouseBrink(int x, int y)
{
	if (!Micekb)
	{
		Micek[0].type = 0;
		Micek[0].mi.dwFlags = 0x0001;
		Micekb = true;
	}
```

```
	Micek[0].mi.dx = x;
	Micek[0].mi.dy = y;
	SendInput(1, Micek, size);
}
__declspec(dllexport) void LeftClick()
{
	if (!Micelb)
	{
		Micel[0].type = 0;
		Micel[0].mi.dwFlags = 0x0002;
		Micelb = true;
	}
	SendInput(1, Micel, size);
}
__declspec(dllexport) void LeftClickF()
{
	if (!Micelfb)
	{
		Micelf[0].type = 0;
		Micelf[0].mi.dwFlags = 0x0004;
		Micelfb = true;
	}
	SendInput(1, Micelf, size);
}
__declspec(dllexport) void RightClick()
{
	if (!Micercb)
	{
		Micerc[0].type = 0;
		Micerc[0].mi.dwFlags = 0x0008;
		Micercb = true;
	}
	SendInput(1, Micerc, size);
}
__declspec(dllexport) void RightClickF()
{
	if (!Micercfb)
	{
		Micercf[0].type = 0;
		Micercf[0].mi.dwFlags = 0x0010;
		Micercfb = true;
	}
	SendInput(1, Micercf, size);
}
__declspec(dllexport) void MiddleClick()
{
	if (!Micemcb)
	{
		Micemc[0].type = 0;
		Micemc[0].mi.dwFlags = 0x0020;
		Micemcb = true;
	}
	SendInput(1, Micemc, size);
}
__declspec(dllexport) void MiddleClickF()
{
	if (!Micemcfb)
	{
		Micemcf[0].type = 0;
		Micemcf[0].mi.dwFlags = 0x0040;
		Micemcfb = true;
	}
	SendInput(1, Micemcf, size);
}
__declspec(dllexport) void WheelDownF()
{
```

```
		if (!Micewdb)
		{
			Micewd[0].type = 0;
			Micewd[0].mi.mouseData = -120;
			Micewd[0].mi.dwFlags = 0x0800;
			Micewdb = true;
		}
		SendInput(1, Micewd, size);
	}
	__declspec(dllexport) void WheelUpF()
	{
		if (!Micewub)
		{
			Micewu[0].type = 0;
			Micewu[0].mi.mouseData = 120;
			Micewu[0].mi.dwFlags = 0x0800;
			Micewub = true;
		}
		SendInput(1, Micewu, size);
	}
}
```

4. Win32 C++ DLL InputSending Codes

```
#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <windows.h>
#include <mmsystem.h>
#include <winioctl.h>
#pragma comment(lib, "winmm.lib")
#pragma comment(lib, "ntdll.lib")
extern "C"
{
	__declspec(dllexport) void MoveMouseTo(int x, int y)
	{
		mouse_event(0x8001, x, y, 0, 0);
	}
	__declspec(dllexport) void MoveMouseBy(int x, int y)
	{
		mouse_event(0x0001, x, y, 0, 0);
	}
	__declspec(dllexport) void SendKey(UINT bVk, UINT bScan)
	{
		keybd_event(bVk, bScan, 0, 0);
	}
	__declspec(dllexport) void SendKeyF(UINT bVk, UINT bScan)
	{
		keybd_event(bVk, bScan, 0x0002, 0);
	}
	__declspec(dllexport) void SendKeyArrows(UINT bVk, UINT bScan)
	{
		keybd_event(bVk, bScan, 0x0001 | 0x0008, 0);
		keybd_event(bVk, bScan, 0, 0);
	}
	__declspec(dllexport) void SendKeyArrowsF(UINT bVk, UINT bScan)
	{
		keybd_event(bVk, bScan, 0x0002 | 0x0001 | 0x0008, 0);
		keybd_event(bVk, bScan, 0x0002, 0);
	}
	__declspec(dllexport) void SendMouseEventButtonLeft()
	{
		mouse_event(0x0002, 0, 0, 0, 0);
	}
	__declspec(dllexport) void SendMouseEventButtonLeftF()
	{
```

```
		mouse_event(0x0004, 0, 0, 0, 0);
	}
	__declspec(dllexport) void SendMouseEventButtonRight()
	{
		mouse_event(0x0008, 0, 0, 0, 0);
	}
	__declspec(dllexport) void SendMouseEventButtonRightF()
	{
		mouse_event(0x0010, 0, 0, 0, 0);
	}
	__declspec(dllexport) void SendMouseEventButtonMiddle()
	{
		mouse_event(0x0020, 0, 0, 0, 0);
	}
	__declspec(dllexport) void SendMouseEventButtonMiddleF()
	{
		mouse_event(0x0040, 0, 0, 0, 0);
	}
	__declspec(dllexport) void SendMouseEventButtonWheelUp()
	{
		mouse_event(0x0800, 0, 0, 120, 0);
	}
	__declspec(dllexport) void SendMouseEventButtonWheelDown()
	{
		mouse_event(0x0800, 0, 0, -120, 0);
	}
}
```

5. *C# Windows Form WiimoteTheory Codes*

```
using System;
using System.Windows.Forms;
using System.Drawing;
using Microsoft.Win32.SafeHandles;
using System.Runtime.InteropServices;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using System.IO;
using System.Diagnostics;
using System.Collections;
using System.ComponentModel;
using System.Text.RegularExpressions;
namespace WiimoteTheory
{
    public partial class Form1 : Form
    {
        public static double index = 0, REGISTER_IR = 0x04b00030, REGISTER_EXTENSION_INIT_1
= 0x04a400f0, REGISTER_EXTENSION_INIT_2 = 0x04a400fb, REGISTER_EXTENSION_TYPE = 0x04a400fa,
REGISTER_EXTENSION_CALIBRATION = 0x04a40020, REGISTER_MOTIONPLUS_INIT = 0x04a600fe, WidthS,
HeightS, keys123, keys456, keysEnterTab, irx2e, iry2e, irx3e, iry3e, irxe, irye, irx, iry,
irxc, iryc, iryn, irxpp, irypp, mWSIRSensorsXcam, mWSIRSensorsYcam, mWSIRSensors0X,
mWSIRSensors0Y, mWSIRSensors1X, mWSIRSensors1Y, mousexi, mouseyi, irxev, iryev, irxemin,
iryemin, irxemax, iryemax, iryppv, iryppmin, iryppmax, mousexn, mouseyn, mousexpn,
mouseypn, mousexbn, mouseybn, mousexp, mouseyp, mousexpm, mouseypm, mousexpp, mouseypp,
Rand1swps, Rand1swms, Rand2swyps, Rand2swyms, Rand2swps, Rand2swms, mWSRawValuesX,
mWSRawValuesY, mWSRawValuesZ, mWSNunchuckStateRawJoystickX, mWSNunchuckStateRawJoystickY,
mWSIRSensors0Xcam, mWSIRSensors1Xcam, mWSIRSensors0Ycam, mWSIRSensors1Ycam,
mWSNunchuckStateRawValuesX, mWSNunchuckStateRawValuesY, mWSNunchuckStateRawValuesZ,
signirx, signiry, absirx, absiry, RollWiimoteAngle, _rollwiimoteanglechanged, aimpluscount,
centercursorposcount, stickviewxinit, stickviewyinit, readingfilecount,
brinktitanfalltimecount, bo3timecount, calibrationinit, mWSRawValuesXcam, mWSRawValuesYcam,
mWSRawValuesZcam, mWSNunchuckStateRawJoystickXcam, mWSNunchuckStateRawJoystickYcam,
MyAngle, Acceleration, Breaking, pushrcount, cancelreloadcount, doubleacount,
doubleaacount, randtabwiimote1switch, oneswitchcount, doubleapushrcount, rapidfirecount,
keystodown, pushtodown, norecoilcount, antistcount, switchtowheelwithwheelscriptcount,
```

```
switchtowheelwithoutwheelscriptcount, switchtowheelcount, switchtostickcount,
ticktimecount, watchK = 50, watchK1 = 2, watchK2 = 0, watchM = 50, watchM1 = 2, watchM2 =
0, rightpowerslidecount, Upview, Downview, Rightview, Leftview, mWSIR0notfound = 0,
signchangewheelY1, signchangewheelY2;
        public static bool bool2swyps, bool2swyms, bool2swps, bool2swms, bool1swps,
bool1swms, mWSIR1foundcam, mWSIR0foundcam, mWSIR1found, mWSIR0found, mWSIRswitch,
mWSButtonStateA, mWSButtonStateB, mWSButtonStateMinus, mWSButtonStateHome,
mWSButtonStatePlus, mWSButtonStateOne, mWSButtonStateTwo, mWSButtonStateUp,
mWSButtonStateDown, mWSButtonStateLeft, mWSButtonStateRight, mWSButtonStateUpcam,
mWSButtonStateDowncam, mWSButtonStateLeftcam, mWSButtonStateRightcam, mWSNunchuckStateC,
mWSNunchuckStateZ, stickviewswitch, _Rollwiimoteanglechanged, runningoff, _getstate,
_Getstate, getstate, Getstate, mWSButtonStateAio, mWSButtonStateOnecam,
mWSButtonStateTwocam, cancelreloadbool, doubleabool, oneswitchbool, mWSNunchuckStateCio,
mWSNunchuckStateZio, nunchucktodown2released, mWSButtonStateBcam, foraorcison,
enableautoloadoflastfile, notpressing1and2, reconfiguration, randA, randC, randZ,
readingfile, _value, signchangewheelY, lockchangefeaturesandoptions;
        public static string VID = "057e", PID1 = "0330", PID2 = "0306", path;
        public static byte[] buff = new byte[] { 0x55 }, mBuff = new byte[22], aBuffer =
new byte[22], bBuffer = new byte[22];
        public static byte Type = 0x12, IR = 0x13, WriteMemory = 0x16, ReadMemory = 0x16,
IRExtensionAccel = 0x37;
        public static Guid guid = new System.Guid();
        public static uint hDevInfo, CurrentResolution = 0;
        public static SafeFileHandle _hFile, handleRight;
        public static bool[] _Valuechanged = new bool[92], _valuechanged = new bool[92],
_Value = new bool[92];
        public static Dictionary<string, bool> Fbool = new Dictionary<string, bool>(40),
Abool = new Dictionary<string, bool>(40);
        public static Dictionary<string, double> Fvar = new Dictionary<string, double>(40);
        public static Dictionary<string, Point2D> actionassign = new Dictionary<string,
Point2D>(40);
        public static Dictionary<string, string> action = new Dictionary<string,
string>(40);
        public static List<double> valListXn = new List<double>(), valListYn = new
List<double>();
        public static BackgroundWorker backgroundWorkerS = new BackgroundWorker();
        public static Task taskD, taskM, taskK;
        public static ThreadStart threadstart;
        public static Thread thread;
        private static Stopwatch diffM = new Stopwatch(), diffK = new Stopwatch();
        private const string vendor_id = "57e", vendor_id_ = "057e", product_r1 = "0330",
product_r2 = "0306";
        private static System.IO.FileStream mStream;
        public static double Rollwiimoteanglechanged
        {
            get { return _rollwiimoteanglechanged; }
            set
            {
                if (_rollwiimoteanglechanged != value)
                    _Rollwiimoteanglechanged = true;
                else
                    _Rollwiimoteanglechanged = false;
                _rollwiimoteanglechanged = value;
            }
        }
        public bool this[int i]
        {
            get { return _valuechanged[i]; }
            set
            {
                if (_valuechanged[i] != value)
                    _Valuechanged[i] = true;
                else
                    _Valuechanged[i] = false;
                _valuechanged[i] = value;
                if (_Valuechanged[i] & value)
```

```
            _Value[i] = true;
        if (_Valuechanged[i] & !value)
            _Value[i] = false;
    }
}
public static ushort VK_Tab = (ushort)(0x09);
public static ushort VK_Return = (ushort)(0x0D);
public static ushort VK_LEFT = (ushort)(0x25);
public static ushort VK_UP = (ushort)(0x26);
public static ushort VK_RIGHT = (ushort)(0x27);
public static ushort VK_DOWN = (ushort)(0x28);
public static ushort VK_0 = (ushort)(0x30);
public static ushort VK_1 = (ushort)(0x31);
public static ushort VK_2 = (ushort)(0x32);
public static ushort VK_3 = (ushort)(0x33);
public static ushort VK_4 = (ushort)(0x34);
public static ushort VK_5 = (ushort)(0x35);
public static ushort VK_6 = (ushort)(0x36);
public static ushort VK_7 = (ushort)(0x37);
public static ushort VK_8 = (ushort)(0x38);
public static ushort VK_9 = (ushort)(0x39);
public static ushort VK_A = (ushort)(0x41);
public static ushort VK_D = (ushort)(0x44);
public static ushort VK_Q = (ushort)(0x51);
public static ushort VK_R = (ushort)(0x52);
public static ushort VK_S = (ushort)(0x53);
public static ushort VK_W = (ushort)(0x57);
public static ushort VK_Z = (ushort)(0x5A);
public static ushort S_Tab = (ushort)MapVirtualKey(0x09, 0);
public static ushort S_Return = (ushort)MapVirtualKey(0x0D, 0);
public static ushort S_LEFT = (ushort)MapVirtualKey(0x25, 0);
public static ushort S_UP = (ushort)MapVirtualKey(0x26, 0);
public static ushort S_RIGHT = (ushort)MapVirtualKey(0x27, 0);
public static ushort S_DOWN = (ushort)MapVirtualKey(0x28, 0);
public static ushort S_0 = (ushort)MapVirtualKey(0x30, 0);
public static ushort S_1 = (ushort)MapVirtualKey(0x31, 0);
public static ushort S_2 = (ushort)MapVirtualKey(0x32, 0);
public static ushort S_3 = (ushort)MapVirtualKey(0x33, 0);
public static ushort S_4 = (ushort)MapVirtualKey(0x34, 0);
public static ushort S_5 = (ushort)MapVirtualKey(0x35, 0);
public static ushort S_6 = (ushort)MapVirtualKey(0x36, 0);
public static ushort S_7 = (ushort)MapVirtualKey(0x37, 0);
public static ushort S_8 = (ushort)MapVirtualKey(0x38, 0);
public static ushort S_9 = (ushort)MapVirtualKey(0x39, 0);
public static ushort S_A = (ushort)MapVirtualKey(0x41, 0);
public static ushort S_D = (ushort)MapVirtualKey(0x44, 0);
public static ushort S_Q = (ushort)MapVirtualKey(0x51, 0);
public static ushort S_R = (ushort)MapVirtualKey(0x52, 0);
public static ushort S_S = (ushort)MapVirtualKey(0x53, 0);
public static ushort S_W = (ushort)MapVirtualKey(0x57, 0);
public static ushort S_Z = (ushort)MapVirtualKey(0x5A, 0);
public static Point2D point2d(UInt16 x, UInt16 y)
{
    Point2D point;
    point.X = x;
    point.Y = y;
    return point;
}
public struct Point2D
{
    public UInt16 X, Y;
}
public static Point2D assign(string keys)
{
    uint vkcode = 0;
    uint bscancode = 0;
```

```
switch (keys)
{
    case " ":
        vkcode = 0x777;
        bscancode = 0x777;
        break;
    case "right click":
        vkcode = 0x888;
        bscancode = 0x888;
        break;
    case "left click":
        vkcode = 0x999;
        bscancode = 0x999;
        break;
    case "middle click":
        vkcode = 0x666;
        bscancode = 0x666;
        break;
    case "wheel up":
        vkcode = 0x444;
        bscancode = 0x444;
        break;
    case "wheel down":
        vkcode = 0x333;
        bscancode = 0x333;
        break;
    case "0-4":
        vkcode = 0x111;
        bscancode = 0x111;
        break;
    case "5-9":
        vkcode = 0x222;
        bscancode = 0x222;
        break;
    case "Enter/Tab":
        vkcode = 0x555;
        bscancode = 0x555;
        break;
    case "left":
        vkcode = 0x25;
        break;
    case "right":
        vkcode = 0x27;
        break;
    case "up":
        vkcode = 0x26;
        break;
    case "down":
        vkcode = 0x28;
        break;
    case "W":
        vkcode = 0x57;
        break;
    case "A":
        vkcode = 0x41;
        break;
    case "Z":
        vkcode = 0x5A;
        break;
    case "Q":
        vkcode = 0x51;
        break;
    case "S":
        vkcode = 0x53;
        break;
    case "D":
```

```
        vkcode = 0x44;
        break;
    case "E":
        vkcode = 0x45;
        break;
    case "R":
        vkcode = 0x52;
        break;
    case "F":
        vkcode = 0x46;
        break;
    case "J":
        vkcode = 0x4A;
        break;
    case "K":
        vkcode = 0x4B;
        break;
    case "B":
        vkcode = 0x42;
        break;
    case "N":
        vkcode = 0x4E;
        break;
    case "X":
        vkcode = 0x58;
        break;
    case "Y":
        vkcode = 0x59;
        break;
    case "U":
        vkcode = 0x55;
        break;
    case "C":
        vkcode = 0x43;
        break;
    case "T":
        vkcode = 0x54;
        break;
    case "G":
        vkcode = 0x47;
        break;
    case "H":
        vkcode = 0x48;
        break;
    case "V":
        vkcode = 0x56;
        break;
    case "Tab":
        vkcode = 0x09;
        break;
    case "Space":
        vkcode = 0x20;
        break;
    case "Enter":
        vkcode = 0x0D;
        break;
    case "Shift":
        vkcode = 0x10;
        break;
    case "Control":
        vkcode = 0x11;
        break;
    case "Escape":
        vkcode = 0x1B;
        break;
    case "L":
```

```
        vkcode = 0x4C;
        break;
    case "M":
        vkcode = 0x4D;
        break;
    case "P":
        vkcode = 0x50;
        break;
    case "O":
        vkcode = 0x4F;
        break;
    case "I":
        vkcode = 0x49;
        break;
    case "Apostrophe":
        vkcode = 0xDE;
        break;
    case "Back":
        vkcode = 0x08;
        break;
    case "0":
        vkcode = 0x30;
        break;
    case "1":
        vkcode = 0x31;
        break;
    case "2":
        vkcode = 0x32;
        break;
    case "3":
        vkcode = 0x33;
        break;
    case "4":
        vkcode = 0x34;
        break;
    case "5":
        vkcode = 0x35;
        break;
    case "6":
        vkcode = 0x36;
        break;
    case "7":
        vkcode = 0x37;
        break;
    case "8":
        vkcode = 0x38;
        break;
    case "9":
        vkcode = 0x39;
        break;
    case "Alt":
        vkcode = 0x12;
        break;
    case "F1":
        vkcode = 0x70;
        break;
    case "F2":
        vkcode = 0x71;
        break;
    case "F3":
        vkcode = 0x72;
        break;
    case "F4":
        vkcode = 0x73;
        break;
    case "F5":
```

```
                    vkcode = 0x74;
                    break;
                case "F6":
                    vkcode = 0x75;
                    break;
                case "F7":
                    vkcode = 0x76;
                    break;
                case "F8":
                    vkcode = 0x77;
                    break;
                case "F9":
                    vkcode = 0x78;
                    break;
                case "F10":
                    vkcode = 0x79;
                    break;
                case "F11":
                    vkcode = 0x7A;
                    break;
                case "F12":
                    vkcode = 0x7B;
                    break;
                case "LControl":
                    vkcode = 0xA2;
                    break;
                case "RControl":
                    vkcode = 0xA3;
                    break;
                case "LShift":
                    vkcode = 0xA0;
                    break;
                case "RShift":
                    vkcode = 0xA1;
                    break;
                case "Capslock":
                    vkcode = 0x14;
                    break;
            }
            if (vkcode != 0x111 & vkcode != 0x222 & vkcode != 0x333 & vkcode != 0x444 &
vkcode != 0x555 & vkcode != 0x666 & vkcode != 0x777 & vkcode != 0x888 & vkcode != 0x999)
                bscancode = MapVirtualKey(vkcode, 0);
            return point2d((UInt16)vkcode, (UInt16)bscancode);
        }
        [DllImport("WiimotePairing.dll", EntryPoint = "connect")]
        private static unsafe extern bool connect();
        [DllImport("WiimotePairing.dll", EntryPoint = "disconnect")]
        private static unsafe extern bool disconnect();
        [DllImport("hid.dll")]
        public static unsafe extern void HidD_GetHidGuid(out Guid gHid);
        [DllImport("hid.dll")]
        public extern unsafe static bool HidD_SetOutputReport(IntPtr HidDeviceObject,
byte[] lpReportBuffer, uint ReportBufferLength);
        [DllImport("setupapi.dll")]
        public static unsafe extern IntPtr SetupDiGetClassDevs(ref Guid ClassGuid, string
Enumerator, IntPtr hwndParent, UInt32 Flags);
        [DllImport("setupapi.dll")]
        public static unsafe extern Boolean SetupDiEnumDeviceInterfaces(IntPtr hDevInfo,
IntPtr devInvo, ref Guid interfaceClassGuid, Int32 memberIndex, ref
SP_DEVICE_INTERFACE_DATA deviceInterfaceData);
        [DllImport("setupapi.dll")]
        public static unsafe extern Boolean SetupDiGetDeviceInterfaceDetail(IntPtr
hDevInfo, ref SP_DEVICE_INTERFACE_DATA deviceInterfaceData, IntPtr
deviceInterfaceDetailData, UInt32 deviceInterfaceDetailDataSize, out UInt32 requiredSize,
IntPtr deviceInfoData);
        [DllImport("setupapi.dll")]
```

```
        public static unsafe extern Boolean SetupDiGetDeviceInterfaceDetail(IntPtr
hDevInfo, ref SP_DEVICE_INTERFACE_DATA deviceInterfaceData, ref
SP_DEVICE_INTERFACE_DETAIL_DATA deviceInterfaceDetailData, UInt32
deviceInterfaceDetailDataSize, out UInt32 requiredSize, IntPtr deviceInfoData);
        [DllImport("Kernel32.dll")]
        public static unsafe extern SafeFileHandle CreateFile(string fileName,
[MarshalAs(UnmanagedType.U4)] FileAccess fileAccess, [MarshalAs(UnmanagedType.U4)]
FileShare fileShare, IntPtr securityAttributes, [MarshalAs(UnmanagedType.U4)] FileMode
creationDisposition, [MarshalAs(UnmanagedType.U4)] uint flags, IntPtr template);
        [DllImport("user32.dll")]
        public static extern bool GetAsyncKeyState(System.Windows.Forms.Keys vKey);
        [DllImport("user32.dll")]
        public static extern void SetPhysicalCursorPos(int X, int Y);
        [DllImport("user32.dll")]
        public static extern void SetCaretPos(int X, int Y);
        [DllImport("user32.dll")]
        public static extern void SetCursorPos(int X, int Y);
        [DllImport("winmm.dll", EntryPoint = "timeBeginPeriod")]
        public static extern uint TimeBeginPeriod(uint ms);
        [DllImport("winmm.dll", EntryPoint = "timeEndPeriod")]
        public static extern uint TimeEndPeriod(uint ms);
        [DllImport("ntdll.dll", EntryPoint = "NtSetTimerResolution")]
        public static extern void NtSetTimerResolution(uint DesiredResolution, bool
SetResolution, ref uint CurrentResolution);
        [DllImport("system32/user32.dll")]
        public static extern uint MapVirtualKey(uint uCode, uint uMapType);
        [DllImport("InputSending.dll", EntryPoint = "MoveMouseTo", CallingConvention =
CallingConvention.Cdecl)]
        public static extern void MoveMouseTo(int x, int y);
        [DllImport("InputSending.dll", EntryPoint = "MoveMouseBy", CallingConvention =
CallingConvention.Cdecl)]
        public static extern void MoveMouseBy(int x, int y);
        [DllImport("InputSending.dll", EntryPoint = "SendKey", CallingConvention =
CallingConvention.Cdecl)]
        public static extern void SendKey(UInt16 bVk, UInt16 bScan);
        [DllImport("InputSending.dll", EntryPoint = "SendKeyF", CallingConvention =
CallingConvention.Cdecl)]
        public static extern void SendKeyF(UInt16 bVk, UInt16 bScan);
        [DllImport("InputSending.dll", EntryPoint = "SendKeyArrows", CallingConvention =
CallingConvention.Cdecl)]
        public static extern void SendKeyArrows(UInt16 bVk, UInt16 bScan);
        [DllImport("InputSending.dll", EntryPoint = "SendKeyArrowsF", CallingConvention =
CallingConvention.Cdecl)]
        public static extern void SendKeyArrowsF(UInt16 bVk, UInt16 bScan);
        [DllImport("InputSending.dll", EntryPoint = "SendMouseEventButtonLeft",
CallingConvention = CallingConvention.Cdecl)]
        public static extern void SendMouseEventButtonLeft();
        [DllImport("InputSending.dll", EntryPoint = "SendMouseEventButtonLeftF",
CallingConvention = CallingConvention.Cdecl)]
        public static extern void SendMouseEventButtonLeftF();
        [DllImport("InputSending.dll", EntryPoint = "SendMouseEventButtonRight",
CallingConvention = CallingConvention.Cdecl)]
        public static extern void SendMouseEventButtonRight();
        [DllImport("InputSending.dll", EntryPoint = "SendMouseEventButtonRightF",
CallingConvention = CallingConvention.Cdecl)]
        public static extern void SendMouseEventButtonRightF();
        [DllImport("InputSending.dll", EntryPoint = "SendMouseEventButtonMiddle",
CallingConvention = CallingConvention.Cdecl)]
        public static extern void SendMouseEventButtonMiddle();
        [DllImport("InputSending.dll", EntryPoint = "SendMouseEventButtonMiddleF",
CallingConvention = CallingConvention.Cdecl)]
        public static extern void SendMouseEventButtonMiddleF();
        [DllImport("InputSending.dll", EntryPoint = "SendMouseEventButtonWheelUp",
CallingConvention = CallingConvention.Cdecl)]
        public static extern void SendMouseEventButtonWheelUp();
```

```
        [DllImport("InputSending.dll", EntryPoint = "SendMouseEventButtonWheelDown",
CallingConvention = CallingConvention.Cdecl)]
        public static extern void SendMouseEventButtonWheelDown();
        [DllImport("SendInputLibrary.dll", EntryPoint = "SimulateKeyDown",
CallingConvention = CallingConvention.Cdecl)]
        public static extern void SimulateKeyDown(UInt16 keyCode, UInt16 bScan);
        [DllImport("SendInputLibrary.dll", EntryPoint = "SimulateKeyUp", CallingConvention
= CallingConvention.Cdecl)]
        public static extern void SimulateKeyUp(UInt16 keyCode, UInt16 bScan);
        [DllImport("SendInputLibrary.dll", EntryPoint = "SimulateKeyDownArrows",
CallingConvention = CallingConvention.Cdecl)]
        public static extern void SimulateKeyDownArrows(UInt16 keyCode, UInt16 bScan);
        [DllImport("SendInputLibrary.dll", EntryPoint = "SimulateKeyUpArrows",
CallingConvention = CallingConvention.Cdecl)]
        public static extern void SimulateKeyUpArrows(UInt16 keyCode, UInt16 bScan);
        [DllImport("SendInputLibrary.dll", EntryPoint = "MouseMW3", CallingConvention =
CallingConvention.Cdecl)]
        public static extern void MouseMW3(int x, int y);
        [DllImport("SendInputLibrary.dll", EntryPoint = "MouseBrink", CallingConvention =
CallingConvention.Cdecl)]
        public static extern void MouseBrink(int x, int y);
        [DllImport("SendInputLibrary.dll", EntryPoint = "LeftClick", CallingConvention =
CallingConvention.Cdecl)]
        public static extern void LeftClick();
        [DllImport("SendInputLibrary.dll", EntryPoint = "LeftClickF", CallingConvention =
CallingConvention.Cdecl)]
        public static extern void LeftClickF();
        [DllImport("SendInputLibrary.dll", EntryPoint = "RightClick", CallingConvention =
CallingConvention.Cdecl)]
        public static extern void RightClick();
        [DllImport("SendInputLibrary.dll", EntryPoint = "RightClickF", CallingConvention =
CallingConvention.Cdecl)]
        public static extern void RightClickF();
        [DllImport("SendInputLibrary.dll", EntryPoint = "MiddleClick", CallingConvention =
CallingConvention.Cdecl)]
        public static extern void MiddleClick();
        [DllImport("SendInputLibrary.dll", EntryPoint = "MiddleClickF", CallingConvention =
CallingConvention.Cdecl)]
        public static extern void MiddleClickF();
        [DllImport("SendInputLibrary.dll", EntryPoint = "WheelDownF", CallingConvention =
CallingConvention.Cdecl)]
        public static extern void WheelDownF();
        [DllImport("SendInputLibrary.dll", EntryPoint = "WheelUpF", CallingConvention =
CallingConvention.Cdecl)]
        public static extern void WheelUpF();
        public static void doMouseMW3(int x, int y)
        {
            if (Fbool["//driver mouse"])
                MoveMouseTo(x, y);
            else
                MouseMW3(x, y);
        }
        public static void doMouseBrink(int x, int y)
        {
            if (Fbool["//driver mouse"])
                MoveMouseBy(x, y);
            else
                MouseBrink(x, y);
        }
        public static void doSimulateKeyDown(UInt16 keyCode, UInt16 bScan)
        {
            if (Fbool["//driver keyboard"])
                SendKey(keyCode, bScan);
            else
                SimulateKeyDown(keyCode, bScan);
        }
```

```
public static void doSimulateKeyUp(UInt16 keyCode, UInt16 bScan)
{
    if (Fbool["//driver keyboard"])
        SendKeyF(keyCode, bScan);
    else
        SimulateKeyUp(keyCode, bScan);
}
public static void doSimulateKeyDownArrows(UInt16 keyCode, UInt16 bScan)
{
    if (Fbool["//driver keyboard"])
        SendKeyArrows(keyCode, bScan);
    else
        SimulateKeyDownArrows(keyCode, bScan);
}
public static void doSimulateKeyUpArrows(UInt16 keyCode, UInt16 bScan)
{
    if (Fbool["//driver keyboard"])
        SendKeyArrowsF(keyCode, bScan);
    else
        SimulateKeyUpArrows(keyCode, bScan);
}
public static void doLeftClick()
{
    if (Fbool["//driver mouse"])
        SendMouseEventButtonLeft();
    else
        LeftClick();
}
public static void doLeftClickF()
{
    if (Fbool["//driver mouse"])
        SendMouseEventButtonLeftF();
    else
        LeftClickF();
}
public static void doRightClick()
{
    if (Fbool["//driver mouse"])
        SendMouseEventButtonRight();
    else
        RightClick();
}
public static void doRightClickF()
{
    if (Fbool["//driver mouse"])
        SendMouseEventButtonRightF();
    else
        RightClickF();
}
public static void doMiddleClick()
{
    if (Fbool["//driver mouse"])
        SendMouseEventButtonMiddle();
    else
        MiddleClick();
}
public static void doMiddleClickF()
{
    if (Fbool["//driver mouse"])
        SendMouseEventButtonMiddleF();
    else
        MiddleClickF();
}
public static void doWheelDownF()
{
    if (Fbool["//driver mouse"])
```

```
                SendMouseEventButtonWheelDown();
            else
                WheelDownF();
        }
        public static void doWheelUpF()
        {
            if (Fbool["//driver mouse"])
                SendMouseEventButtonWheelUp();
            else
                WheelUpF();
        }
        public enum EFileAttributes : uint
        {
            Overlapped = 0x40000000,
            Normal = 0x80
        };
        public struct SP_DEVICE_INTERFACE_DATA
        {
            public int cbSize;
            public Guid InterfaceClassGuid;
            public int Flags;
            public IntPtr RESERVED;
        }
        public struct SP_DEVICE_INTERFACE_DETAIL_DATA
        {
            public UInt32 cbSize;

[System.Runtime.InteropServices.MarshalAs(System.Runtime.InteropServices.UnmanagedType.ByVa
lTStr, SizeConst = 256)]
            public string DevicePath;
        }
        private bool ScanRight()
        {
            int index = 0;
            System.Guid guid;
            HidD_GetHidGuid(out guid);
            System.IntPtr hDevInfo = SetupDiGetClassDevs(ref guid, null, new
System.IntPtr(), 0x00000010);
            SP_DEVICE_INTERFACE_DATA diData = new SP_DEVICE_INTERFACE_DATA();
            diData.cbSize = System.Runtime.InteropServices.Marshal.SizeOf(diData);
            while (SetupDiEnumDeviceInterfaces(hDevInfo, new System.IntPtr(), ref guid,
index, ref diData))
            {
                System.UInt32 size;
                SetupDiGetDeviceInterfaceDetail(hDevInfo, ref diData, new System.IntPtr(),
0, out size, new System.IntPtr());
                SP_DEVICE_INTERFACE_DETAIL_DATA diDetail = new
SP_DEVICE_INTERFACE_DETAIL_DATA();
                diDetail.cbSize = 5;
                if (SetupDiGetDeviceInterfaceDetail(hDevInfo, ref diData, ref diDetail,
size, out size, new System.IntPtr()))
                {
                    if ((diDetail.DevicePath.Contains(vendor_id) |
diDetail.DevicePath.Contains(vendor_id_)) & (diDetail.DevicePath.Contains(product_r1) |
diDetail.DevicePath.Contains(product_r2)))
                    {
                        path = diDetail.DevicePath;
                        WiimoteFound(path);
                        WiimoteFound(path);
                        WiimoteFound(path);
                        return true;
                    }
                }
                index++;
            }
            return false;
```

```
        }
        public void WiimoteFound(string path)
        {
            SafeFileHandle handle = null;
            do
            {
                handle = CreateFile(path, FileAccess.ReadWrite, FileShare.ReadWrite,
IntPtr.Zero, FileMode.Open, (uint)EFileAttributes.Overlapped, IntPtr.Zero);
                WriteData(handle, IR, (int)REGISTER_IR, new byte[] { 0x08 }, 1);
                WriteData(handle, Type, (int)REGISTER_EXTENSION_INIT_1, new byte[] { 0x55
}, 1);
                WriteData(handle, Type, (int)REGISTER_EXTENSION_INIT_2, new byte[] { 0x00
}, 1);
                WriteData(handle, Type, (int)REGISTER_MOTIONPLUS_INIT, new byte[] { 0x04 },
1);
                ReadData(handle, 0x0016, 7);
                ReadData(handle, (int)REGISTER_EXTENSION_TYPE, 6);
                ReadData(handle, (int)REGISTER_EXTENSION_CALIBRATION, 16);
                ReadData(handle, (int)REGISTER_EXTENSION_CALIBRATION, 32);
            }
            while (handle.IsInvalid);
            mStream = new System.IO.FileStream(handle, System.IO.FileAccess.ReadWrite, 22,
true);
        }
        public static void ReadData(SafeFileHandle _hFile, int address, short size)
        {
            mBuff[0] = (byte)ReadMemory;
            mBuff[1] = (byte)((address & 0xff000000) >> 24);
            mBuff[2] = (byte)((address & 0x00ff0000) >> 16);
            mBuff[3] = (byte)((address & 0x0000ff00) >> 8);
            mBuff[4] = (byte)(address & 0x000000ff);
            mBuff[5] = (byte)((size & 0xff00) >> 8);
            mBuff[6] = (byte)(size & 0xff);
            HidD_SetOutputReport(_hFile.DangerousGetHandle(), mBuff, 22);
        }
        public static void WriteData(SafeFileHandle _hFile, byte mbuff, int address, byte[]
buff, short size)
        {
            mBuff[0] = (byte)mbuff;
            mBuff[1] = (byte)(0x04);
            mBuff[2] = (byte)IRExtensionAccel;
            Array.Copy(buff, 0, mBuff, 3, 1);
            HidD_SetOutputReport(_hFile.DangerousGetHandle(), mBuff, 22);
            mBuff[0] = (byte)WriteMemory;
            mBuff[1] = (byte)(((address & 0xff000000) >> 24));
            mBuff[2] = (byte)((address & 0x00ff0000) >> 16);
            mBuff[3] = (byte)((address & 0x0000ff00) >> 8);
            mBuff[4] = (byte)((address & 0x000000ff) >> 0);
            mBuff[5] = (byte)size;
            Array.Copy(buff, 0, mBuff, 6, 1);
            HidD_SetOutputReport(_hFile.DangerousGetHandle(), mBuff, 22);
        }
        private void Wiimote_thrD()
        {
            for (; ; )
            {
                if (runningoff)
                    return;
                try
                {
                    mStream.Read(aBuffer, 0, 22);
                    bBuffer = aBuffer;
                    readingfile = true;
                }
                catch { }
            }
```

```
        }
        public void miceevent(UInt16 micetypeeventX, UInt16 micetypeeventY)
        {
            if (micetypeeventX == 0x888)
            {
                doRightClick();
                return;
            }
            if (micetypeeventX == 0x999)
            {
                doLeftClick();
                return;
            }
            if (micetypeeventX == 0x666)
            {
                doMiddleClick();
                return;
            }
            if (micetypeeventX == 0x444)
            {
                doWheelUpF();
                return;
            }
            if (micetypeeventX == 0x333)
            {
                doWheelDownF();
                return;
            }
            if (micetypeeventX == 0x25 | micetypeeventX == 0x26 | micetypeeventX == 0x27 |
micetypeeventX == 0x28)
            {
                doSimulateKeyDownArrows(micetypeeventX, micetypeeventY);
                return;
            }
            if (micetypeeventX == 0x111)
            {
                if (keys123 == 0)
                    doSimulateKeyDown(VK_0, S_0);
                if (keys123 == 1)
                    doSimulateKeyDown(VK_1, S_1);
                if (keys123 == 2)
                    doSimulateKeyDown(VK_2, S_2);
                if (keys123 == 3)
                    doSimulateKeyDown(VK_3, S_3);
                if (keys123 == 4)
                    doSimulateKeyDown(VK_4, S_4);
                return;
            }
            if (micetypeeventX == 0x222)
            {
                if (keys456 == 0)
                    doSimulateKeyDown(VK_5, S_5);
                if (keys456 == 1)
                    doSimulateKeyDown(VK_6, S_6);
                if (keys456 == 2)
                    doSimulateKeyDown(VK_7, S_7);
                if (keys456 == 3)
                    doSimulateKeyDown(VK_8, S_8);
                if (keys456 == 4)
                    doSimulateKeyDown(VK_9, S_9);
                return;
            }
            if (micetypeeventX == 0x555)
            {
                if (keysEnterTab == 0)
                    doSimulateKeyDown(VK_Return, S_Return);
```

```
            if (keysEnterTab == 1)
                doSimulateKeyDown(VK_Tab, S_Tab);
            return;
        }
        if (micetypeeventX != 0x777)
        {
            doSimulateKeyDown(micetypeeventX, micetypeeventY);
            return;
        }
    }
    public void miceeventf(UInt16 micetypeeventX, UInt16 micetypeeventY)
    {
        if (micetypeeventX == 0x888)
        {
            doRightClickF();
            return;
        }
        if (micetypeeventX == 0x999)
        {
            doLeftClickF();
            return;
        }
        if (micetypeeventX == 0x666)
        {
            doMiddleClickF();
            return;
        }
        if (micetypeeventX == 0x444)
            return;
        if (micetypeeventX == 0x333)
            return;
        if (micetypeeventX == 0x25 | micetypeeventX == 0x26 | micetypeeventX == 0x27 |
micetypeeventX == 0x28)
        {
            doSimulateKeyUpArrows(micetypeeventX, micetypeeventY);
            return;
        }
        if (micetypeeventX == 0x111)
        {
            if (keys123 == 0)
            {
                doSimulateKeyUp(VK_0, S_0);
                keys123 = 1;
            }
            else
            {
                if (keys123 == 1)
                {
                    doSimulateKeyUp(VK_1, S_1);
                    keys123 = 2;
                }
                else
                    if (keys123 == 2)
                    {
                        doSimulateKeyUp(VK_2, S_2);
                        keys123 = 3;
                    }
                    else
                        if (keys123 == 3)
                        {
                            doSimulateKeyUp(VK_3, S_3);
                            keys123 = 4;
                        }
                        else
                            if (keys123 == 4)
                            {
```

```
                            doSimulateKeyUp(VK_4, S_4);
                            keys123 = 0;
                        }
            }
            return;
        }
        if (micetypeeventX == 0x222)
        {
            if (keys456 == 0)
            {
                doSimulateKeyUp(VK_5, S_5);
                keys456 = 1;
            }
            else
            {
                if (keys456 == 1)
                {
                    doSimulateKeyUp(VK_6, S_6);
                    keys456 = 2;
                }
                else
                    if (keys456 == 2)
                    {
                        doSimulateKeyUp(VK_7, S_7);
                        keys456 = 3;
                    }
                    else
                        if (keys456 == 3)
                        {
                            doSimulateKeyUp(VK_8, S_8);
                            keys456 = 4;
                        }
                        else
                            if (keys456 == 4)
                            {
                                doSimulateKeyUp(VK_9, S_9);
                                keys456 = 0;
                            }
            }
            return;
        }
        if (micetypeeventX == 0x555)
        {
            if (keysEnterTab == 0)
            {
                doSimulateKeyUp(VK_Return, S_Return);
                keysEnterTab = 1;
            }
            else
            {
                if (keysEnterTab == 1)
                {
                    doSimulateKeyUp(VK_Tab, S_Tab);
                    keysEnterTab = 0;
                }
            }
            return;
        }
        if (micetypeeventX != 0x777)
        {
            doSimulateKeyUp(micetypeeventX, micetypeeventY);
            return;
        }
    }
    public void switchwheelfix()
    {
```

```
            foreach (bool Value in _Value)
            {
                if (Value)
                {
                    _value = true;
                    break;
                }
                Thread.Sleep(1);
            }
            if (_value)
            {
                doSimulateKeyUp(VK_A, S_A);
                doSimulateKeyUp(VK_Q, S_Q);
                doSimulateKeyUpArrows(VK_LEFT, S_LEFT);
                doSimulateKeyUp(VK_D, S_D);
                doSimulateKeyUpArrows(VK_RIGHT, S_RIGHT);
                doSimulateKeyUp(VK_W, S_W);
                doSimulateKeyUp(VK_Z, S_Z);
                doSimulateKeyUpArrows(VK_UP, S_UP);
                doSimulateKeyUp(VK_S, S_S);
                doSimulateKeyUpArrows(VK_DOWN, S_DOWN);
                doRightClickF();
                doLeftClickF();
                doMiddleClickF();
                doSimulateKeyUp(VK_0, S_0);
                doSimulateKeyUp(VK_1, S_1);
                doSimulateKeyUp(VK_2, S_2);
                doSimulateKeyUp(VK_3, S_3);
                doSimulateKeyUp(VK_4, S_4);
                doSimulateKeyUp(VK_5, S_5);
                doSimulateKeyUp(VK_6, S_6);
                doSimulateKeyUp(VK_7, S_7);
                doSimulateKeyUp(VK_8, S_8);
                doSimulateKeyUp(VK_9, S_9);
                doSimulateKeyUp(VK_Return, S_Return);
                doSimulateKeyUp(VK_Tab, S_Tab);
                doSimulateKeyUp(actionassign["//wiimote alone up"].X,
actionassign["//wiimote alone up"].Y);
                doSimulateKeyUp(actionassign["//wiimote alone down"].X,
actionassign["//wiimote alone down"].Y);
                doSimulateKeyUp(actionassign["//wiimote b"].X, actionassign["//wiimote
b"].Y);
                doSimulateKeyUp(actionassign["//wiimote a"].X, actionassign["//wiimote
a"].Y);
                doSimulateKeyUp(actionassign["//wiimote plus"].X, actionassign["//wiimote
plus"].Y);
                doSimulateKeyUp(actionassign["//wiimote minus"].X, actionassign["//wiimote
minus"].Y);
                doSimulateKeyUp(actionassign["//wiimote alone roll left"].X,
actionassign["//wiimote alone roll left"].Y);
                doSimulateKeyUp(actionassign["//wiimote alone home"].X,
actionassign["//wiimote alone home"].Y);
                doSimulateKeyUp(actionassign["//wiimote alone a and b"].X,
actionassign["//wiimote alone a and b"].Y);
                doSimulateKeyUp(actionassign["//wiimote alone left"].X,
actionassign["//wiimote alone left"].Y);
                doSimulateKeyUp(actionassign["//wiimote alone right"].X,
actionassign["//wiimote alone right"].Y);
                doSimulateKeyUp(actionassign["//wiimote alone 1"].X,
actionassign["//wiimote alone 1"].Y);
                doSimulateKeyUp(actionassign["//wiimote alone 2"].X,
actionassign["//wiimote alone 2"].Y);
                doSimulateKeyUp(actionassign["//wiimote nunchuck roll right"].X,
actionassign["//wiimote nunchuck roll right"].Y);
                doSimulateKeyUp(actionassign["//wiimote nunchuck roll left"].X,
actionassign["//wiimote nunchuck roll left"].Y);
```

```
                doSimulateKeyUp(actionassign["//wiimote to front"].X,
actionassign["//wiimote to front"].Y);
                doSimulateKeyUp(actionassign["//nunchuck to down"].X,
actionassign["//nunchuck to down"].Y);
                doSimulateKeyUp(actionassign["//nunchuck to down 2"].X,
actionassign["//nunchuck to down 2"].Y);
                doSimulateKeyUp(actionassign["//nunchuck to down 3"].X,
actionassign["//nunchuck to down 3"].Y);
                doSimulateKeyUp(actionassign["//nunchuck z and c"].X,
actionassign["//nunchuck z and c"].Y);
                doSimulateKeyUp(actionassign["//nunchuck c"].X, actionassign["//nunchuck
c"].Y);
                doSimulateKeyUp(actionassign["//nunchuck z"].X, actionassign["//nunchuck
z"].Y);
                doSimulateKeyUp(actionassign["//wiimote nunchuck down"].X,
actionassign["//wiimote nunchuck down"].Y);
                doSimulateKeyUp(actionassign["//wiimote nunchuck home"].X,
actionassign["//wiimote nunchuck home"].Y);
                doSimulateKeyUp(actionassign["//wiimote nunchuck left"].X,
actionassign["//wiimote nunchuck left"].Y);
                doSimulateKeyUp(actionassign["//wiimote nunchuck right"].X,
actionassign["//wiimote nunchuck right"].Y);
                doSimulateKeyUp(actionassign["//wiimote nunchuck up"].X,
actionassign["//wiimote nunchuck up"].Y);
                doSimulateKeyUp(actionassign["//wiimote nunchuck 1"].X,
actionassign["//wiimote nunchuck 1"].Y);
                doSimulateKeyUp(actionassign["//wiimote nunchuck 2"].X,
actionassign["//wiimote nunchuck 2"].Y);
                doSimulateKeyUp(actionassign["//1 tab switch"].X, actionassign["//1 tab
switch"].Y);
                doSimulateKeyUp(actionassign["//cancel reload x"].X, actionassign["//cancel
reload x"].Y);
                doSimulateKeyUp(actionassign["//double A r"].X, actionassign["//double A
r"].Y);
            }
            _value = false;
        }
        public void clearList()
        {
            valListXn.Clear();
            valListYn.Clear();
        }
        public void setControlsAndOptions()
        {
            if (Fbool["//lock features and options"])
                lockchangefeaturesandoptions = true;
        }
        public void Assignating()
        {
            if (!Fbool["//rebind keys"])
            {
                action["//wiimote to front"] = cmBWNTOFRONT.Items[0].ToString();
                action["//wiimote plus"] = cmBWNPLUS.Items[0].ToString();
                action["//wiimote minus"] = cmBWNMINUS.Items[0].ToString();
                action["//cancel reload x"] = cmBWNCANCELRELOAD.Items[0].ToString();
                action["//double A r"] = cmBWNDOUBLEA.Items[0].ToString();
                action["//1 tab switch"] = cmBWN1SWITCH.Items[0].ToString();
                action["//wiimote alone home"] =
cmBWAHOME.Items[0].ToString().Replace("(R)", "");
                action["//wiimote alone a and b"] = cmBWAAB.Items[0].ToString();
                action["//wiimote alone 1"] = cmBWAONE.Items[0].ToString();
                action["//wiimote alone 2"] = cmBWATWO.Items[0].ToString();
                action["//nunchuck z and c"] = cmBWNCZ.Items[0].ToString();
                action["//nunchuck z"] = cmBWNZ.Items[0].ToString();
                action["//wiimote nunchuck home"] =
cmBWNHOME.Items[0].ToString().Replace("(R)", "");
```

```
                action["//wiimote nunchuck up"] = cmBWNUP.Items[0].ToString();
                action["//wiimote nunchuck 2"] = cmBWNTWO.Items[0].ToString();
                action["//wiimote alone roll left"] = cmBWAROLLLEFT.Items[0].ToString();
                action["//wiimote nunchuck roll right"] =
cmBWNROLLRIGHT.Items[0].ToString();
                action["//wiimote nunchuck roll left"] = cmBWNROLLLEFT.Items[0].ToString();
                action["//nunchuck to down 2"] = cmBWNTODOWN2.Items[0].ToString();
                action["//nunchuck to down"] = cmBWNTODOWN1.Items[0].ToString();
                action["//nunchuck to down 3"] = cmBWNTODOWN3.Items[0].ToString();
                action["//wiimote b"] = cmBWNB.Items[0].ToString();
                action["//wiimote a"] = cmBWNA.Items[0].ToString();
                action["//wiimote alone up"] = cmBWAUP.Items[0].ToString();
                action["//wiimote alone down"] = cmBWADOWN.Items[0].ToString();
                action["//wiimote alone left"] = cmBWALEFT.Items[0].ToString();
                action["//wiimote alone right"] = cmBWARIGHT.Items[0].ToString();
                action["//nunchuck c"] = cmBWNC.Items[0].ToString();
                action["//wiimote nunchuck down"] = cmBWNDOWN.Items[0].ToString();
                action["//wiimote alone wheel"] = cmBWAWHEELa.Items[0].ToString();
                action["//nunchuck stick"] = cmBWNSTICK.Items[0].ToString();
                action["//wiimote nunchuck right"] = cmBWNRIGHT.Items[0].ToString();
                action["//wiimote nunchuck left"] = cmBWNLEFT.Items[0].ToString();
                action["//wiimote nunchuck 1"] = cmBWNONE.Items[0].ToString();
            }
            cmBWNTOFRONT.Text = action["//wiimote to front"];
            cmBWNPLUS.Text = action["//wiimote plus"];
            cmBWNMINUS.Text = action["//wiimote minus"];
            cmBWN1SWITCH.Text = action["//1 tab switch"];
            cmBWAAB.Text = action["//wiimote alone a and b"];
            cmBWAONE.Text = action["//wiimote alone 1"];
            cmBWATWO.Text = action["//wiimote alone 2"];
            cmBWNCZ.Text = action["//nunchuck z and c"];
            cmBWNZ.Text = action["//nunchuck z"];
            cmBWNUP.Text = action["//wiimote nunchuck up"];
            cmBWNTWO.Text = action["//wiimote nunchuck 2"];
            cmBWAROLLLEFT.Text = action["//wiimote alone roll left"];
            cmBWNTODOWN2.Text = action["//nunchuck to down 2"];
            cmBWNTODOWN1.Text = action["//nunchuck to down"];
            cmBWNTODOWN3.Text = action["//nunchuck to down 3"];
            cmBWNB.Text = action["//wiimote b"];
            cmBWNA.Text = action["//wiimote a"];
            cmBWAUP.Text = action["//wiimote alone up"];
            cmBWADOWN.Text = action["//wiimote alone down"];
            cmBWALEFT.Text = action["//wiimote alone left"];
            cmBWARIGHT.Text = action["//wiimote alone right"];
            cmBWNC.Text = action["//nunchuck c"];
            cmBWNDOWN.Text = action["//wiimote nunchuck down"];
            cmBWNSTICK.Text = action["//nunchuck stick"];
            cmBWNRIGHT.Text = action["//wiimote nunchuck right"];
            cmBWNLEFT.Text = action["//wiimote nunchuck left"];
            cmBWNONE.Text = action["//wiimote nunchuck 1"];
            if (chkBF8S.Checked)
            {
                cmBWNHOME.Text = action["//wiimote nunchuck home"];
                cmBWAHOME.Text = action["//wiimote alone home"];
            }
            else
            {
                cmBWNHOME.Text = action["//wiimote nunchuck home"] + "(R)";
                cmBWAHOME.Text = action["//wiimote alone home"] + "(R)";
            }
            if (!chkBF9S.Checked)
                cmBWAWHEELa.Text = " ";
            else
                cmBWAWHEELa.Text = action["//wiimote alone wheel"];
            if (chkBF10S.Checked)
            {
```

```
                cmBWNROLLLEFT.Text = " ";
                cmBWNROLLRIGHT.Text = " ";
            }
            else
            {
                cmBWNROLLLEFT.Text = action["//wiimote nunchuck roll left"];
                cmBWNROLLRIGHT.Text = action["//wiimote nunchuck roll right"];
            }
            if (!chkBF2C.Checked)
                cmBWNCANCELRELOAD.Text = " ";
            else
                cmBWNCANCELRELOAD.Text = action["//cancel reload x"];
            if (!chkBF3C.Checked)
                cmBWNDOUBLEA.Text = " ";
            else
                cmBWNDOUBLEA.Text = action["//double A r"];
            actionassign["//wiimote to front"] = assign(action["//wiimote to front"]);
            actionassign["//wiimote plus"] = assign(action["//wiimote plus"]);
            actionassign["//wiimote minus"] = assign(action["//wiimote minus"]);
            actionassign["//cancel reload x"] = assign(action["//cancel reload x"]);
            actionassign["//double A r"] = assign(action["//double A r"]);
            actionassign["//1 tab switch"] = assign(action["//1 tab switch"]);
            actionassign["//wiimote alone home"] = assign(action["//wiimote alone
home"].Replace("(R)", ""));
            actionassign["//wiimote alone a and b"] = assign(action["//wiimote alone a and
b"]);
            actionassign["//wiimote alone 1"] = assign(action["//wiimote alone 1"]);
            actionassign["//wiimote alone 2"] = assign(action["//wiimote alone 2"]);
            actionassign["//nunchuck z and c"] = assign(action["//nunchuck z and c"]);
            actionassign["//nunchuck z"] = assign(action["//nunchuck z"]);
            actionassign["//wiimote nunchuck home"] = assign(action["//wiimote nunchuck
home"].Replace("(R)", ""));
            actionassign["//wiimote nunchuck up"] = assign(action["//wiimote nunchuck
up"]);
            actionassign["//wiimote nunchuck 2"] = assign(action["//wiimote nunchuck 2"]);
            actionassign["//wiimote alone roll left"] = assign(action["//wiimote alone roll
left"]);
            actionassign["//wiimote nunchuck roll right"] = assign(action["//wiimote
nunchuck roll right"]);
            actionassign["//wiimote nunchuck roll left"] = assign(action["//wiimote
nunchuck roll left"]);
            actionassign["//nunchuck to down 2"] = assign(action["//nunchuck to down 2"]);
            actionassign["//nunchuck to down"] = assign(action["//nunchuck to down"]);
            actionassign["//nunchuck to down 3"] = assign(action["//nunchuck to down 3"]);
            actionassign["//wiimote b"] = assign(action["//wiimote b"]);
            actionassign["//wiimote a"] = assign(action["//wiimote a"]);
            actionassign["//wiimote alone up"] = assign(action["//wiimote alone up"]);
            actionassign["//wiimote alone down"] = assign(action["//wiimote alone down"]);
            actionassign["//wiimote alone left"] = assign(action["//wiimote alone left"]);
            actionassign["//wiimote alone right"] = assign(action["//wiimote alone
right"]);
            actionassign["//nunchuck c"] = assign(action["//nunchuck c"]);
            actionassign["//wiimote nunchuck down"] = assign(action["//wiimote nunchuck
down"]);
            actionassign["//wiimote nunchuck 1"] = assign(action["//wiimote nunchuck 1"]);
            actionassign["//wiimote nunchuck right"] = assign(action["//wiimote nunchuck
right"]);
            actionassign["//wiimote nunchuck left"] = assign(action["//wiimote nunchuck
left"]);
        }
        public void SelectOptions()
        {
            if (Fbool["//A view on"])
            {
                if (!Fbool["//A C swap"])
                    this[3] = mWSButtonStateAio | (mWSButtonStateA & foraorcison);
```

```
                else
                    this[3] = mWSNunchuckStateCio | (mWSNunchuckStateC & foraorcison);
                if (_Valuechanged[3] & !this[3])
                {
                    Fbool["//warface"] = false;
                    Fbool["//mw3"] = false;
                    Fbool["//cursor"] = false;
                    Fbool["//titanfall"] = false;
                    Fbool["//brink"] = false;
                    Fbool["//metro"] = false;
                    Fbool["//fake"] = false;
                    Fbool["//bo3"] = false;
                    Fbool["//xaim"] = false;
                }
                if (_Valuechanged[3] & this[3])
                {
                    Fbool["//warface"] = chkBF5.Checked;
                    Fbool["//mw3"] = chkBF8.Checked;
                    Fbool["//cursor"] = chkBF4.Checked;
                    Fbool["//titanfall"] = chkBF3.Checked;
                    Fbool["//brink"] = chkBF1.Checked;
                    Fbool["//metro"] = chkBF2.Checked;
                    Fbool["//fake"] = chkBF7.Checked;
                    Fbool["//bo3"] = chkBF6.Checked;
                    Fbool["//xaim"] = chkBF9.Checked;
                }
            }
            getstate = GetAsyncKeyState(System.Windows.Forms.Keys.ControlKey);
            _getstate = GetAsyncKeyState(System.Windows.Forms.Keys.ShiftKey);
            _Getstate = GetAsyncKeyState(System.Windows.Forms.Keys.CapsLock);
            this[78] = getstate & _getstate & _Getstate;
            if (_Valuechanged[78] & this[78] & !Fbool["//lock features and options"])
                Fbool["//lock features and options"] = true;
            else
                if (_Valuechanged[78] & this[78] & Fbool["//lock features and options"])
                    Fbool["//lock features and options"] = false;
            this[89] = GetAsyncKeyState(System.Windows.Forms.Keys.Decimal);
            if (_Valuechanged[89] & this[89] & !Getstate)
                Getstate = true;
            else
                if (_Valuechanged[89] & this[89] & Getstate)
                    Getstate = false;
            if (!Fbool["//lock features and options"] | lockchangefeaturesandoptions)
            {
                this[1] = mWSNunchuckStateZ & mWSButtonStateOne;
                this[2] = mWSNunchuckStateC & mWSButtonStateOne;
                if (mWSNunchuckStateZ & mWSButtonStateOne)
                    switchtowheelwithwheelscriptcount += 1;
                if (_Valuechanged[1] & this[1])
                    switchtowheelwithwheelscriptcount = 0;
                if (mWSNunchuckStateC & mWSButtonStateOne)
                    switchtowheelwithoutwheelscriptcount += 1;
                if (_Valuechanged[2] & this[2])
                    switchtowheelwithoutwheelscriptcount = 0;
                if ((_Valuechanged[1] & !this[1] & switchtowheelwithwheelscriptcount < 10)
| (_Valuechanged[2] & !this[2] & switchtowheelwithoutwheelscriptcount < 10))
                {
                    if (_Valuechanged[1] & !this[1])
                        Fbool["//wheel script"] = true;
                    if (_Valuechanged[2] & !this[2])
                        Fbool["//wheel script"] = false;
                    if (Fbool["//nunchuck"])
                    {
                        Fbool["//nunchuck"] = false;
                        Fbool["//warface"] = false;
                        Fbool["//mw3"] = false;
```

```
                Fbool["//cursor"] = false;
                Fbool["//titanfall"] = false;
                Fbool["//brink"] = false;
                Fbool["//metro"] = false;
                Fbool["//fake"] = false;
                Fbool["//bo3"] = false;
                Fbool["//xaim"] = false;
            }
            else
            {
                Fbool["//nunchuck"] = true;
                Fbool["//warface"] = chkBF5.Checked;
                Fbool["//mw3"] = chkBF8.Checked;
                Fbool["//cursor"] = chkBF4.Checked;
                Fbool["//titanfall"] = chkBF3.Checked;
                Fbool["//brink"] = chkBF1.Checked;
                Fbool["//metro"] = chkBF2.Checked;
                Fbool["//fake"] = chkBF7.Checked;
                Fbool["//bo3"] = chkBF6.Checked;
                Fbool["//xaim"] = chkBF9.Checked;
            }
            switchwheelfix();
        }
        this[4] = GetAsyncKeyState(System.Windows.Forms.Keys.F1) & !_getstate &
!getstate;
        if ((_Valuechanged[4] & !this[4]) | Abool["//brink"])
        {
            if (Fbool["//brink"] == false)
            {
                Fbool["//brink"] = true;
                chkBF1.Checked = true;
            }
            else
            {
                Fbool["//brink"] = false;
                chkBF1.Checked = false;
            }
            Abool["//brink"] = false;
        }
        this[5] = GetAsyncKeyState(System.Windows.Forms.Keys.F2) & !_getstate &
!getstate;
        if ((_Valuechanged[5] & !this[5]) | Abool["//metro"])
        {
            if (Fbool["//metro"] == false)
            {
                Fbool["//metro"] = true;
                chkBF2.Checked = true;
            }
            else
            {
                Fbool["//metro"] = false;
                chkBF2.Checked = false;
            }
            Abool["//metro"] = false;
        }
        this[6] = GetAsyncKeyState(System.Windows.Forms.Keys.F3) & !_getstate &
!getstate;
        if ((_Valuechanged[6] & !this[6]) | Abool["//titanfall"])
        {
            if (Fbool["//titanfall"] == false)
            {
                Fbool["//titanfall"] = true;
                chkBF3.Checked = true;
            }
            else
            {
```

```
                    Fbool["//titanfall"] = false;
                    chkBF3.Checked = false;
                }
                Abool["//titanfall"] = false;
            }
            this[7] = GetAsyncKeyState(System.Windows.Forms.Keys.F4) & !_getstate &
!getstate;
            if ((_Valuechanged[7] & !this[7]) | Abool["//cursor"])
            {
                if (Fbool["//cursor"] == false)
                {
                    Fbool["//cursor"] = true;
                    chkBF4.Checked = true;
                }
                else
                {
                    Fbool["//cursor"] = false;
                    chkBF4.Checked = false;
                }
                Abool["//cursor"] = false;
            }
            this[8] = GetAsyncKeyState(System.Windows.Forms.Keys.F12) & !_getstate &
!getstate;
            if ((_Valuechanged[8] & !this[8]) | Abool["//A press I/O"])
            {
                if (Fbool["//A press I/O"] == false)
                {
                    Fbool["//A press I/O"] = true;
                    chkBF12.Checked = true;
                }
                else
                {
                    Fbool["//A press I/O"] = false;
                    chkBF12.Checked = false;
                }
                Abool["//A press I/O"] = false;
            }
            this[9] = GetAsyncKeyState(System.Windows.Forms.Keys.F5) & !_getstate &
!getstate;
            if ((_Valuechanged[9] & !this[9]) | Abool["//warface"])
            {
                if (Fbool["//warface"] == false)
                {
                    Fbool["//warface"] = true;
                    chkBF5.Checked = true;
                }
                else
                {
                    Fbool["//warface"] = false;
                    chkBF5.Checked = false;
                }
                Abool["//warface"] = false;
            }
            this[10] = GetAsyncKeyState(System.Windows.Forms.Keys.F6) & !_getstate &
!getstate;
            if ((_Valuechanged[10] & !this[10]) | Abool["//bo3"])
            {
                if (Fbool["//bo3"] == false)
                {
                    Fbool["//bo3"] = true;
                    chkBF6.Checked = true;
                }
                else
                {
                    Fbool["//bo3"] = false;
                    chkBF6.Checked = false;
```

```
                }
                Abool["//bo3"] = false;
            }
            this[11] = GetAsyncKeyState(System.Windows.Forms.Keys.F7) & !_getstate &
!getstate;
            if ((_Valuechanged[11] & !this[11]) | Abool["//fake"])
            {
                if (Fbool["//fake"] == false)
                {
                    Fbool["//fake"] = true;
                    chkBF7.Checked = true;
                }
                else
                {
                    Fbool["//fake"] = false;
                    chkBF7.Checked = false;
                }
                Abool["//fake"] = false;
            }
            this[12] = GetAsyncKeyState(System.Windows.Forms.Keys.F8) & !_getstate &
!getstate;
            if ((_Valuechanged[12] & !this[12]) | Abool["//mw3"])
            {
                if (Fbool["//mw3"] == false)
                {
                    Fbool["//mw3"] = true;
                    chkBF8.Checked = true;
                }
                else
                {
                    Fbool["//mw3"] = false;
                    chkBF8.Checked = false;
                }
                Abool["//mw3"] = false;
            }
            this[14] = GetAsyncKeyState(System.Windows.Forms.Keys.F9) & _getstate &
!getstate;
            if ((_Valuechanged[14] & !this[14]) | Abool["//wheel script"])
            {
                if (Fbool["//wheel script"] == false)
                {
                    Fbool["//wheel script"] = true;
                    chkBF9S.Checked = true;
                }
                else
                {
                    Fbool["//wheel script"] = false;
                    chkBF9S.Checked = false;
                }
                Abool["//wheel script"] = false;
            }
            this[15] = GetAsyncKeyState(System.Windows.Forms.Keys.F11) & !_getstate &
!getstate;
            if ((_Valuechanged[15] & !this[15]) | Abool["//A accuracy"])
            {
                if (Fbool["//A accuracy"] == false)
                {
                    Fbool["//A accuracy"] = true;
                    chkBF11.Checked = true;
                }
                else
                {
                    Fbool["//A accuracy"] = false;
                    chkBF11.Checked = false;
                }
                Abool["//A accuracy"] = false;
```

```
                }
                this[16] = GetAsyncKeyState(System.Windows.Forms.Keys.F10) & _getstate &
!getstate;
                if ((_Valuechanged[16] & !this[16]) | Abool["//no roll qe"])
                {
                    if (Fbool["//no roll qe"] == false)
                    {
                        Fbool["//no roll qe"] = true;
                        chkBF10S.Checked = true;
                    }
                    else
                    {
                        Fbool["//no roll qe"] = false;
                        chkBF10S.Checked = false;
                    }
                    Abool["//no roll qe"] = false;
                }
                this[17] = GetAsyncKeyState(System.Windows.Forms.Keys.F8) & _getstate &
!getstate;
                if ((_Valuechanged[17] & !this[17]) | Abool["//Home f only"])
                {
                    if (Fbool["//Home f only"] == false)
                    {
                        Fbool["//Home f only"] = true;
                        chkBF8S.Checked = true;
                    }
                    else
                    {
                        Fbool["//Home f only"] = false;
                        chkBF8S.Checked = false;
                    }
                    Abool["//Home f only"] = false;
                }
                this[18] = GetAsyncKeyState(System.Windows.Forms.Keys.F1) & _getstate &
!getstate;
                if ((_Valuechanged[18] & !this[18]) | Abool["//A view on"])
                {
                    if (Fbool["//A view on"] == false)
                    {
                        Fbool["//A view on"] = true;
                        chkBF1S.Checked = true;
                    }
                    else
                    {
                        Fbool["//A view on"] = false;
                        chkBF1S.Checked = false;
                    }
                    Abool["//A view on"] = false;
                }
                this[19] = GetAsyncKeyState(System.Windows.Forms.Keys.F6) & !_getstate &
getstate;
                if ((_Valuechanged[19] & !this[19]) | Abool["//1 tab I/O"])
                {
                    if (Fbool["//1 tab I/O"] == false)
                    {
                        Fbool["//1 tab I/O"] = true;
                        chkBF6C.Checked = true;
                    }
                    else
                    {
                        Fbool["//1 tab I/O"] = false;
                        chkBF6C.Checked = false;
                    }
                    Abool["//1 tab I/O"] = false;
                }
```

```
                this[20] = GetAsyncKeyState(System.Windows.Forms.Keys.F2) & _getstate &
!getstate;
                if ((_Valuechanged[20] & !this[20]) | Abool["//A aim plus"])
                {
                    if (Fbool["//A aim plus"] == false)
                    {
                        Fbool["//A aim plus"] = true;
                        chkBF2S.Checked = true;
                    }
                    else
                    {
                        Fbool["//A aim plus"] = false;
                        chkBF2S.Checked = false;
                    }
                    Abool["//A aim plus"] = false;
                }
                this[21] = GetAsyncKeyState(System.Windows.Forms.Keys.F9) & !_getstate &
!getstate;
                if ((_Valuechanged[21] & !this[21]) | Abool["//xaim"])
                {
                    if (Fbool["//xaim"] == false)
                    {
                        Fbool["//xaim"] = true;
                        chkBF9.Checked = true;
                    }
                    else
                    {
                        Fbool["//xaim"] = false;
                        chkBF9.Checked = false;
                    }
                    Abool["//xaim"] = false;
                }
                this[23] = GetAsyncKeyState(System.Windows.Forms.Keys.F11) & _getstate &
!getstate;
                if ((_Valuechanged[23] & !this[23]) | Abool["//Home -> - + +"])
                {
                    if (Fbool["//Home -> - + +"] == false)
                    {
                        Fbool["//Home -> - + +"] = true;
                        chkBF11S.Checked = true;
                    }
                    else
                    {
                        Fbool["//Home -> - + +"] = false;
                        chkBF11S.Checked = false;
                    }
                    Abool["//Home -> - + +"] = false;
                }
                this[24] = GetAsyncKeyState(System.Windows.Forms.Keys.F6) & _getstate &
!getstate;
                if ((_Valuechanged[24] & !this[24]) | Abool["//1 and 2 view"])
                {
                    if (Fbool["//1 and 2 view"] == false)
                    {
                        Fbool["//1 and 2 view"] = true;
                        chkBF6S.Checked = true;
                    }
                    else
                    {
                        Fbool["//1 and 2 view"] = false;
                        chkBF6S.Checked = false;
                    }
                    Abool["//1 and 2 view"] = false;
                }
                this[25] = GetAsyncKeyState(System.Windows.Forms.Keys.F12) & _getstate &
!getstate;
```

```
                if ((_Valuechanged[25] & !this[25]) | Abool["//nunchuck"])
                {
                    if (Fbool["//nunchuck"] == false)
                    {
                        Fbool["//nunchuck"] = true;
                        chkBF12S.Checked = true;
                    }
                    else
                    {
                        Fbool["//nunchuck"] = false;
                        chkBF12S.Checked = false;
                    }
                    Abool["//nunchuck"] = false;
                }
                this[26] = mWSNunchuckStateZ & mWSNunchuckStateC & mWSButtonStateDown;
                if (mWSNunchuckStateZ & mWSNunchuckStateC & mWSButtonStateDown)
                    switchtowheelcount += 1;
                if (_Valuechanged[26] & this[26])
                    switchtowheelcount = 0;
                this[27] = GetAsyncKeyState(System.Windows.Forms.Keys.F4) & _getstate &
!getstate;
                if ((_Valuechanged[27] & !this[27]) | Abool["//wheel view"] |
(_Valuechanged[26] & !this[26] & switchtowheelcount < 10))
                {
                    if (Fbool["//wheel view"] == false)
                    {
                        Fbool["//wheel view"] = true;
                        chkBF4S.Checked = true;
                    }
                    else
                    {
                        Fbool["//wheel view"] = false;
                        chkBF4S.Checked = false;
                    }
                    Abool["//wheel view"] = false;
                    switchwheelfix();
                }
                this[28] = mWSNunchuckStateZ & mWSNunchuckStateC & mWSButtonStateUp;
                if (mWSNunchuckStateZ & mWSNunchuckStateC & mWSButtonStateUp)
                    switchtostickcount += 1;
                if (_Valuechanged[28] & this[28])
                    switchtostickcount = 0;
                this[29] = GetAsyncKeyState(System.Windows.Forms.Keys.F3) & _getstate &
!getstate;
                if ((_Valuechanged[29] & !this[29]) | Abool["//stick view"] |
(_Valuechanged[28] & !this[28] & switchtostickcount < 10))
                {
                    if (Fbool["//stick view"] == false)
                    {
                        Fbool["//stick view"] = true;
                        chkBF3S.Checked = true;
                        if (_Valuechanged[28] & !this[28])
                            stickviewswitch = true;
                    }
                    else
                    {
                        Fbool["//stick view"] = false;
                        chkBF3S.Checked = false;
                        if (_Valuechanged[28] & !this[28])
                            stickviewswitch = false;
                    }
                    Abool["//stick view"] = false;
                    switchwheelfix();
                }
                this[66] = GetAsyncKeyState(System.Windows.Forms.Keys.F2) & !_getstate &
getstate;
```

```
                if ((_Valuechanged[66] & !this[66]) | Abool["//cancel reload x"])
                {
                    if (Fbool["//cancel reload x"] == false)
                    {
                        Fbool["//cancel reload x"] = true;
                        chkBF2C.Checked = true;
                    }
                    else
                    {
                        Fbool["//cancel reload x"] = false;
                        chkBF2C.Checked = false;
                    }
                    Abool["//cancel reload x"] = false;
                }
                this[67] = GetAsyncKeyState(System.Windows.Forms.Keys.F3) & !_getstate &
getstate;
                if ((_Valuechanged[67] & !this[67]) | Abool["//double A r"])
                {
                    if (Fbool["//double A r"] == false)
                    {
                        Fbool["//double A r"] = true;
                        chkBF3C.Checked = true;
                    }
                    else
                    {
                        Fbool["//double A r"] = false;
                        chkBF3C.Checked = false;
                    }
                    Abool["//double A r"] = false;
                }
                this[68] = GetAsyncKeyState(System.Windows.Forms.Keys.F4) & !_getstate &
getstate;
                if ((_Valuechanged[68] & !this[68]) | Abool["//1 tab switch"])
                {
                    if (Fbool["//1 tab switch"] == false)
                    {
                        Fbool["//1 tab switch"] = true;
                        chkBF4C.Checked = true;
                    }
                    else
                    {
                        Fbool["//1 tab switch"] = false;
                        chkBF4C.Checked = false;
                    }
                    Abool["//1 tab switch"] = false;
                }
                this[69] = GetAsyncKeyState(System.Windows.Forms.Keys.F1) & !_getstate &
getstate;
                if ((_Valuechanged[69] & !this[69]) | Abool["//push r 1'"])
                {
                    if (Fbool["//push r 1'"] == false)
                    {
                        Fbool["//push r 1'"] = true;
                        chkBF1C.Checked = true;
                    }
                    else
                    {
                        Fbool["//push r 1'"] = false;
                        chkBF1C.Checked = false;
                    }
                    Abool["//push r 1'"] = false;
                }
                this[74] = GetAsyncKeyState(System.Windows.Forms.Keys.F5) & !_getstate &
getstate;
                if ((_Valuechanged[74] & !this[74]) | Abool["//1 B swap"])
                {
```

```
                    if (Fbool["//1 B swap"] == false)
                    {
                        Fbool["//1 B swap"] = true;
                        chkBF5C.Checked = true;
                    }
                    else
                    {
                        Fbool["//1 B swap"] = false;
                        chkBF5C.Checked = false;
                    }
                    Abool["//1 B swap"] = false;
                }
                this[75] = GetAsyncKeyState(System.Windows.Forms.Keys.F7) & _getstate &
!getstate;
                if ((_Valuechanged[75] & !this[75]) | Abool["//A C swap"])
                {
                    if (Fbool["//A C swap"] == false)
                    {
                        Fbool["//A C swap"] = true;
                        chkBF7S.Checked = true;
                    }
                    else
                    {
                        Fbool["//A C swap"] = false;
                        chkBF7S.Checked = false;
                    }
                    Abool["//A C swap"] = false;
                }
                this[83] = GetAsyncKeyState(System.Windows.Forms.Keys.F7) & !_getstate &
getstate;
                if ((_Valuechanged[83] & !this[83]) | Abool["//A slide A+B"])
                {
                    if (Fbool["//A slide A+B"] == false)
                    {
                        Fbool["//A slide A+B"] = true;
                        chkBF7C.Checked = true;
                    }
                    else
                    {
                        Fbool["//A slide A+B"] = false;
                        chkBF7C.Checked = false;
                    }
                    Abool["//A slide A+B"] = false;
                }
                this[85] = GetAsyncKeyState(System.Windows.Forms.Keys.F5) & _getstate &
!getstate;
                if ((_Valuechanged[85] & !this[85]) | Abool["//dpad view"])
                {
                    if (Fbool["//dpad view"] == false)
                    {
                        Fbool["//dpad view"] = true;
                        chkBF5S.Checked = true;
                    }
                    else
                    {
                        Fbool["//dpad view"] = false;
                        chkBF5S.Checked = false;
                    }
                    Abool["//dpad view"] = false;
                }
                this[86] = GetAsyncKeyState(System.Windows.Forms.Keys.F8) & !_getstate &
getstate;
                if ((_Valuechanged[86] & !this[86]) | Abool["//A B switch"])
                {
                    if (Fbool["//A B switch"] == false)
                    {
```

```
                Fbool["//A B switch"] = true;
                chkBF8C.Checked = true;
            }
            else
            {
                Fbool["//A B switch"] = false;
                chkBF8C.Checked = false;
            }
            Abool["//A B switch"] = false;
        }
        this[87] = GetAsyncKeyState(System.Windows.Forms.Keys.F9) & !_getstate &
getstate;
        if ((_Valuechanged[87] & !this[87]) | Abool["//Z press I/O"])
        {
            if (Fbool["//Z press I/O"] == false)
            {
                Fbool["//Z press I/O"] = true;
                chkBF9C.Checked = true;
            }
            else
            {
                Fbool["//Z press I/O"] = false;
                chkBF9C.Checked = false;
            }
            Abool["//Z press I/O"] = false;
        }
        this[63] = GetAsyncKeyState(System.Windows.Forms.Keys.F10) & !_getstate &
!getstate;
        if ((_Valuechanged[63] & !this[63]) | Abool["//driver mouse"])
        {
            if (Fbool["//driver mouse"] == false)
            {
                Fbool["//driver mouse"] = true;
                chkBF10.Checked = true;
            }
            else
            {
                Fbool["//driver mouse"] = false;
                chkBF10.Checked = false;
            }
            Abool["//driver mouse"] = false;
        }
        this[22] = GetAsyncKeyState(System.Windows.Forms.Keys.F12) & !_getstate &
getstate;
        if ((_Valuechanged[22] & !this[22]) | Abool["//driver keyboard"])
        {
            if (Fbool["//driver keyboard"] == false)
            {
                Fbool["//driver keyboard"] = true;
                chkBF12C.Checked = true;
            }
            else
            {
                Fbool["//driver keyboard"] = false;
                chkBF12C.Checked = false;
            }
            Abool["//driver keyboard"] = false;
        }
        this[90] = GetAsyncKeyState(System.Windows.Forms.Keys.F10) & !_getstate &
getstate;
        if ((_Valuechanged[90] & !this[90]) | Abool["//stick arrows"])
        {
            if (Fbool["//stick arrows"] == false)
            {
                Fbool["//stick arrows"] = true;
                chkBF10C.Checked = true;
```

```
                    }
                    else
                    {
                        Fbool["//stick arrows"] = false;
                        chkBF10C.Checked = false;
                    }
                    Abool["//stick arrows"] = false;
                }
                this[91] = GetAsyncKeyState(System.Windows.Forms.Keys.F11) & !_getstate &
getstate;
                if ((_Valuechanged[91] & !this[91]) | Abool["//A roll qe"])
                {
                    if (Fbool["//A roll qe"] == false)
                    {
                        Fbool["//A roll qe"] = true;
                        chkBF11C.Checked = true;
                    }
                    else
                    {
                        Fbool["//A roll qe"] = false;
                        chkBF11C.Checked = false;
                    }
                    Abool["//A roll qe"] = false;
                }
                if (lockchangefeaturesandoptions)
                    lockchangefeaturesandoptions = false;
            }
        }
        public void WiimoteABMinusPlusTofront()
        {
            if (!(Fbool["//1 and 2 view"] & ((Fbool["//stick view"] & !Fbool["//wheel
view"]) | (Fbool["//stick view"] & Fbool["//wheel view"]))))
            {
                if (Fbool["//1 tab switch"] & Fbool["//1 B swap"])
                {
                    if (Fbool["//1 tab I/O"])
                    {
                        if (oneswitchbool)
                        {
                            oneswitchcount++;
                            if (oneswitchcount == 1)
                                miceevent(actionassign["//wiimote b"].X,
actionassign["//wiimote b"].Y);
                            if (oneswitchcount == (int)(Fvar["//1 tab switch press delay
time extra setting"] / (Fvar["//tick time"] < 1f ? watchK : Fvar["//tick time"] * watchK)))
                                miceeventf(actionassign["//wiimote b"].X,
actionassign["//wiimote b"].Y);
                            if (oneswitchcount >= (int)(Fvar["//1 tab switch press delay
time extra setting"] / (Fvar["//tick time"] < 1f ? watchK : Fvar["//tick time"] * watchK) +
Fvar["//1 tab switch interval time extra setting"] / (Fvar["//tick time"] < 1f ? watchK :
Fvar["//tick time"] * watchK)))
                                oneswitchcount = 0;
                        }
                        this[80] = mWSButtonStateB;
                        if (_Valuechanged[80] & this[80])
                            if (randtabwiimote1switch == 0)
                            {
                                oneswitchbool = true;
                                randtabwiimote1switch = 1;
                            }
                            else
                                if (randtabwiimote1switch == 1)
                                {
                                    miceeventf(actionassign["//wiimote b"].X,
actionassign["//wiimote b"].Y);
                                    oneswitchcount = 0;
```

```
                                oneswitchbool = false;
                                randtabwiimote1switch = 0;
                            }
                        }
                        else
                        {
                            if (!Fbool["//A B switch"])
                            {
                                if (mWSButtonStateB)
                                {
                                    rapidfirecount++;
                                    if (rapidfirecount == 1)
                                        miceevent(actionassign["//wiimote b"].X,
actionassign["//wiimote b"].Y);
                                    if (rapidfirecount == (int)(Fvar["//1 tab switch press
delay time extra setting"] / (Fvar["//tick time"] < 1f ? watchK : Fvar["//tick time"] *
watchK)))
                                        miceeventf(actionassign["//wiimote b"].X,
actionassign["//wiimote b"].Y);
                                    if (rapidfirecount >= (int)(Fvar["//1 tab switch press
delay time extra setting"] / (Fvar["//tick time"] < 1f ? watchK : Fvar["//tick time"] *
watchK) + Fvar["//1 tab switch interval time extra setting"] / (Fvar["//tick time"] < 1f ?
watchK : Fvar["//tick time"] * watchK)))
                                        rapidfirecount = 0;
                                }
                                this[76] = mWSButtonStateB;
                                if (_Valuechanged[76] & !this[76])
                                {
                                    miceeventf(actionassign["//wiimote b"].X,
actionassign["//wiimote b"].Y);
                                    rapidfirecount = 0;
                                }
                            }
                            else
                            {
                                if (mWSButtonStateB & ((mWSButtonStateA & !Fbool["//A C swap"])
| (mWSNunchuckStateC & Fbool["//A C swap"])))
                                {
                                    rapidfirecount++;
                                    if (rapidfirecount == 1)
                                        miceevent(actionassign["//wiimote b"].X,
actionassign["//wiimote b"].Y);
                                    if (rapidfirecount == (int)(Fvar["//1 tab switch press
delay time extra setting"] / (Fvar["//tick time"] < 1f ? watchK : Fvar["//tick time"] *
watchK)))
                                        miceeventf(actionassign["//wiimote b"].X,
actionassign["//wiimote b"].Y);
                                    if (rapidfirecount >= (int)(Fvar["//1 tab switch press
delay time extra setting"] / (Fvar["//tick time"] < 1f ? watchK : Fvar["//tick time"] *
watchK) + Fvar["//1 tab switch interval time extra setting"] / (Fvar["//tick time"] < 1f ?
watchK : Fvar["//tick time"] * watchK)))
                                        rapidfirecount = 0;
                                }
                                this[76] = mWSButtonStateB & ((mWSButtonStateA & !Fbool["//A C
swap"]) | (mWSNunchuckStateC & Fbool["//A C swap"]));
                                if (_Valuechanged[76] & !this[76])
                                {
                                    miceeventf(actionassign["//wiimote b"].X,
actionassign["//wiimote b"].Y);
                                    rapidfirecount = 0;
                                }
                                this[30] = mWSButtonStateB & !((mWSButtonStateA & !Fbool["//A C
swap"]) | (mWSNunchuckStateC & Fbool["//A C swap"]));
                                if (_Valuechanged[30] & this[30])
                                    miceevent(actionassign["//wiimote b"].X,
actionassign["//wiimote b"].Y);
```

```
                        if (_Valuechanged[30] & !this[30])
                            miceeventf(actionassign["//wiimote b"].X,
actionassign["//wiimote b"].Y);
                    }
                }
            }
            else
            {
                if (Fbool["//1 tab I/O"] & Fbool["//1 B swap"])
                {
                    this[82] = mWSButtonStateB;
                    if (_Valuechanged[82] & this[82])
                        if (randtabwiimote1switch == 0)
                        {
                            miceevent(actionassign["//wiimote b"].X,
actionassign["//wiimote b"].Y);
                            randtabwiimote1switch = 1;
                        }
                        else
                            if (randtabwiimote1switch == 1)
                            {
                                miceeventf(actionassign["//wiimote b"].X,
actionassign["//wiimote b"].Y);
                                randtabwiimote1switch = 0;
                            }
                }
                else
                {
                    this[30] = mWSButtonStateB;
                    if (_Valuechanged[30] & this[30])
                        miceevent(actionassign["//wiimote b"].X,
actionassign["//wiimote b"].Y);
                    if (_Valuechanged[30] & !this[30])
                        miceeventf(actionassign["//wiimote b"].X,
actionassign["//wiimote b"].Y);
                }
            }
            this[31] = mWSButtonStateAio | (mWSButtonStateA & foraorcison);
            if (_Valuechanged[31] & this[31])
            {
                rightpowerslidecount = 0;
                miceevent(actionassign["//wiimote a"].X, actionassign["//wiimote
a"].Y);
            }
            if (_Valuechanged[31] & !this[31])
                miceeventf(actionassign["//wiimote a"].X, actionassign["//wiimote
a"].Y);
        }
        if (Fbool["//push r 1'"] & pushrcount >= 1)
        {
            pushrcount++;
            if (pushrcount == 2)
                miceevent(actionassign["//wiimote to front"].X, actionassign["//wiimote
to front"].Y);
            if (pushrcount >= Fvar["//wiimote to front push r time extra setting"] /
(Fvar["//tick time"] < 1f ? watchK : Fvar["//tick time"] * watchK))
            {
                miceeventf(actionassign["//wiimote to front"].X,
actionassign["//wiimote to front"].Y);
                pushrcount = 0;
            }
        }
        this[32] = (mWSRawValuesZ > 0 ? mWSRawValuesZ : -mWSRawValuesZ) >= 40f &
(mWSRawValuesY > 0 ? mWSRawValuesY : -mWSRawValuesY) >= 40f & (mWSRawValuesX > 0 ?
mWSRawValuesX : -mWSRawValuesX) >= 40f;
        if (_Valuechanged[32] & this[32])
```

```
                miceevent(actionassign["//wiimote to front"].X, actionassign["//wiimote to
front"].Y);
            if (_Valuechanged[32] & !this[32])
            {
                miceeventf(actionassign["//wiimote to front"].X, actionassign["//wiimote to
front"].Y);
                if (Fbool["//push r 1'"])
                    pushrcount = 1;
            }
            this[33] = mWSButtonStatePlus | (mWSButtonStateHome & Fbool["//Home -> - +
+"]);
            if (_Valuechanged[33] & this[33])
                miceevent(actionassign["//wiimote plus"].X, actionassign["//wiimote
plus"].Y);
            if (_Valuechanged[33] & !this[33])
                miceeventf(actionassign["//wiimote plus"].X, actionassign["//wiimote
plus"].Y);
            this[34] = mWSButtonStateMinus | (mWSButtonStateHome & Fbool["//Home -> - +
+"]);
            if (_Valuechanged[34] & this[34])
                miceevent(actionassign["//wiimote minus"].X, actionassign["//wiimote
minus"].Y);
            if (_Valuechanged[34] & !this[34])
                miceeventf(actionassign["//wiimote minus"].X, actionassign["//wiimote
minus"].Y);
            if (Fbool["//cancel reload x"])
            {
                if ((mWSRawValuesZ > 0 ? mWSRawValuesZ : -mWSRawValuesZ) >= 40f &
(mWSRawValuesY > 0 ? mWSRawValuesY : -mWSRawValuesY) >= 40f & (mWSRawValuesX > 0 ?
mWSRawValuesX : -mWSRawValuesX) >= 40f)
                    cancelreloadcount = 1;
                if (cancelreloadcount >= 1)
                    cancelreloadcount++;
                if (((!Fbool["//A C swap"] & mWSButtonStateA) | (Fbool["//A C swap"] &
mWSNunchuckStateC)) & cancelreloadcount >= 1)
                    cancelreloadbool = true;
                else
                    cancelreloadbool = false;
                if (cancelreloadcount >= Fvar["//cancel reload waiting A or C time extra
setting"] / (Fvar["//tick time"] < 1f ? watchK : Fvar["//tick time"] * watchK))
                    cancelreloadcount = 0;
                this[70] = cancelreloadbool;
                if (_Valuechanged[70] & this[70])
                    miceevent(actionassign["//cancel reload x"].X, actionassign["//cancel
reload x"].Y);
                if (_Valuechanged[70] & !this[70])
                {
                    miceeventf(actionassign["//cancel reload x"].X, actionassign["//cancel
reload x"].Y);
                    cancelreloadcount = 0;
                }
            }
            if (Fbool["//double A r"])
            {
                if (Fbool["//push r 1'"] & doubleapushrcount >= 1)
                {
                    doubleapushrcount++;
                    if (doubleapushrcount == 2)
                        miceevent(actionassign["//double A r"].X, actionassign["//double A
r"].Y);
                    if (doubleapushrcount >= Fvar["//double A push r time extra setting"] /
(Fvar["//tick time"] < 1f ? watchK : Fvar["//tick time"] * watchK))
                    {
                        miceeventf(actionassign["//double A r"].X, actionassign["//double A
r"].Y);
                        doubleapushrcount = 0;
```

```
                    }
                }
                if ((!Fbool["//A C swap"] & mWSButtonStateA) | (Fbool["//A C swap"] &
mWSNunchuckStateC))
                    doubleacount = 1;
                if (doubleacount >= 1)
                    doubleacount++;
                if (((!Fbool["//A C swap"] & !mWSButtonStateA) | (Fbool["//A C swap"] &
!mWSNunchuckStateC)) & doubleacount >= 1)
                    doubleaacount = 1;
                if (doubleaacount >= 1)
                    doubleaacount++;
                if (((!Fbool["//A C swap"] & mWSButtonStateA) | (Fbool["//A C swap"] &
mWSNunchuckStateC)) & doubleaacount >= 1)
                    doubleabool = true;
                else
                    doubleabool = false;
                if (doubleacount >= 100f / (Fvar["//tick time"] < 1f ? watchK :
Fvar["//tick time"] * watchK))
                    doubleacount = 0;
                if (doubleaacount >= 100f / (Fvar["//tick time"] < 1f ? watchK :
Fvar["//tick time"] * watchK))
                    doubleaacount = 0;
                this[71] = doubleabool;
                if (_Valuechanged[71] & this[71])
                    miceevent(actionassign["//double A r"].X, actionassign["//double A
r"].Y);
                if (_Valuechanged[71] & !this[71])
                {
                    miceeventf(actionassign["//double A r"].X, actionassign["//double A
r"].Y);
                    if (Fbool["//push r 1'"])
                        doubleapushrcount = 1;
                    doubleacount = 0;
                    doubleaacount = 0;
                }
            }
            if (!(Fbool["//1 and 2 view"] & (Fbool["//wheel view"] & !Fbool["//stick
view"])))
            {
                if (Fbool["//1 tab switch"] & !Fbool["//1 B swap"])
                {
                    if (Fbool["//1 tab I/O"])
                    {
                        if (oneswitchbool)
                        {
                            oneswitchcount++;
                            if (oneswitchcount == 1)
                                miceevent(actionassign["//1 tab switch"].X,
actionassign["//1 tab switch"].Y);
                            if (oneswitchcount == (int)(Fvar["//1 tab switch press delay
time extra setting"] / (Fvar["//tick time"] < 1f ? watchK : Fvar["//tick time"] * watchK)))
                                miceeventf(actionassign["//1 tab switch"].X,
actionassign["//1 tab switch"].Y);
                            if (oneswitchcount >= (int)(Fvar["//1 tab switch press delay
time extra setting"] / (Fvar["//tick time"] < 1f ? watchK : Fvar["//tick time"] * watchK) +
Fvar["//1 tab switch interval time extra setting"] / (Fvar["//tick time"] < 1f ? watchK :
Fvar["//tick time"] * watchK)))
                                oneswitchcount = 0;
                        }
                        this[72] = mWSButtonStateOne;
                        if (_Valuechanged[72] & this[72])
                            if (randtabwiimote1switch == 0)
                            {
                                oneswitchbool = true;
                                randtabwiimote1switch = 1;
```

```
                            }
                            else
                                if (randtabwiimote1switch == 1)
                                {
                                    miceeventf(actionassign["//1 tab switch"].X,
actionassign["//1 tab switch"].Y);
                                    oneswitchcount = 0;
                                    oneswitchbool = false;
                                    randtabwiimote1switch = 0;
                                }
                        }
                        else
                        {
                            if (mWSButtonStateOne)
                            {
                                oneswitchcount++;
                                if (oneswitchcount == 1)
                                    miceevent(actionassign["//1 tab switch"].X,
actionassign["//1 tab switch"].Y);
                                if (oneswitchcount == (int)(Fvar["//1 tab switch press delay
time extra setting"] / (Fvar["//tick time"] < 1f ? watchK : Fvar["//tick time"] * watchK)))
                                    miceeventf(actionassign["//1 tab switch"].X,
actionassign["//1 tab switch"].Y);
                                if (oneswitchcount >= (int)(Fvar["//1 tab switch press delay
time extra setting"] / (Fvar["//tick time"] < 1f ? watchK : Fvar["//tick time"] * watchK) +
Fvar["//1 tab switch interval time extra setting"] / (Fvar["//tick time"] < 1f ? watchK :
Fvar["//tick time"] * watchK)))
                                    oneswitchcount = 0;
                            }
                            this[73] = mWSButtonStateOne;
                            if (_Valuechanged[73] & !this[73])
                            {
                                miceeventf(actionassign["//1 tab switch"].X, actionassign["//1
tab switch"].Y);
                                oneswitchcount = 0;
                            }
                        }
                    }
                    else
                    {
                        if (Fbool["//1 tab I/O"] & !Fbool["//1 B swap"])
                        {
                            this[81] = mWSButtonStateOne;
                            if (_Valuechanged[81] & this[81])
                                if (randtabwiimote1switch == 0)
                                {
                                    miceevent(actionassign["//1 tab switch"].X,
actionassign["//1 tab switch"].Y);
                                    randtabwiimote1switch = 1;
                                }
                                else
                                    if (randtabwiimote1switch == 1)
                                    {
                                        miceeventf(actionassign["//1 tab switch"].X,
actionassign["//1 tab switch"].Y);
                                        randtabwiimote1switch = 0;
                                    }
                        }
                        else
                        {
                            this[79] = mWSButtonStateOne;
                            if (_Valuechanged[79] & this[79])
                                miceevent(actionassign["//1 tab switch"].X, actionassign["//1
tab switch"].Y);
                            if (_Valuechanged[79] & !this[79])
```

```
                            miceeventf(actionassign["//1 tab switch"].X, actionassign["//1
tab switch"].Y);
                    }
                }
            }
        }
        public void WiimoteAloneButtons()
        {
            this[35] = mWSButtonStateUp;
            if (_Valuechanged[35] & this[35])
                miceevent(actionassign["//wiimote alone up"].X, actionassign["//wiimote
alone up"].Y);
            if (_Valuechanged[35] & !this[35])
                miceeventf(actionassign["//wiimote alone up"].X, actionassign["//wiimote
alone up"].Y);
            this[36] = mWSButtonStateDown;
            if (_Valuechanged[36] & this[36])
                miceevent(actionassign["//wiimote alone down"].X, actionassign["//wiimote
alone down"].Y);
            if (_Valuechanged[36] & !this[36])
                miceeventf(actionassign["//wiimote alone down"].X, actionassign["//wiimote
alone down"].Y);
            this[37] = mWSButtonStateHome;
            if (_Valuechanged[37] & this[37])
            {
                miceevent(actionassign["//wiimote alone home"].X, actionassign["//wiimote
alone home"].Y);
                if (!Fbool["//Home f only"])
                    doSimulateKeyDown(VK_R, S_R);
            }
            if (_Valuechanged[37] & !this[37])
            {
                miceeventf(actionassign["//wiimote alone home"].X, actionassign["//wiimote
alone home"].Y);
                if (!Fbool["//Home f only"])
                    doSimulateKeyUp(VK_R, S_R);
            }
            this[38] = mWSButtonStateA & mWSButtonStateB;
            if (_Valuechanged[38] & this[38])
                miceevent(actionassign["//wiimote alone a and b"].X,
actionassign["//wiimote alone a and b"].Y);
            if (_Valuechanged[38] & !this[38])
                miceeventf(actionassign["//wiimote alone a and b"].X,
actionassign["//wiimote alone a and b"].Y);
            if (Fbool["//A slide A+B"])
            {
                if (mWSButtonStateAio | (mWSButtonStateA & foraorcison))
                    rightpowerslidecount++;
                this[84] = (rightpowerslidecount >= (Fvar["//slide time"] + Fvar["//slide
init"]) / (Fvar["//tick time"] < 1f ? watchK : Fvar["//tick time"] * watchK)) &
(rightpowerslidecount <= (Fvar["//slide time"] + Fvar["//slide init"] + 50) / (Fvar["//tick
time"] < 1f ? watchK : Fvar["//tick time"] * watchK)) & (mWSButtonStateAio |
(mWSButtonStateA & foraorcison));
                if (_Valuechanged[84] & this[84])
                    miceevent(actionassign["//wiimote alone a and b"].X,
actionassign["//wiimote alone a and b"].Y);
                if (_Valuechanged[84] & !this[84])
                {
                    rightpowerslidecount = Fvar["//slide init"] / (Fvar["//tick time"] < 1f
? watchK : Fvar["//tick time"] * watchK);
                    miceeventf(actionassign["//wiimote alone a and b"].X,
actionassign["//wiimote alone a and b"].Y);
                }
            }
            this[39] = mWSButtonStateLeft;
            if (_Valuechanged[39] & this[39])
```

```
                miceevent(actionassign["//wiimote alone left"].X, actionassign["//wiimote
alone left"].Y);
            if (_Valuechanged[39] & !this[39])
                miceeventf(actionassign["//wiimote alone left"].X, actionassign["//wiimote
alone left"].Y);
            this[40] = mWSButtonStateRight;
            if (_Valuechanged[40] & this[40])
                miceevent(actionassign["//wiimote alone right"].X, actionassign["//wiimote
alone right"].Y);
            if (_Valuechanged[40] & !this[40])
                miceeventf(actionassign["//wiimote alone right"].X, actionassign["//wiimote
alone right"].Y);
            if (!(Fbool["//1 and 2 view"] & (Fbool["//wheel view"] & !Fbool["//stick
view"])))
            {
                this[41] = mWSButtonStateOne;
                if (_Valuechanged[41] & this[41])
                    miceevent(actionassign["//wiimote alone 1"].X, actionassign["//wiimote
alone 1"].Y);
                if (_Valuechanged[41] & !this[41])
                    miceeventf(actionassign["//wiimote alone 1"].X, actionassign["//wiimote
alone 1"].Y);
                this[42] = mWSButtonStateTwo;
                if (_Valuechanged[42] & this[42])
                    miceevent(actionassign["//wiimote alone 2"].X, actionassign["//wiimote
alone 2"].Y);
                if (_Valuechanged[42] & !this[42])
                    miceeventf(actionassign["//wiimote alone 2"].X, actionassign["//wiimote
alone 2"].Y);
            }
        }
        public void WiimoteNunchuckButtons()
        {
            this[43] = mWSNunchuckStateZ & mWSNunchuckStateC & !mWSButtonStateOne;
            if (_Valuechanged[43] & this[43])
                miceevent(actionassign["//nunchuck z and c"].X, actionassign["//nunchuck z
and c"].Y);
            if (_Valuechanged[43] & !this[43])
                miceeventf(actionassign["//nunchuck z and c"].X, actionassign["//nunchuck z
and c"].Y);
            if (!Fbool["//A C swap"])
                this[44] = mWSNunchuckStateC & !mWSNunchuckStateZ & !mWSButtonStateOne;
            else
                this[44] = (mWSNunchuckStateCio | (mWSNunchuckStateC & foraorcison)) &
!mWSButtonStateOne;
            if (_Valuechanged[44] & this[44])
                miceevent(actionassign["//nunchuck c"].X, actionassign["//nunchuck c"].Y);
            if (_Valuechanged[44] & !this[44])
                miceeventf(actionassign["//nunchuck c"].X, actionassign["//nunchuck c"].Y);
            this[45] = (mWSNunchuckStateZ | mWSNunchuckStateZio) & !mWSButtonStateOne;
            if (_Valuechanged[45] & this[45])
                miceevent(actionassign["//nunchuck z"].X, actionassign["//nunchuck z"].Y);
            if (_Valuechanged[45] & !this[45])
                miceeventf(actionassign["//nunchuck z"].X, actionassign["//nunchuck z"].Y);
            this[46] = mWSButtonStateDown;
            if (_Valuechanged[46] & this[46])
                miceevent(actionassign["//wiimote nunchuck down"].X,
actionassign["//wiimote nunchuck down"].Y);
            if (_Valuechanged[46] & !this[46])
                miceeventf(actionassign["//wiimote nunchuck down"].X,
actionassign["//wiimote nunchuck down"].Y);
            this[47] = mWSButtonStateHome;
            if (_Valuechanged[47] & this[47])
            {
                miceevent(actionassign["//wiimote nunchuck home"].X,
actionassign["//wiimote nunchuck home"].Y);
```

```
                if (!Fbool["//Home f only"])
                    doSimulateKeyDown(VK_R, S_R);
            }
            if (_Valuechanged[47] & !this[47])
            {
                miceeventf(actionassign["//wiimote nunchuck home"].X,
actionassign["//wiimote nunchuck home"].Y);
                if (!Fbool["//Home f only"])
                    doSimulateKeyUp(VK_R, S_R);
            }
            this[48] = mWSButtonStateLeft;
            if (_Valuechanged[48] & this[48])
                miceevent(actionassign["//wiimote nunchuck left"].X,
actionassign["//wiimote nunchuck left"].Y);
            if (_Valuechanged[48] & !this[48])
                miceeventf(actionassign["//wiimote nunchuck left"].X,
actionassign["//wiimote nunchuck left"].Y);
            this[49] = mWSButtonStateRight;
            if (_Valuechanged[49] & this[49])
                miceevent(actionassign["//wiimote nunchuck right"].X,
actionassign["//wiimote nunchuck right"].Y);
            if (_Valuechanged[49] & !this[49])
                miceeventf(actionassign["//wiimote nunchuck right"].X,
actionassign["//wiimote nunchuck right"].Y);
            this[50] = mWSButtonStateUp;
            if (_Valuechanged[50] & this[50])
                miceevent(actionassign["//wiimote nunchuck up"].X, actionassign["//wiimote
nunchuck up"].Y);
            if (_Valuechanged[50] & !this[50])
                miceeventf(actionassign["//wiimote nunchuck up"].X, actionassign["//wiimote
nunchuck up"].Y);
            if (!(Fbool["//1 and 2 view"] & (Fbool["//wheel view"] & !Fbool["//stick
view"])))
            {
                this[51] = mWSButtonStateOne & !(mWSNunchuckStateC | mWSNunchuckStateZ |
mWSNunchuckStateZio);
                if (_Valuechanged[51] & this[51])
                    miceevent(actionassign["//wiimote nunchuck 1"].X,
actionassign["//wiimote nunchuck 1"].Y);
                if (_Valuechanged[51] & !this[51])
                    miceeventf(actionassign["//wiimote nunchuck 1"].X,
actionassign["//wiimote nunchuck 1"].Y);
                this[52] = mWSButtonStateTwo;
                if (_Valuechanged[52] & this[52])
                    miceevent(actionassign["//wiimote nunchuck 2"].X,
actionassign["//wiimote nunchuck 2"].Y);
                if (_Valuechanged[52] & !this[52])
                    miceeventf(actionassign["//wiimote nunchuck 2"].X,
actionassign["//wiimote nunchuck 2"].Y);
            }
        }
        public void WiimoteAloneAnalogic()
        {
            if (Fbool["//wheel script"])
            {
                if (mWSRawValuesY > 3)
                    Rand1swps = Rand1swps + mWSRawValuesY;
                if (mWSRawValuesY < -3)
                    Rand1swms = Rand1swms + mWSRawValuesY;
                if (Rand1swps >= Fvar["//wheel script gyroscope limit out"] / (Fvar["//tick
time"] < 1f ? watchK : Fvar["//tick time"] * watchK))
                {
                    bool1swps = true;
                    Rand1swps = 0;
                }
                else
```

```
                    bool1swps = false;
                if (Rand1swms <= -Fvar["//wheel script gyroscope limit out"] /
(Fvar["//tick time"] < 1f ? watchK : Fvar["//tick time"] * watchK))
                {
                    bool1swms = true;
                    Rand1swms = 0;
                }
                else
                    bool1swms = false;
                this[53] = (mWSRawValuesY >= Fvar["//wheel script gyroscope limit in"] |
bool1swps) & Fbool["//wheel script"];
                if (_Valuechanged[53] & this[53])
                {
                    if (action["//wiimote alone wheel"] == "AD")
                        doSimulateKeyDown(VK_A, S_A);
                    if (action["//wiimote alone wheel"] == "QD")
                        doSimulateKeyDown(VK_Q, S_Q);
                    if (action["//wiimote alone wheel"] == "left right arrow keys")
                        doSimulateKeyDownArrows(VK_LEFT, S_LEFT);
                }
                if (_Valuechanged[53] & !this[53])
                {
                    if (action["//wiimote alone wheel"] == "AD")
                        doSimulateKeyUp(VK_A, S_A);
                    if (action["//wiimote alone wheel"] == "QD")
                        doSimulateKeyUp(VK_Q, S_Q);
                    if (action["//wiimote alone wheel"] == "left right arrow keys")
                        doSimulateKeyUpArrows(VK_LEFT, S_LEFT);
                }
                this[54] = (mWSRawValuesY <= -Fvar["//wheel script gyroscope limit in"] |
bool1swms) & Fbool["//wheel script"];
                if (_Valuechanged[54] & this[54])
                {
                    if (action["//wiimote alone wheel"] == "AD")
                        doSimulateKeyDown(VK_D, S_D);
                    if (action["//wiimote alone wheel"] == "QD")
                        doSimulateKeyDown(VK_D, S_D);
                    if (action["//wiimote alone wheel"] == "left right arrow keys")
                        doSimulateKeyDownArrows(VK_RIGHT, S_RIGHT);
                }
                if (_Valuechanged[54] & !this[54])
                {
                    if (action["//wiimote alone wheel"] == "AD")
                        doSimulateKeyUp(VK_D, S_D);
                    if (action["//wiimote alone wheel"] == "QD")
                        doSimulateKeyUp(VK_D, S_D);
                    if (action["//wiimote alone wheel"] == "left right arrow keys")
                        doSimulateKeyUpArrows(VK_RIGHT, S_RIGHT);
                }
            }
            this[55] = (mWSRawValuesX > 0 ? mWSRawValuesX : -mWSRawValuesX) >= 24f;
            if (_Valuechanged[55] & this[55])
                miceevent(actionassign["//wiimote alone roll left"].X,
actionassign["//wiimote alone roll left"].Y);
            if (_Valuechanged[55] & !this[55])
                miceeventf(actionassign["//wiimote alone roll left"].X,
actionassign["//wiimote alone roll left"].Y);
        }
        public void WiimoteNunchuckAnalogic()
        {
            if (Fbool["//wheel script"])
            {
                if (mWSNunchuckStateRawJoystickX > 15)
                    Rand2swps = Rand2swps + mWSNunchuckStateRawJoystickX;
                if (mWSNunchuckStateRawJoystickX < -15)
                    Rand2swms = Rand2swms + mWSNunchuckStateRawJoystickX;
```

```
                if (mWSNunchuckStateRawJoystickY > 15)
                    Rand2swyps = Rand2swyps + mWSNunchuckStateRawJoystickY;
                if (mWSNunchuckStateRawJoystickY < -15)
                    Rand2swyms = Rand2swyms + mWSNunchuckStateRawJoystickY;
                if (Rand2swps >= Fvar["//wheel script stick limit out"] / (Fvar["//tick
time"] < 1f ? watchK : Fvar["//tick time"] * watchK))
                {
                    bool2swps = true;
                    Rand2swps = 0;
                }
                else
                    bool2swps = false;
                if (Rand2swms <= -Fvar["//wheel script stick limit out"] / (Fvar["//tick
time"] < 1f ? watchK : Fvar["//tick time"] * watchK))
                {
                    bool2swms = true;
                    Rand2swms = 0;
                }
                else
                    bool2swms = false;
                if (Rand2swyps >= Fvar["//wheel script stick limit out"] / (Fvar["//tick
time"] < 1f ? watchK : Fvar["//tick time"] * watchK))
                {
                    bool2swyps = true;
                    Rand2swyps = 0;
                }
                else
                    bool2swyps = false;
                if (Rand2swyms <= -Fvar["//wheel script stick limit out"] / (Fvar["//tick
time"] < 1f ? watchK : Fvar["//tick time"] * watchK))
                {
                    bool2swyms = true;
                    Rand2swyms = 0;
                }
                else
                    bool2swyms = false;
            }
            if (!stickviewswitch)
            {
                this[65] = RollWiimoteAngle < -19 & Fbool["//stick arrows"];
                this[56] = (mWSNunchuckStateRawJoystickX > 33f & !Fbool["//wheel script"])
| ((mWSNunchuckStateRawJoystickX >= Fvar["//wheel script stick limit in"] | bool2swps) &
Fbool["//wheel script"]);
                this[57] = (mWSNunchuckStateRawJoystickX < -33f & !Fbool["//wheel script"])
| ((mWSNunchuckStateRawJoystickX <= -Fvar["//wheel script stick limit in"] | bool2swms) &
Fbool["//wheel script"]);
                this[58] = (mWSNunchuckStateRawJoystickY > 33f & !Fbool["//wheel script"])
| ((mWSNunchuckStateRawJoystickY >= Fvar["//wheel script stick limit in"] | bool2swyps) &
Fbool["//wheel script"]);
                this[59] = (mWSNunchuckStateRawJoystickY < -33f & !Fbool["//wheel script"])
| ((mWSNunchuckStateRawJoystickY <= -Fvar["//wheel script stick limit in"] | bool2swyms) &
Fbool["//wheel script"]);
                if ((RollWiimoteAngle >= -19 & Fbool["//stick arrows"]) | !Fbool["//stick
arrows"])
                {
                    if (_Valuechanged[56] & this[56])
                    {
                        if (action["//nunchuck stick"] == "WASD")
                            doSimulateKeyDown(VK_D, S_D);
                        if (action["//nunchuck stick"] == "ZQSD")
                            doSimulateKeyDown(VK_D, S_D);
                        if (action["//nunchuck stick"] == "arrow keys")
                            doSimulateKeyDownArrows(VK_RIGHT, S_RIGHT);
                    }
                    if (_Valuechanged[56] & !this[56])
                    {
```

```
            if (action["//nunchuck stick"] == "WASD")
                doSimulateKeyUp(VK_D, S_D);
            if (action["//nunchuck stick"] == "ZQSD")
                doSimulateKeyUp(VK_D, S_D);
            if (action["//nunchuck stick"] == "arrow keys")
                doSimulateKeyUpArrows(VK_RIGHT, S_RIGHT);
        }
        if (_Valuechanged[57] & this[57])
        {
            if (action["//nunchuck stick"] == "WASD")
                doSimulateKeyDown(VK_A, S_A);
            if (action["//nunchuck stick"] == "ZQSD")
                doSimulateKeyDown(VK_Q, S_Q);
            if (action["//nunchuck stick"] == "arrow keys")
                doSimulateKeyDownArrows(VK_LEFT, S_LEFT);
        }
        if (_Valuechanged[57] & !this[57])
        {
            if (action["//nunchuck stick"] == "WASD")
                doSimulateKeyUp(VK_A, S_A);
            if (action["//nunchuck stick"] == "ZQSD")
                doSimulateKeyUp(VK_Q, S_Q);
            if (action["//nunchuck stick"] == "arrow keys")
                doSimulateKeyUpArrows(VK_LEFT, S_LEFT);
        }
        if (_Valuechanged[58] & this[58])
        {
            if (action["//nunchuck stick"] == "WASD")
                doSimulateKeyDown(VK_W, S_W);
            if (action["//nunchuck stick"] == "ZQSD")
                doSimulateKeyDown(VK_Z, S_Z);
            if (action["//nunchuck stick"] == "arrow keys")
                doSimulateKeyDownArrows(VK_UP, S_UP);
        }
        if (_Valuechanged[58] & !this[58])
        {
            if (action["//nunchuck stick"] == "WASD")
                doSimulateKeyUp(VK_W, S_W);
            if (action["//nunchuck stick"] == "ZQSD")
                doSimulateKeyUp(VK_Z, S_Z);
            if (action["//nunchuck stick"] == "arrow keys")
                doSimulateKeyUpArrows(VK_UP, S_UP);
        }
        if (_Valuechanged[59] & this[59])
        {
            if (action["//nunchuck stick"] == "WASD")
                doSimulateKeyDown(VK_S, S_S);
            if (action["//nunchuck stick"] == "ZQSD")
                doSimulateKeyDown(VK_S, S_S);
            if (action["//nunchuck stick"] == "arrow keys")
                doSimulateKeyDownArrows(VK_DOWN, S_DOWN);
        }
        if (_Valuechanged[59] & !this[59])
        {
            if (action["//nunchuck stick"] == "WASD")
                doSimulateKeyUp(VK_S, S_S);
            if (action["//nunchuck stick"] == "ZQSD")
                doSimulateKeyUp(VK_S, S_S);
            if (action["//nunchuck stick"] == "arrow keys")
                doSimulateKeyUpArrows(VK_DOWN, S_DOWN);
        }
    }
    else
    {
        if (_Valuechanged[56] & this[56])
            doSimulateKeyDownArrows(VK_RIGHT, S_RIGHT);
```

```
                    if (_Valuechanged[56] & !this[56])
                        doSimulateKeyUpArrows(VK_RIGHT, S_RIGHT);
                    if (_Valuechanged[57] & this[57])
                        doSimulateKeyDownArrows(VK_LEFT, S_LEFT);
                    if (_Valuechanged[57] & !this[57])
                        doSimulateKeyUpArrows(VK_LEFT, S_LEFT);
                    if (_Valuechanged[58] & this[58])
                        doSimulateKeyDownArrows(VK_UP, S_UP);
                    if (_Valuechanged[58] & !this[58])
                        doSimulateKeyUpArrows(VK_UP, S_UP);
                    if (_Valuechanged[59] & this[59])
                        doSimulateKeyDownArrows(VK_DOWN, S_DOWN);
                    if (_Valuechanged[59] & !this[59])
                        doSimulateKeyUpArrows(VK_DOWN, S_DOWN);
                }
                if (_Valuechanged[65] & this[65])
                {
                    if (action["//nunchuck stick"] == "WASD")
                        doSimulateKeyUp(VK_D, S_D);
                    if (action["//nunchuck stick"] == "ZQSD")
                        doSimulateKeyUp(VK_D, S_D);
                    if (action["//nunchuck stick"] == "arrow keys")
                        doSimulateKeyUpArrows(VK_RIGHT, S_RIGHT);
                    if (action["//nunchuck stick"] == "WASD")
                        doSimulateKeyUp(VK_A, S_A);
                    if (action["//nunchuck stick"] == "ZQSD")
                        doSimulateKeyUp(VK_Q, S_Q);
                    if (action["//nunchuck stick"] == "arrow keys")
                        doSimulateKeyUpArrows(VK_LEFT, S_LEFT);
                    if (action["//nunchuck stick"] == "WASD")
                        doSimulateKeyUp(VK_W, S_W);
                    if (action["//nunchuck stick"] == "ZQSD")
                        doSimulateKeyUp(VK_Z, S_Z);
                    if (action["//nunchuck stick"] == "arrow keys")
                        doSimulateKeyUpArrows(VK_UP, S_UP);
                    if (action["//nunchuck stick"] == "WASD")
                        doSimulateKeyUp(VK_S, S_S);
                    if (action["//nunchuck stick"] == "ZQSD")
                        doSimulateKeyUp(VK_S, S_S);
                    if (action["//nunchuck stick"] == "arrow keys")
                        doSimulateKeyUpArrows(VK_DOWN, S_DOWN);
                }
                if (_Valuechanged[65] & !this[65])
                {
                    doSimulateKeyUpArrows(VK_RIGHT, S_RIGHT);
                    doSimulateKeyUpArrows(VK_LEFT, S_LEFT);
                    doSimulateKeyUpArrows(VK_UP, S_UP);
                    doSimulateKeyUpArrows(VK_DOWN, S_DOWN);
                }
            }
            if (pushtodown >= 1)
            {
                pushtodown++;
                if (pushtodown == 2)
                    miceevent(actionassign["//nunchuck to down 2"].X,
actionassign["//nunchuck to down 2"].Y);
                if (pushtodown >= Fvar["//second nunchuck to down push v time extra
setting"] / (Fvar["//tick time"] < 1f ? watchK : Fvar["//tick time"] * watchK))
                {
                    miceeventf(actionassign["//nunchuck to down 2"].X,
actionassign["//nunchuck to down 2"].Y);
                    pushtodown = 0;
                }
            }
```

```
            this[62] = (mWSNunchuckStateRawValuesY) > 33f & !((mWSRawValuesZ > 0 ?
mWSRawValuesZ : -mWSRawValuesZ) >= 40f & (mWSRawValuesY > 0 ? mWSRawValuesY : -
mWSRawValuesY) >= 40f & (mWSRawValuesX > 0 ? mWSRawValuesX : -mWSRawValuesX) >= 40f);
            if (_Valuechanged[62] & this[62])
            {
                if (keystodown == 0)
                    miceevent(actionassign["//nunchuck to down"].X,
actionassign["//nunchuck to down"].Y);
                if (keystodown == 1)
                {
                    miceevent(actionassign["//nunchuck to down 2"].X,
actionassign["//nunchuck to down 2"].Y);
                    nunchucktodown2released = false;
                }
                if (keystodown == 2)
                {
                    miceevent(actionassign["//nunchuck to down 3"].X,
actionassign["//nunchuck to down 3"].Y);
                    nunchucktodown2released = false;
                }
            }
            if (_Valuechanged[62] & !this[62])
            {
                if (keystodown == 0)
                {
                    miceeventf(actionassign["//nunchuck to down"].X,
actionassign["//nunchuck to down"].Y);
                    keystodown = 1;
                }
                else
                {
                    if (keystodown == 1)
                    {
                        miceeventf(actionassign["//nunchuck to down 2"].X,
actionassign["//nunchuck to down 2"].Y);
                        if (actionassign["//nunchuck to down"].X ==
actionassign["//nunchuck to down 3"].X)
                            keystodown = 0;
                        else
                            keystodown = 2;
                        pushtodown = 1;
                        nunchucktodown2released = true;
                    }
                    else
                    {
                        if (keystodown == 2)
                        {
                            miceeventf(actionassign["//nunchuck to down 3"].X,
actionassign["//nunchuck to down 3"].Y);
                            keystodown = 0;
                            nunchucktodown2released = true;
                        }
                    }
                }
            }
            if (nunchucktodown2released & (mWSNunchuckStateZ | (mWSNunchuckStateC &
!Fbool["//A C swap"]) | (mWSButtonStateA & Fbool["//A C swap"])))
                keystodown = 0;
            if (!Fbool["//no roll qe"])
            {
                this[60] = mWSNunchuckStateRawValuesX > 41 & ((mWSButtonStateA & Fbool["//A
roll qe"]) | !Fbool["//A roll qe"]);
                if (_Valuechanged[60] & this[60])
                    miceevent(actionassign["//wiimote nunchuck roll right"].X,
actionassign["//wiimote nunchuck roll right"].Y);
                if (_Valuechanged[60] & !this[60])
```

```
                    miceeventf(actionassign["//wiimote nunchuck roll right"].X,
actionassign["//wiimote nunchuck roll right"].Y);
                this[61] = mWSNunchuckStateRawValuesX < -41 & ((mWSButtonStateA &
Fbool["//A roll qe"]) | !Fbool["//A roll qe"]);
                if (_Valuechanged[61] & this[61])
                    miceevent(actionassign["//wiimote nunchuck roll left"].X,
actionassign["//wiimote nunchuck roll left"].Y);
                if (_Valuechanged[61] & !this[61])
                    miceeventf(actionassign["//wiimote nunchuck roll left"].X,
actionassign["//wiimote nunchuck roll left"].Y);
            }
        }
        public static void desktopcursorposition(int X, int Y)
        {
            System.Windows.Forms.Cursor.Position = new System.Drawing.Point(X, Y);
            SetCursorPos(X, Y);
            SetPhysicalCursorPos(X, Y);
            SetCaretPos(X, Y);
        }
        private double Scale(double value, double min, double max, double minScale, double
maxScale)
        {
            double scaled = minScale + (double)(value - min) / (max - min) * (maxScale -
minScale);
            return scaled;
        }
        private void WiimoteIR()
        {
            if (!Fbool["//stick view"] & !Fbool["//wheel view"] & !Fbool["//dpad view"])
            {
                mWSIR0found = (bBuffer[6] | ((bBuffer[8] >> 4) & 0x03) << 8) > 1 &
(bBuffer[6] | ((bBuffer[8] >> 4) & 0x03) << 8) < 1023;
                mWSIR1found = (bBuffer[9] | ((bBuffer[8] >> 0) & 0x03) << 8) > 1 &
(bBuffer[9] | ((bBuffer[8] >> 0) & 0x03) << 8) < 1023;
                if (mWSIR0notfound == 0 & mWSIR1found)
                    mWSIR0notfound = 1;
                if (mWSIR0notfound == 1 & !mWSIR0found & !mWSIR1found)
                    mWSIR0notfound = 2;
                if (mWSIR0notfound == 2 & mWSIR0found)
                {
                    mWSIR0notfound = 0;
                    if (!mWSIRswitch)
                        mWSIRswitch = true;
                    else
                        mWSIRswitch = false;
                }
                if (mWSIR0notfound == 0 & mWSIR0found)
                    mWSIR0notfound = 0;
                if (mWSIR0notfound == 0 & !mWSIR0found & !mWSIR1found)
                    mWSIR0notfound = 0;
                if (mWSIR0notfound == 1 & mWSIR0found)
                    mWSIR0notfound = 0;
                if (mWSIR0found)
                {
                    mWSIRSensors0X = (bBuffer[6] | ((bBuffer[8] >> 4) & 0x03) << 8);
                    mWSIRSensors0Y = (bBuffer[7] | ((bBuffer[8] >> 6) & 0x03) << 8);
                }
                if (mWSIR1found)
                {
                    mWSIRSensors1X = (bBuffer[9] | ((bBuffer[8] >> 0) & 0x03) << 8);
                    mWSIRSensors1Y = (bBuffer[10] | ((bBuffer[8] >> 2) & 0x03) << 8);
                }
                if (mWSIRswitch)
                {
                    mWSIR0foundcam = mWSIR0found;
                    mWSIR1foundcam = mWSIR1found;
```

```
                    mWSIRSensors0Xcam = mWSIRSensors0X - 512f;
                    mWSIRSensors0Ycam = mWSIRSensors0Y - 384f;
                    mWSIRSensors1Xcam = mWSIRSensors1X - 512f;
                    mWSIRSensors1Ycam = mWSIRSensors1Y - 384f;
                }
                else
                {
                    mWSIR1foundcam = mWSIR0found;
                    mWSIR0foundcam = mWSIR1found;
                    mWSIRSensors1Xcam = mWSIRSensors0X - 512f;
                    mWSIRSensors1Ycam = mWSIRSensors0Y - 384f;
                    mWSIRSensors0Xcam = mWSIRSensors1X - 512f;
                    mWSIRSensors0Ycam = mWSIRSensors1Y - 384f;
                }
                if (mWSIR0foundcam & mWSIR1foundcam)
                {
                    irx2e = mWSIRSensors0Xcam;
                    iry2e = mWSIRSensors0Ycam;
                    irx3e = mWSIRSensors1Xcam;
                    iry3e = mWSIRSensors1Ycam;
                    mWSIRSensorsXcam = mWSIRSensors0Xcam - mWSIRSensors1Xcam;
                    mWSIRSensorsYcam = mWSIRSensors0Ycam - mWSIRSensors1Ycam;
                }
                if (mWSIR0foundcam & !mWSIR1foundcam)
                {
                    irx2e = mWSIRSensors0Xcam;
                    iry2e = mWSIRSensors0Ycam;
                    irx3e = mWSIRSensors0Xcam - mWSIRSensorsXcam;
                    iry3e = mWSIRSensors0Ycam - mWSIRSensorsYcam;
                }
                if (mWSIR1foundcam & !mWSIR0foundcam)
                {
                    irx3e = mWSIRSensors1Xcam;
                    iry3e = mWSIRSensors1Ycam;
                    irx2e = mWSIRSensors1Xcam + mWSIRSensorsXcam;
                    iry2e = mWSIRSensors1Ycam + mWSIRSensorsYcam;
                }
                MyAngle = (irx2e - irx3e > 0 ? 1f : -1f) * (iry2e - iry3e) / 6000f;
                irxc = irx2e + irx3e;
                iryc = iry2e + iry3e;
                irxe = irxc * (1f - (MyAngle > 0 ? MyAngle : -MyAngle)) + MyAngle * iryc;
                irye = iryc * (1f - (MyAngle > 0 ? MyAngle : -MyAngle)) - MyAngle * irxc;
                irxemax = 1400f - Fvar["//irxinit"];
                irxemin = -1400f - Fvar["//irxinit"];
                iryemax = 768f - Fvar["//iryinit"];
                iryemin = -768f - Fvar["//iryinit"];
                if ((Fbool["//A aim plus"] | Fbool["//A accuracy"]) & (((!Fbool["//A C
swap"] & mWSButtonStateAio) | (Fbool["//A C swap"] & mWSNunchuckStateCio)) | (((!Fbool["//A
C swap"] & mWSButtonStateA) | (Fbool["//A C swap"] & mWSNunchuckStateC)) & foraorcison)))
                {
                    aimpluscount += 1f * watchM;
                    if (aimpluscount >= Fvar["//aim plus latency time extra setting"])
                        aimpluscount = Fvar["//aim plus latency time extra setting"];
                }
                else
                {
                    aimpluscount -= 1f * watchM;
                    if (aimpluscount <= 0)
                        aimpluscount = 0;
                }
                if ((Fvar["//aim speed accuracy multipler of center axis x extra setting"]
!= 0 | Fvar["//aim speed accuracy size of center axis x extra setting"] != 0) & (irxe > 0 ?
irxe : -irxe) > Fvar["//aim speed accuracy size of center axis x extra setting"])
                    irx = irxe - Fvar["//irxinit"] >= 0 ? Scale(irxe - Fvar["//irxinit"],
0f, irxemax, (Fvar["//aim speed accuracy multipler of center axis x extra setting"] *
(Fbool["//A accuracy"] ? aimpluscount / Fvar["//aim plus latency time extra setting"] : 1))
```

```
/ 100f, 1024f) : Scale(irxe - Fvar["//irxinit"], irxemin, 0f, -1024f, -(Fvar["//aim speed
accuracy multipler of center axis x extra setting"] * (Fbool["//A accuracy"] ? aimpluscount
/ Fvar["//aim plus latency time extra setting"] : 1)) / 100f);
                else
                    irx = irxe - Fvar["//irxinit"] >= 0 ? Scale(irxe - Fvar["//irxinit"],
0f, irxemax, 0, 1024f) : Scale(irxe - Fvar["//irxinit"], irxemin, 0f, -1024f, 0);
                if ((Fvar["//aim speed accuracy multipler of center axis y extra setting"]
!= 0 | Fvar["//aim speed accuracy size of center axis y extra setting"] != 0) & (irye > 0 ?
irye : -irye) > Fvar["//aim speed accuracy size of center axis y extra setting"])
                    iry = irye - Fvar["//iryinit"] >= 0 ? Scale(irye - Fvar["//iryinit"],
0f, iryemax, (Fvar["//aim speed accuracy multipler of center axis y extra setting"] *
(Fbool["//A accuracy"] ? aimpluscount / Fvar["//aim plus latency time extra setting"] : 1))
/ 100f, 1024f) : Scale(irye - Fvar["//iryinit"], iryemin, 0f, -1024f, -(Fvar["//aim speed
accuracy multipler of center axis y extra setting"] * (Fbool["//A accuracy"] ? aimpluscount
/ Fvar["//aim plus latency time extra setting"] : 1)) / 100f);
                else
                    iry = irye - Fvar["//iryinit"] >= 0 ? Scale(irye - Fvar["//iryinit"],
0f, iryemax, 0, 1024f) : Scale(irye - Fvar["//iryinit"], iryemin, 0f, -1024f, 0);
                if (Fvar["//no recoil quantity extra setting"] != 0 & Fvar["//no recoil
step quantity"] != 0)
                {
                    mWSButtonStateBcam = (bBuffer[2] & 0x04) != 0;
                    if (mWSButtonStateBcam)
                    {
                        norecoilcount += (Fvar["//no recoil step quantity"] / 100f) *
watchM;
                        if (norecoilcount >= (Fvar["//no recoil quantity extra setting"] >
0 ? Fvar["//no recoil quantity extra setting"] : -Fvar["//no recoil quantity extra
setting"]))
                            norecoilcount = (Fvar["//no recoil quantity extra setting"] > 0
? Fvar["//no recoil quantity extra setting"] : -Fvar["//no recoil quantity extra
setting"]);
                    }
                    else
                    {
                        norecoilcount -= (Fvar["//no recoil step quantity"] / 100f) *
watchM;
                        if (norecoilcount <= 0)
                            norecoilcount = 0;
                    }
                    iryn = (Fvar["//no recoil quantity extra setting"] > 0 ? 1 : -1) *
norecoilcount;
                }
                else
                    iryn = 0;
                if (!Fbool["//A aim plus"])
                {
                    irxpp = irx;
                    irypp = iry + iryn;
                }
                else
                {
                    irxpp = irx * (100f - Fvar["//aim plus quantity extra setting"]) / 100f
+ irx * Fvar["//aim plus quantity extra setting"] / 100f * aimpluscount / Fvar["//aim plus
latency time extra setting"];
                    irypp = iry * (100f - Fvar["//aim plus quantity extra setting"]) / 100f
+ iry * Fvar["//aim plus quantity extra setting"] / 100f * aimpluscount / Fvar["//aim plus
latency time extra setting"] + iryn;
                }
            }
            else
            {
                mWSRawValuesXcam = bBuffer[3] - 135f + calibrationinit;
                mWSRawValuesYcam = bBuffer[4] - 135f + calibrationinit;
                mWSRawValuesZcam = bBuffer[5] - 135f + calibrationinit;
                mWSNunchuckStateRawJoystickXcam = bBuffer[16] - 125f + stickviewxinit;
```

```
                mWSNunchuckStateRawJoystickYcam = bBuffer[17] - 125f + stickviewyinit;
                if (Fbool["//stick view"] & !Fbool["//wheel view"])
                {
                    irxpp = -mWSNunchuckStateRawJoystickXcam * 15f;
                    if (!Fbool["//1 and 2 view"])
                        irypp = -mWSNunchuckStateRawJoystickYcam * 15f;
                }
                if (Fbool["//wheel view"] & !Fbool["//stick view"] & !Fbool["//dpad view"])
                {
                    if (!signchangewheelY)
                        signchangewheelY1 = signchangewheelY2;
                    signchangewheelY2 = mWSRawValuesYcam * 45f;
                    if (mWSRawValuesYcam * 45f > 585f | mWSRawValuesYcam * 45f < -585f)
                    {
                        if (Math.Sign(signchangewheelY1) != Math.Sign(mWSRawValuesYcam *
45f))
                        {
                            signchangewheelY2 = signchangewheelY1;
                            signchangewheelY = true;
                        }
                        else
                        {
                            signchangewheelY2 = mWSRawValuesYcam * 45f;
                            signchangewheelY = false;
                        }
                    }
                    if (!signchangewheelY)
                        irxpp = mWSRawValuesYcam * 45f;
                    if (!Fbool["//1 and 2 view"])
                    {
                        iryppv = mWSRawValuesXcam;
                        iryppmax = 23f - Fvar["//angleinit"];
                        iryppmin = -23f - Fvar["//angleinit"];
                        irypp = iryppv - Fvar["//angleinit"] >= 0 ? Scale(iryppv -
Fvar["//angleinit"], 0f, iryppmax, 0, 1024f) : Scale(iryppv - Fvar["//angleinit"],
iryppmin, 0f, -1024f, 0);
                    }
                }
                if (Fbool["//stick view"] & Fbool["//wheel view"])
                {
                    irxpp = -mWSRawValuesXcam * 45f;
                    if (!Fbool["//1 and 2 view"])
                        irypp = mWSRawValuesZcam * 45f;
                }
                if (Fbool["//dpad view"] & !Fbool["//wheel view"])
                {
                    mWSButtonStateBcam = (bBuffer[2] & 0x04) != 0;
                    mWSButtonStateUpcam = (bBuffer[1] & 0x08) != 0;
                    mWSButtonStateDowncam = (bBuffer[1] & 0x04) != 0;
                    if (mWSButtonStateUpcam & ((Upview <= 1024f & !mWSButtonStateBcam) |
(Upview <= 512f & mWSButtonStateBcam)))
                        Upview += 3f * watchM;
                    if (!mWSButtonStateUpcam & Upview >= 0)
                        Upview -= 3f * watchM;
                    if (mWSButtonStateUpcam & Upview >= 512f & mWSButtonStateBcam)
                        Upview -= 3f * watchM;
                    if (mWSButtonStateDowncam & ((Downview >= -1024f & !mWSButtonStateBcam)
| (Downview >= -512f & mWSButtonStateBcam)))
                        Downview -= 3f * watchM;
                    if (!mWSButtonStateDowncam & Downview <= 0)
                        Downview += 3f * watchM;
                    if (mWSButtonStateDowncam & Downview <= -512f & mWSButtonStateBcam)
                        Downview += 3f * watchM;
                    if (Upview <= 0)
                        Upview = 0;
                    if (Downview >= 0)
```

```
                    Downview = 0;
                if (mWSButtonStateUpcam & !mWSButtonStateDowncam)
                    Downview = 0;
                if (!mWSButtonStateUpcam & mWSButtonStateDowncam)
                    Upview = 0;
                irxpp = Upview + Downview;
                if (!Fbool["//1 and 2 view"])
                {
                    mWSButtonStateRightcam = (bBuffer[1] & 0x02) != 0;
                    mWSButtonStateLeftcam = (bBuffer[1] & 0x01) != 0;
                    if (mWSButtonStateRightcam & ((Rightview >= -1024f &
!mWSButtonStateBcam) | (Rightview >= -512f & mWSButtonStateBcam)))
                        Rightview -= 3f * watchM;
                    if (!mWSButtonStateRightcam & Rightview <= 0)
                        Rightview += 3f * watchM;
                    if (mWSButtonStateRightcam & Rightview <= -512f &
mWSButtonStateBcam)
                        Rightview += 3f * watchM;
                    if (mWSButtonStateLeftcam & ((Leftview <= 1024f &
!mWSButtonStateBcam) | (Leftview <= 512f & mWSButtonStateBcam)))
                        Leftview += 3f * watchM;
                    if (!mWSButtonStateLeftcam & Leftview >= 0)
                        Leftview -= 3f * watchM;
                    if (mWSButtonStateLeftcam & Leftview >= 512f & mWSButtonStateBcam)
                        Leftview -= 3f * watchM;
                    if (Rightview >= 0)
                        Rightview = 0;
                    if (Leftview <= 0)
                        Leftview = 0;
                    if (mWSButtonStateRightcam & !mWSButtonStateLeftcam)
                        Leftview = 0;
                    if (!mWSButtonStateRightcam & mWSButtonStateLeftcam)
                        Rightview = 0;
                    irypp = Rightview + Leftview;
                }
            }
            if (Fbool["//dpad view"] & Fbool["//wheel view"])
            {
                irxpp = mWSRawValuesYcam * 45f;
                if (!Fbool["//1 and 2 view"])
                {
                    mWSButtonStateRightcam = (bBuffer[1] & 0x02) != 0;
                    mWSButtonStateLeftcam = (bBuffer[1] & 0x01) != 0;
                    if (mWSButtonStateRightcam)
                        Rightview = -1024f;
                    if (!mWSButtonStateRightcam)
                        Rightview = 0f;
                    if (mWSButtonStateLeftcam)
                        Leftview = 1024f;
                    if (!mWSButtonStateLeftcam)
                        Leftview = 0f;
                    irypp = Rightview + Leftview;
                }
            }
            if (Fbool["//1 and 2 view"])
            {
                if (Fbool["//wheel view"] & !Fbool["//stick view"])
                {
                    mWSButtonStateOnecam = (bBuffer[2] & 0x02) != 0;
                    mWSButtonStateTwocam = (bBuffer[2] & 0x01) != 0;
                }
                if ((Fbool["//stick view"] & !Fbool["//wheel view"]) | (Fbool["//stick
view"] & Fbool["//wheel view"]))
                {
                    mWSButtonStateOnecam = (bBuffer[2] & 0x08) != 0;
                    mWSButtonStateTwocam = (bBuffer[2] & 0x04) != 0;
```

```
                    }
                    if (mWSButtonStateTwocam & Acceleration >= -1024f)
                        Acceleration -= 3f * watchM;
                    if (!mWSButtonStateTwocam & Acceleration <= 0)
                        Acceleration += 3f * watchM;
                    if (mWSButtonStateOnecam & Breaking <= 1024f)
                        Breaking += 3f * watchM;
                    if (!mWSButtonStateOnecam & Breaking >= 0)
                        Breaking -= 3f * watchM;
                    if (Acceleration >= 0)
                        Acceleration = 0;
                    if (Breaking <= 0)
                        Breaking = 0;
                    if (mWSButtonStateTwocam & !mWSButtonStateOnecam)
                        Breaking = 0;
                    if (!mWSButtonStateTwocam & mWSButtonStateOnecam)
                        Acceleration = 0;
                    irypp = Acceleration + Breaking;
                }
            }
            if (!reconfiguration & Fvar["//smooth time extra setting"] >= 2)
            {
                if (valListXn.Count >= Fvar["//smooth time extra setting"] &
valListYn.Count >= Fvar["//smooth time extra setting"])
                {
                    valListXn.RemoveAt(0);
                    valListXn.Add(irxpp);
                    mousexbn = valListXn.Average();
                    valListYn.RemoveAt(0);
                    valListYn.Add(irypp);
                    mouseybn = valListYn.Average();
                }
                else
                {
                    valListXn.Add(0);
                    valListYn.Add(0);
                }
            }
            else
            {
                mousexbn = irxpp;
                mouseybn = irypp;
            }
            if (Fvar["//anti-tearing outer size"] > 0)
                mousexi = mousexbn / (((mousexbn > 0 ? mousexbn : -mousexbn) *
Fvar["//anti-tearing outer size"] / 100f) / 1024f + (100f - Fvar["//anti-tearing outer
size"]) / 100f);
            if (Fvar["//anti-tearing outer size"] < 0)
                mousexi = mousexbn * (((mousexbn > 0 ? mousexbn : -mousexbn) * -
Fvar["//anti-tearing outer size"] / 100f) / 1024f + (100f + Fvar["//anti-tearing outer
size"]) / 100f);
            if (Fvar["//anti-tearing outer size"] == 0)
                mousexi = mousexbn;
            if (!(Fbool["//1 and 2 view"] & ((!Fbool["//stick view"] & Fbool["//wheel
view"]) | (Fbool["//stick view"] & !Fbool["//wheel view"]) | (Fbool["//stick view"] &
Fbool["//wheel view"]))))
            {
                if (Fvar["//anti-tearing outer size"] > 0)
                    mouseyi = mouseybn / (((mouseybn > 0 ? mouseybn : -mouseybn) *
Fvar["//anti-tearing outer size"] / 100f) / 1024f + (100f - Fvar["//anti-tearing outer
size"]) / 100f);
                if (Fvar["//anti-tearing outer size"] < 0)
                    mouseyi = mouseybn * (((mouseybn > 0 ? mouseybn : -mouseybn) * -
Fvar["//anti-tearing outer size"] / 100f) / 1024f + (100f + Fvar["//anti-tearing outer
size"]) / 100f);
                if (Fvar["//anti-tearing outer size"] == 0)
```

```
                mouseyi = mouseybn;
            }
            else
                mouseyi = irypp;
            mousexpn = (float)(Math.Pow(mousexi > 0 ? mousexi : -mousexi, Fvar["//zoning
quantity"] / 100f) * 1024f / Math.Pow(1024f, Fvar["//zoning quantity"] / 100f)) *
(Fvar["//aim speed axis x quantity"] / 100f) * (mousexi > 0 ? 1f : -1f);
            mouseypn = (float)(Math.Pow(mouseyi > 0 ? mouseyi : -mouseyi, Fvar["//zoning
quantity"] / 100f) * 1024f / Math.Pow(1024f, Fvar["//zoning quantity"] / 100f)) *
(Fvar["//aim speed axis y quantity"] / 100f) * (mouseyi > 0 ? 1f : -1f);
            mousexpm = (float)(Math.Pow(mousexi > 0 ? mousexi : -mousexi, Fvar["//zoning
hardness quantity"] / 100f) * 1024f / Math.Pow(1024f, Fvar["//zoning hardness quantity"] /
100f)) * (Fvar["//aim speed axis x quantity"] / 100f) * (Fvar["//hardness quantity"] /
100f) * (mousexi > 0 ? 1f : -1f);
            mouseypm = (float)(Math.Pow(mouseyi > 0 ? mouseyi : -mouseyi, Fvar["//zoning
hardness quantity"] / 100f) * 1024f / Math.Pow(1024f, Fvar["//zoning hardness quantity"] /
100f)) * (Fvar["//aim speed axis y quantity"] / 100f) * (Fvar["//hardness quantity"] /
100f) * (mouseyi > 0 ? 1f : -1f);
            if (Fbool["//brink"] | Fbool["//titanfall"])
            {
                brinktitanfalltimecount += watchM;
                if (brinktitanfalltimecount >= (Fvar["//brink or titanfall time extra
setting"] / watchM))
                {
                    if (Fbool["//brink"])
                        doMouseBrink((int)(-mousexpn / 4), (int)(mouseypn / 4));
                    if (Fbool["//titanfall"])
                        doMouseMW3((int)(-mousexpn), (int)(mouseypn));
                    brinktitanfalltimecount = 0;
                }
            }
            if (Fbool["//bo3"])
            {
                bo3timecount += watchM;
                if (bo3timecount >= (Fvar["//bo3 time extra setting"]))
                {
                    doMouseMW3((int)(-mousexpn / 2), (int)(mouseypn / 2));
                    bo3timecount = 0;
                }
            }
            if (Fbool["//fake"])
                doMouseMW3((int)(32767.5f - mousexpm), (int)(mouseypm + 32767.5f));
            if (Fbool["//metro"])
            {
                SumX = mousexpp;
                SumY = mouseypp;
                doMouseMW3((int)(32767.5f - mousexpm - mousexpp), (int)(mouseypm + mouseypp
+ 32767.5f));
            }
            if (Fbool["//xaim"])
            {
                if (!mWSIR0foundcam & !mWSIR1foundcam)
                {
                    SumX = mousexpp;
                    SumY = mouseypp;
                }
                doMouseMW3((int)(32767.5f - mousexpm - mousexpp), (int)(mouseypm + mouseypp
+ 32767.5f));
            }
            if (Fbool["//cursor"])
                desktopcursorposition((int)(WidthS - mousexpn * WidthS / 1024f),
(int)(HeightS + mouseypn * HeightS / 1024f));
            if (Fbool["//warface"])
            {
                desktopcursorposition((int)(WidthS + mousexpn * WidthS / 1024f),
(int)(HeightS - mouseypn * HeightS / 1024f));
```

```
                doMouseMW3((int)(32767.5f - mousexpn * 32f), (int)(mouseypn * 32f +
32767.5f));
            }
            if (Fbool["//mw3"])
                doMouseMW3((int)(32767.5f - mousexpn * 32f), (int)(mouseypn * 32f +
32767.5f));
        }
        private void Wiimote()
        {
            ticktimecount++;
            if (!reconfiguration & ticktimecount >= (Fvar["//tick time"] < 1 ? 1 :
Fvar["//tick time"]))
            {
                if (readingfilecount == 0)
                    readingfile = false;
                readingfilecount += watchK;
                if (readingfilecount > 100f / (Fvar["//tick time"] < 1f ? 1 : Fvar["//tick
time"]))
                {
                    if (mWSIR0found & mWSIR1found & mWSButtonStateOne & mWSButtonStateTwo)
                    {
                        centercursorposcount++;
                        if (centercursorposcount == 1)
                        {
                            WidthS = System.Windows.Forms.Screen.PrimaryScreen.Bounds.Width
/ 2;
                            HeightS =
System.Windows.Forms.Screen.PrimaryScreen.Bounds.Height / 2;
                        }
                        if (centercursorposcount > 20)
                        {
                            Fvar["//angleinit"] = bBuffer[3] - 135f + calibrationinit;
                            Fvar["//irxinit"] = irxc * (1f - (MyAngle > 0 ? MyAngle : -
MyAngle)) + MyAngle * iryc;
                            Fvar["//iryinit"] = iryc * (1f - (MyAngle > 0 ? MyAngle : -
MyAngle)) - MyAngle * irxc;
                            txtBangleinit.Text = ((int)Fvar["//angleinit"]).ToString();
                            txtBirxinit.Text = ((int)Fvar["//irxinit"]).ToString();
                            txtBiryinit.Text = ((int)Fvar["//iryinit"]).ToString();
                            centercursorposcount = 0;
                        }
                    }
                    else
                        centercursorposcount = 0;
                    if (Fbool["//nunchuck"])
                    {
                        Rollwiimoteanglechanged = mWSRawValuesX;
                        if (!_Rollwiimoteanglechanged)
                            RollWiimoteAngle = mWSRawValuesX;
                    }
                    SelectOptions();
                    if (!readingfile & !runningoff)
                    {
                        WiimoteFound(path);
                        switchwheelfix();
                    }
                    readingfilecount = 0;
                }
                mWSButtonStateA = (bBuffer[2] & 0x08) != 0;
                mWSButtonStateB = (bBuffer[2] & 0x04) != 0;
                mWSButtonStateMinus = (bBuffer[2] & 0x10) != 0;
                mWSButtonStateHome = (bBuffer[2] & 0x80) != 0;
                mWSButtonStatePlus = (bBuffer[1] & 0x10) != 0;
                mWSButtonStateOne = (bBuffer[2] & 0x02) != 0;
                mWSButtonStateTwo = (bBuffer[2] & 0x01) != 0;
                mWSButtonStateUp = (bBuffer[1] & 0x08) != 0;
```

```
                mWSButtonStateDown = (bBuffer[1] & 0x04) != 0;
                mWSButtonStateLeft = (bBuffer[1] & 0x01) != 0;
                mWSButtonStateRight = (bBuffer[1] & 0x02) != 0;
                mWSRawValuesX = bBuffer[3] - 135f + calibrationinit;
                mWSRawValuesY = bBuffer[4] - 135f + calibrationinit;
                mWSRawValuesZ = bBuffer[5] - 135f + calibrationinit;
                mWSNunchuckStateRawJoystickX = bBuffer[16] - 125f + stickviewxinit;
                mWSNunchuckStateRawJoystickY = bBuffer[17] - 125f + stickviewyinit;
                mWSNunchuckStateRawValuesX = bBuffer[18] - 125f;
                mWSNunchuckStateRawValuesY = bBuffer[19] - 125f;
                mWSNunchuckStateRawValuesZ = bBuffer[20] - 125f;
                mWSNunchuckStateC = (bBuffer[21] & 0x02) == 0;
                mWSNunchuckStateZ = (bBuffer[21] & 0x01) == 0;
                foraorcison = (mWSButtonStateMinus | mWSButtonStatePlus |
mWSButtonStateHome | ((mWSRawValuesZ > 0 ? mWSRawValuesZ : -mWSRawValuesZ) >= 40f &
(mWSRawValuesY > 0 ? mWSRawValuesY : -mWSRawValuesY) >= 40f & (mWSRawValuesX > 0 ?
mWSRawValuesX : -mWSRawValuesX) >= 40f) | (mWSNunchuckStateRawValuesY > 33f &
Fbool["//nunchuck"]) | (mWSNunchuckStateRawJoystickY > 33f & mWSNunchuckStateZ &
Fbool["//nunchuck"]) | mWSButtonStateUp | mWSButtonStateDown | mWSButtonStateLeft |
mWSButtonStateRight);
                if (!Fbool["//A C swap"])
                {
                    if (!Fbool["//A press I/O"])
                        mWSButtonStateAio = mWSButtonStateA;
                    else
                    {
                        this[64] = mWSButtonStateA;
                        if (_Valuechanged[64] & this[64])
                            if (!randA)
                            {
                                mWSButtonStateAio = true;
                                randA = true;
                            }
                            else
                                if (randA)
                                {
                                    mWSButtonStateAio = false;
                                    randA = false;
                                }
                        if (mWSButtonStateAio & foraorcison & !Fbool["//A view on"])
                        {
                            mWSButtonStateAio = false;
                            randA = false;
                        }
                    }
                    mWSNunchuckStateCio = mWSNunchuckStateC;
                }
                WiimoteABMinusPlusTofront();
                if (!Fbool["//nunchuck"])
                {
                    WiimoteAloneAnalogic();
                    WiimoteAloneButtons();
                }
                else
                {
                    if (Fbool["//A C swap"])
                    {
                        if (!Fbool["//A press I/O"])
                            mWSNunchuckStateCio = mWSNunchuckStateC;
                        else
                        {
                            this[77] = mWSNunchuckStateC;
                            if (_Valuechanged[77] & this[77])
                                if (!randC)
                                {
                                    mWSNunchuckStateCio = true;
```

```
                                    randC = true;
                                }
                                else
                                    if (randC)
                                    {
                                        mWSNunchuckStateCio = false;
                                        randC = false;
                                    }
                                if (mWSNunchuckStateCio & foraorcison & !Fbool["//A view on"])
                                {
                                    mWSNunchuckStateCio = false;
                                    randC = false;
                                }
                            }
                            mWSButtonStateAio = mWSButtonStateA;
                        }
                        if (!Fbool["//Z press I/O"])
                            mWSNunchuckStateZio = mWSNunchuckStateZ;
                        else
                        {
                            this[88] = mWSNunchuckStateZ;
                            if (_Valuechanged[88] & this[88])
                                if (!randZ)
                                {
                                    mWSNunchuckStateZio = true;
                                    randZ = true;
                                }
                                else
                                    if (randZ)
                                    {
                                        mWSNunchuckStateZio = false;
                                        randZ = false;
                                    }
                            if (mWSNunchuckStateZio & ((mWSButtonStateA & !Fbool["//A C swap"])
| (mWSNunchuckStateC & Fbool["//A C swap"])))
                            {
                                mWSNunchuckStateZio = false;
                                randZ = false;
                            }
                        }
                        WiimoteNunchuckAnalogic();
                        WiimoteNunchuckButtons();
                    }
                    ticktimecount = 0;
                }
            }
            private void Wiimote_thrK()
            {
                for (; ; )
                {
                    if (runningoff)
                        return;
                    watchK2 = (double)diffK.ElapsedTicks / (Stopwatch.Frequency / (1000L *
1000L));
                    watchK = (watchK2 - watchK1) / 1000f;
                    watchK1 = watchK2;
                    if (Getstate)
                        Wiimote();
                    else
                    {
                        signchangewheelY = false;
                        SelectOptions();
                    }
                    Thread.Sleep(5);
                }
            }
```

```
        private void Wiimote_thrM()
        {
            for (; ; )
            {
                if (runningoff)
                    return;
                watchM2 = (double)diffM.ElapsedTicks / (Stopwatch.Frequency / (1000L *
1000L));
                watchM = (watchM2 - watchM1) / 1000f;
                watchM1 = watchM2;
                if (Getstate)
                    WiimoteIR();
                Thread.Sleep(1);
            }
        }
        public double SumX
        {
            get { return mousexpp; }
            set { mousexpp = value + mousexpn * watchM / 160f; }
        }
        public double SumY
        {
            get { return mouseypp; }
            set { mouseypp = value + mouseypn * watchM / 160f; }
        }
        protected internal Form1()
        {
            InitializeComponent();
        }
        private void Form1_Shown(object sender, EventArgs e)
        {
            TimeBeginPeriod(1);
            NtSetTimerResolution(1, true, ref CurrentResolution);
            System.Diagnostics.Process process = Process.GetCurrentProcess();
            process.PriorityClass = System.Diagnostics.ProcessPriorityClass.RealTime;
            backgroundWorkerS.DoWork += new DoWorkEventHandler(FormStart);
            backgroundWorkerS.RunWorkerAsync();
        }
        private void FormStart(object sender, DoWorkEventArgs e)
        {
            this.Location = new System.Drawing.Point(100, 50);
            txtBangleinit.Text = "0";
            txtBirxinit.Text = "0";
            txtBiryinit.Text = "0";
            txtBwiimotetofrontpush.Text = "1000";
            txtBdoubleapush.Text = "1000";
            txtBcancelreload.Text = "1800";
            txtBsecondnunchucktodown.Text = "500";
            txtBbrinkortitanfall.Text = "30";
            txtBbo3.Text = "15";
            txtBsmooth.Text = "15";
            txtBaimpluslatency.Text = "300";
            txtBaimplusquantity.Text = "30";
            txtBantitearingoutersize.Text = "0";
            txtBhardnessquantity.Text = "100";
            txtBaimspeedaxisxquantity.Text = "100";
            txtBaimspeedaxisyquantity.Text = "100";
            txtBaimspeedaccuracysizex.Text = "0";
            txtBaimspeedaccuracymultiplerx.Text = "0";
            txtBaimspeedaccuracysizey.Text = "0";
            txtBaimspeedaccuracymultiplery.Text = "0";
            txtBnorecoilquantity.Text = "0";
            txtB1tabswitchinterval.Text = "60";
            txtB1tabswitchpressdelay.Text = "10";
            txtBticktime.Text = "6";
            txtBwheelscriptsticklimitin.Text = "33";
```

```
txtBwheelscriptsticklimitout.Text = "2000";
txtBwheelscriptgyroscopelimitin.Text = "20";
txtBwheelscriptgyroscopelimitout.Text = "1200";
txtBzoningquantity.Text = "100";
txtBzoninghardnessquantity.Text = "100";
txtBnorecoilstepquantity.Text = "0";
txtBslideinit.Text = "450";
txtBslidetime.Text = "900";
Fvar.Add("//angleinit", 0);
Fvar.Add("//irxinit", 0);
Fvar.Add("//iryinit", 0);
Fvar.Add("//wiimote to front push r time extra setting", 1000);
Fvar.Add("//double A push r time extra setting", 1000);
Fvar.Add("//cancel reload waiting A or C time extra setting", 1800);
Fvar.Add("//second nunchuck to down push v time extra setting", 500);
Fvar.Add("//brink or titanfall time extra setting", 30);
Fvar.Add("//bo3 time extra setting", 15);
Fvar.Add("//smooth time extra setting", 15);
Fvar.Add("//aim plus latency time extra setting", 300);
Fvar.Add("//aim plus quantity extra setting", 30);
Fvar.Add("//anti-tearing outer size", 0);
Fvar.Add("//hardness quantity", 100);
Fvar.Add("//aim speed axis x quantity", 100);
Fvar.Add("//aim speed axis y quantity", 100);
Fvar.Add("//aim speed accuracy size of center axis x extra setting", 0);
Fvar.Add("//aim speed accuracy multipler of center axis x extra setting", 0);
Fvar.Add("//aim speed accuracy size of center axis y extra setting", 0);
Fvar.Add("//aim speed accuracy multipler of center axis y extra setting", 0);
Fvar.Add("//no recoil quantity extra setting", 0);
Fvar.Add("//1 tab switch interval time extra setting", 60);
Fvar.Add("//1 tab switch press delay time extra setting", 10);
Fvar.Add("//tick time", 6);
Fvar.Add("//wheel script stick limit in", 33);
Fvar.Add("//wheel script stick limit out", 2000);
Fvar.Add("//wheel script gyroscope limit in", 20);
Fvar.Add("//wheel script gyroscope limit out", 1200);
Fvar.Add("//zoning quantity", 100);
Fvar.Add("//zoning hardness quantity", 100);
Fvar.Add("//no recoil step quantity", 0);
Fvar.Add("//slide init", 450);
Fvar.Add("//slide time", 900);
Fbool.Add("//rebind keys", false);
Fbool.Add("//lock features and options", false);
Fbool.Add("//brink", false);
Fbool.Add("//metro", false);
Fbool.Add("//titanfall", false);
Fbool.Add("//cursor", false);
Fbool.Add("//A press I/O", false);
Fbool.Add("//warface", false);
Fbool.Add("//bo3", false);
Fbool.Add("//fake", false);
Fbool.Add("//mw3", false);
Fbool.Add("//wheel script", false);
Fbool.Add("//A accuracy", false);
Fbool.Add("//no roll qe", false);
Fbool.Add("//Home f only", false);
Fbool.Add("//A view on", false);
Fbool.Add("//1 tab I/O", false);
Fbool.Add("//A aim plus", false);
Fbool.Add("//xaim", false);
Fbool.Add("//Home -> - + +", false);
Fbool.Add("//1 and 2 view", false);
Fbool.Add("//nunchuck", false);
Fbool.Add("//wheel view", false);
Fbool.Add("//stick view", false);
Fbool.Add("//cancel reload x", false);
```

```
Fbool.Add("//double A r", false);
Fbool.Add("//1 tab switch", false);
Fbool.Add("//1 B swap", false);
Fbool.Add("//A C swap", false);
Fbool.Add("//push r 1'", false);
Fbool.Add("//A slide A+B", false);
Fbool.Add("//dpad view", false);
Fbool.Add("//A B switch", false);
Fbool.Add("//Z press I/O", false);
Fbool.Add("//driver mouse", false);
Fbool.Add("//driver keyboard", false);
Fbool.Add("//stick arrows", false);
Fbool.Add("//A roll qe", false);
Abool.Add("//brink", false);
Abool.Add("//metro", false);
Abool.Add("//titanfall", false);
Abool.Add("//cursor", false);
Abool.Add("//A press I/O", false);
Abool.Add("//warface", false);
Abool.Add("//bo3", false);
Abool.Add("//fake", false);
Abool.Add("//mw3", false);
Abool.Add("//wheel script", false);
Abool.Add("//A accuracy", false);
Abool.Add("//no roll qe", false);
Abool.Add("//Home f only", false);
Abool.Add("//A view on", false);
Abool.Add("//1 tab I/O", false);
Abool.Add("//A aim plus", false);
Abool.Add("//xaim", false);
Abool.Add("//Home -> - + +", false);
Abool.Add("//1 and 2 view", false);
Abool.Add("//nunchuck", false);
Abool.Add("//wheel view", false);
Abool.Add("//stick view", false);
Abool.Add("//cancel reload x", false);
Abool.Add("//double A r", false);
Abool.Add("//1 tab switch", false);
Abool.Add("//1 B swap", false);
Abool.Add("//A C swap", false);
Abool.Add("//push r 1'", false);
Abool.Add("//A slide A+B", false);
Abool.Add("//dpad view", false);
Abool.Add("//A B switch", false);
Abool.Add("//Z press I/O", false);
Abool.Add("//driver mouse", false);
Abool.Add("//driver keyboard", false);
Abool.Add("//stick arrows", false);
Abool.Add("//A roll qe", false);
action.Add("//cancel reload x", "");
action.Add("//double A r", "");
action.Add("//1 tab switch", "");
action.Add("//nunchuck stick", "");
action.Add("//wiimote alone wheel", "");
action.Add("//wiimote alone up", "");
action.Add("//wiimote alone down", "");
action.Add("//wiimote b", "");
action.Add("//wiimote a", "");
action.Add("//wiimote plus", "");
action.Add("//wiimote minus", "");
action.Add("//wiimote alone roll left", "");
action.Add("//wiimote alone home", "");
action.Add("//wiimote alone a and b", "");
action.Add("//wiimote alone left", "");
action.Add("//wiimote alone right", "");
action.Add("//wiimote alone 1", "");
```

```
action.Add("//wiimote alone 2", "");
action.Add("//wiimote nunchuck roll right", "");
action.Add("//wiimote nunchuck roll left", "");
action.Add("//wiimote to front", "");
action.Add("//nunchuck to down", "");
action.Add("//nunchuck to down 2", "");
action.Add("//nunchuck to down 3", "");
action.Add("//nunchuck z and c", "");
action.Add("//nunchuck c", "");
action.Add("//nunchuck z", "");
action.Add("//wiimote nunchuck down", "");
action.Add("//wiimote nunchuck home", "");
action.Add("//wiimote nunchuck left", "");
action.Add("//wiimote nunchuck right", "");
action.Add("//wiimote nunchuck up", "");
action.Add("//wiimote nunchuck 1", "");
action.Add("//wiimote nunchuck 2", "");
actionassign.Add("//cancel reload x", new Point2D());
actionassign.Add("//double A r", new Point2D());
actionassign.Add("//1 tab switch", new Point2D());
actionassign.Add("//nunchuck stick", new Point2D());
actionassign.Add("//wiimote alone wheel", new Point2D());
actionassign.Add("//wiimote alone up", new Point2D());
actionassign.Add("//wiimote alone down", new Point2D());
actionassign.Add("//wiimote b", new Point2D());
actionassign.Add("//wiimote a", new Point2D());
actionassign.Add("//wiimote plus", new Point2D());
actionassign.Add("//wiimote minus", new Point2D());
actionassign.Add("//wiimote alone roll left", new Point2D());
actionassign.Add("//wiimote alone home", new Point2D());
actionassign.Add("//wiimote alone a and b", new Point2D());
actionassign.Add("//wiimote alone left", new Point2D());
actionassign.Add("//wiimote alone right", new Point2D());
actionassign.Add("//wiimote alone 1", new Point2D());
actionassign.Add("//wiimote alone 2", new Point2D());
actionassign.Add("//wiimote nunchuck roll right", new Point2D());
actionassign.Add("//wiimote nunchuck roll left", new Point2D());
actionassign.Add("//wiimote to front", new Point2D());
actionassign.Add("//nunchuck to down", new Point2D());
actionassign.Add("//nunchuck to down 2", new Point2D());
actionassign.Add("//nunchuck to down 3", new Point2D());
actionassign.Add("//nunchuck z and c", new Point2D());
actionassign.Add("//nunchuck c", new Point2D());
actionassign.Add("//nunchuck z", new Point2D());
actionassign.Add("//wiimote nunchuck down", new Point2D());
actionassign.Add("//wiimote nunchuck home", new Point2D());
actionassign.Add("//wiimote nunchuck left", new Point2D());
actionassign.Add("//wiimote nunchuck right", new Point2D());
actionassign.Add("//wiimote nunchuck up", new Point2D());
actionassign.Add("//wiimote nunchuck 1", new Point2D());
actionassign.Add("//wiimote nunchuck 2", new Point2D());
do
    System.Threading.Thread.Sleep(1);
while (!connect() & !notpressing1and2);
if (!notpressing1and2)
{
    WidthS = System.Windows.Forms.Screen.PrimaryScreen.Bounds.Width / 2;
    HeightS = System.Windows.Forms.Screen.PrimaryScreen.Bounds.Height / 2;
    do
        Thread.Sleep(1);
    while (!ScanRight());
    taskD = new Task(Wiimote_thrD);
    taskD.Start();
    System.Threading.Thread.Sleep(1000);
    calibrationinit = -bBuffer[4] + 135f;
    stickviewxinit = -bBuffer[16] + 125f;
```

```
                stickviewyinit = -bBuffer[17] + 125f;
                try
                {
                    System.IO.StreamReader file = new
System.IO.StreamReader("initfile.txt");
                    file.ReadLine();
                    string pathtolastfile = file.ReadLine();
                    file.ReadLine();
                    enableautoloadoflastfile = bool.Parse(file.ReadLine());
                    file.Close();
                    if (pathtolastfile != "" & enableautoloadoflastfile)
                        openConfig(pathtolastfile);
                    else
                        Assignating();
                }
                catch
                {
                    using (System.IO.StreamWriter createdfile =
System.IO.File.AppendText("initfile.txt"))
                    {
                        createdfile.WriteLine("//path to last open or save file");
                        createdfile.WriteLine("");
                        createdfile.WriteLine("//enable autoload of last open or save
file");
                        createdfile.WriteLine("True");
                        createdfile.Close();
                        Assignating();
                    }
                }
                taskK = new Task(Wiimote_thrK);
                taskK.Start();
                diffK.Start();
                taskM = new Task(Wiimote_thrM);
                taskM.Start();
                diffM.Start();
            }
        }
        public void button1_Click(object sender, EventArgs e)//[STAThread]
        {
            switchwheelfix();
            String myRead;
            System.Windows.Forms.OpenFileDialog openFileDialog1 = new
System.Windows.Forms.OpenFileDialog();
            openFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
            openFileDialog1.FilterIndex = 2;
            openFileDialog1.RestoreDirectory = true;
            if (openFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            {
                myRead = openFileDialog1.FileName;
                openConfig(myRead);
                savePathInitFile(myRead);
            }
        }
        public void openConfig(string myRead)
        {
            reconfiguration = true;
            System.Threading.Thread.Sleep(1000);
            try
            {
                System.IO.StreamReader file = new System.IO.StreamReader(myRead);
                file.ReadLine();
                file.ReadLine();
                Abool["//brink"] = bool.Parse(file.ReadLine());
                file.ReadLine();
                Abool["//metro"] = bool.Parse(file.ReadLine());
                file.ReadLine();
```

```
Abool["//titanfall"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//cursor"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//warface"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//bo3"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//fake"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//mw3"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//xaim"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//A press I/O"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//A accuracy"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//no roll qe"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//Home f only"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//wheel script"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//A view on"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//1 tab I/O"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//A aim plus"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//Home -> - + +"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//stick view"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//wheel view"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//1 and 2 view"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//nunchuck"] = bool.Parse(file.ReadLine());
file.ReadLine();
Fbool["//rebind keys"] = bool.Parse(file.ReadLine());
file.ReadLine();
Fbool["//lock features and options"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//1 B swap"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//A C swap"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//push r 1'"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//cancel reload x"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//double A r"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//1 tab switch"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//A slide A+B"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//dpad view"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//A B switch"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//Z press I/O"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//driver mouse"] = bool.Parse(file.ReadLine());
file.ReadLine();
```

```
Abool["//driver keyboard"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//stick arrows"] = bool.Parse(file.ReadLine());
file.ReadLine();
Abool["//A roll qe"] = bool.Parse(file.ReadLine());
file.ReadLine();
action["//cancel reload x"] = file.ReadLine();
file.ReadLine();
action["//double A r"] = file.ReadLine();
file.ReadLine();
action["//1 tab switch"] = file.ReadLine();
file.ReadLine();
action["//nunchuck stick"] = file.ReadLine();
file.ReadLine();
action["//wiimote alone wheel"] = file.ReadLine();
file.ReadLine();
action["//wiimote alone up"] = file.ReadLine();
file.ReadLine();
action["//wiimote alone down"] = file.ReadLine();
file.ReadLine();
action["//wiimote b"] = file.ReadLine();
file.ReadLine();
action["//wiimote a"] = file.ReadLine();
file.ReadLine();
action["//wiimote plus"] = file.ReadLine();
file.ReadLine();
action["//wiimote minus"] = file.ReadLine();
file.ReadLine();
action["//wiimote alone roll left"] = file.ReadLine();
file.ReadLine();
action["//wiimote alone home"] = file.ReadLine();
file.ReadLine();
action["//wiimote alone a and b"] = file.ReadLine();
file.ReadLine();
action["//wiimote alone left"] = file.ReadLine();
file.ReadLine();
action["//wiimote alone right"] = file.ReadLine();
file.ReadLine();
action["//wiimote alone 1"] = file.ReadLine();
file.ReadLine();
action["//wiimote alone 2"] = file.ReadLine();
file.ReadLine();
action["//wiimote nunchuck roll right"] = file.ReadLine();
file.ReadLine();
action["//wiimote nunchuck roll left"] = file.ReadLine();
file.ReadLine();
action["//wiimote to front"] = file.ReadLine();
file.ReadLine();
action["//nunchuck to down"] = file.ReadLine();
file.ReadLine();
action["//nunchuck to down 2"] = file.ReadLine();
file.ReadLine();
action["//nunchuck to down 3"] = file.ReadLine();
file.ReadLine();
action["//nunchuck z and c"] = file.ReadLine();
file.ReadLine();
action["//nunchuck c"] = file.ReadLine();
file.ReadLine();
action["//nunchuck z"] = file.ReadLine();
file.ReadLine();
action["//wiimote nunchuck down"] = file.ReadLine();
file.ReadLine();
action["//wiimote nunchuck home"] = file.ReadLine();
file.ReadLine();
action["//wiimote nunchuck left"] = file.ReadLine();
file.ReadLine();
```

```
                action["//wiimote nunchuck right"] = file.ReadLine();
                file.ReadLine();
                action["//wiimote nunchuck up"] = file.ReadLine();
                file.ReadLine();
                action["//wiimote nunchuck 1"] = file.ReadLine();
                file.ReadLine();
                action["//wiimote nunchuck 2"] = file.ReadLine();
                file.ReadLine();
                Fvar["//angleinit"] = Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-
9-]", ""));
                file.ReadLine();
                Fvar["//irxinit"] = Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-
]", ""));
                file.ReadLine();
                Fvar["//iryinit"] = Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-
]", ""));
                file.ReadLine();
                Fvar["//wiimote to front push r time extra setting"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//double A push r time extra setting"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//cancel reload waiting A or C time extra setting"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//second nunchuck to down push v time extra setting"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//brink or titanfall time extra setting"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//bo3 time extra setting"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//smooth time extra setting"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//aim plus latency time extra setting"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//aim plus quantity extra setting"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//anti-tearing outer size"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//hardness quantity"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//aim speed axis x quantity"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//aim speed axis y quantity"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//aim speed accuracy size of center axis x extra setting"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//aim speed accuracy multipler of center axis x extra setting"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//aim speed accuracy size of center axis y extra setting"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
```

```
                Fvar["//aim speed accuracy multipler of center axis y extra setting"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//no recoil quantity extra setting"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//1 tab switch interval time extra setting"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//1 tab switch press delay time extra setting"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//tick time"] = Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-
9-]", ""));
                file.ReadLine();
                Fvar["//wheel script stick limit in"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//wheel script stick limit out"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//wheel script gyroscope limit in"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//wheel script gyroscope limit out"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//zoning quantity"] = Convert.ToSingle(Regex.Replace(file.ReadLine(),
"[^0-9-]", ""));
                file.ReadLine();
                Fvar["//zoning hardness quantity"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//no recoil step quantity"] =
Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
                file.ReadLine();
                Fvar["//slide init"] = Convert.ToSingle(Regex.Replace(file.ReadLine(),
"[^0-9-]", ""));
                file.ReadLine();
                Fvar["//slide time"] = Convert.ToSingle(Regex.Replace(file.ReadLine(),
"[^0-9-]", ""));
                file.Close();
                cmBWNSTICK.Text = action["//nunchuck stick"];
                cmBWNC.Text = action["//nunchuck c"];
                cmBWNZ.Text = action["//nunchuck z"];
                cmBWNTODOWN1.Text = action["//nunchuck to down"];
                cmBWNTODOWN2.Text = action["//nunchuck to down 2"];
                cmBWNTODOWN3.Text = action["//nunchuck to down 3"];
                cmBWNCZ.Text = action["//nunchuck z and c"];
                cmBWNB.Text = action["//wiimote b"];
                cmBWNA.Text = action["//wiimote a"];
                cmBWNMINUS.Text = action["//wiimote minus"];
                cmBWNPLUS.Text = action["//wiimote plus"];
                cmBWNTOFRONT.Text = action["//wiimote to front"];
                cmBWNDOWN.Text = action["//wiimote nunchuck down"];
                cmBWNUP.Text = action["//wiimote nunchuck up"];
                cmBWNLEFT.Text = action["//wiimote nunchuck left"];
                cmBWNRIGHT.Text = action["//wiimote nunchuck right"];
                cmBWNONE.Text = action["//wiimote nunchuck 1"];
                cmBWNTWO.Text = action["//wiimote nunchuck 2"];
                cmBWN1SWITCH.Text = action["//1 tab switch"];
                cmBWNB.Text = action["//wiimote b"];
                cmBWNA.Text = action["//wiimote a"];
                cmBWAAB.Text = action["//wiimote alone a and b"];
                cmBWNMINUS.Text = action["//wiimote minus"];
                cmBWNPLUS.Text = action["//wiimote plus"];
```

```
                cmBWNTOFRONT.Text = action["//wiimote to front"];
                cmBWAROLLLEFT.Text = action["//wiimote alone roll left"];
                cmBWADOWN.Text = action["//wiimote alone down"];
                cmBWAUP.Text = action["//wiimote alone up"];
                cmBWALEFT.Text = action["//wiimote alone left"];
                cmBWARIGHT.Text = action["//wiimote alone right"];
                cmBWAONE.Text = action["//wiimote alone 1"];
                cmBWATWO.Text = action["//wiimote alone 2"];
                cmBWN1SWITCH.Text = action["//1 tab switch"];
                if (Abool["//Home f only"])
                {
                    cmBWNHOME.Text = action["//wiimote nunchuck home"].Replace("(R)", "");
                    cmBWAHOME.Text = action["//wiimote alone home"].Replace("(R)", "");
                }
                else
                {
                    cmBWNHOME.Text = action["//wiimote nunchuck home"].Replace("(R)", "");
                    cmBWAHOME.Text = action["//wiimote alone home"].Replace("(R)", "");
                    cmBWNHOME.Text = action["//wiimote nunchuck home"] + "(R)";
                    cmBWAHOME.Text = action["//wiimote alone home"] + "(R)";
                }
                if (!Abool["//wheel script"])
                    cmBWAWHEELa.Text = " ";
                else
                    cmBWAWHEELa.Text = action["//wiimote alone wheel"];
                if (Abool["//no roll qe"])
                {
                    cmBWNROLLLEFT.Text = " ";
                    cmBWNROLLRIGHT.Text = " ";
                }
                else
                {
                    cmBWNROLLRIGHT.Text = action["//wiimote nunchuck roll right"];
                    cmBWNROLLLEFT.Text = action["//wiimote nunchuck roll left"];
                }
                if (!Abool["//cancel reload x"])
                    cmBWNCANCELRELOAD.Text = " ";
                else
                    cmBWNCANCELRELOAD.Text = action["//cancel reload x"];
                if (!Abool["//double A r"])
                    cmBWNDOUBLEA.Text = " ";
                else
                    cmBWNDOUBLEA.Text = action["//double A r"];
                txtBangleinit.Text = Fvar["//angleinit"].ToString();
                txtBirxinit.Text = Fvar["//irxinit"].ToString();
                txtBiryinit.Text = Fvar["//iryinit"].ToString();
                txtBwiimotetofrontpush.Text = Fvar["//wiimote to front push r time extra
setting"].ToString();
                txtBdoubleapush.Text = Fvar["//double A push r time extra
setting"].ToString();
                txtBcancelreload.Text = Fvar["//cancel reload waiting A or C time extra
setting"].ToString();
                txtBsecondnunchucktodown.Text = Fvar["//second nunchuck to down push v time
extra setting"].ToString();
                txtBbrinkortitanfall.Text = Fvar["//brink or titanfall time extra
setting"].ToString();
                txtBbo3.Text = Fvar["//bo3 time extra setting"].ToString();
                txtBsmooth.Text = Fvar["//smooth time extra setting"].ToString();
                txtBaimpluslatency.Text = Fvar["//aim plus latency time extra
setting"].ToString();
                txtBaimplusquantity.Text = Fvar["//aim plus quantity extra
setting"].ToString();
                txtBantitearingoutersize.Text = Fvar["//anti-tearing outer
size"].ToString();
                txtBhardnessquantity.Text = Fvar["//hardness quantity"].ToString();
```

```
                txtBaimspeedaxisxquantity.Text = Fvar["//aim speed axis x
quantity"].ToString();
                txtBaimspeedaxisyquantity.Text = Fvar["//aim speed axis y
quantity"].ToString();
                txtBaimspeedaccuracysizex.Text = Fvar["//aim speed accuracy size of center
axis x extra setting"].ToString();
                txtBaimspeedaccuracymultiplerx.Text = Fvar["//aim speed accuracy multipler
of center axis x extra setting"].ToString();
                txtBaimspeedaccuracysizey.Text = Fvar["//aim speed accuracy size of center
axis y extra setting"].ToString();
                txtBaimspeedaccuracymultiplery.Text = Fvar["//aim speed accuracy multipler
of center axis y extra setting"].ToString();
                txtBnorecoilquantity.Text = Fvar["//no recoil quantity extra
setting"].ToString();
                txtB1tabswitchinterval.Text = Fvar["//1 tab switch interval time extra
setting"].ToString();
                txtB1tabswitchpressdelay.Text = Fvar["//1 tab switch press delay time extra
setting"].ToString();
                txtBticktime.Text = Fvar["//tick time"].ToString();
                txtBwheelscriptsticklimitin.Text = Fvar["//wheel script stick limit
in"].ToString();
                txtBwheelscriptsticklimitout.Text = Fvar["//wheel script stick limit
out"].ToString();
                txtBwheelscriptgyroscopelimitin.Text = Fvar["//wheel script gyroscope limit
in"].ToString();
                txtBwheelscriptgyroscopelimitout.Text = Fvar["//wheel script gyroscope
limit out"].ToString();
                txtBzoningquantity.Text = Fvar["//zoning quantity"].ToString();
                txtBzoninghardnessquantity.Text = Fvar["//zoning hardness
quantity"].ToString();
                txtBnorecoilstepquantity.Text = Fvar["//no recoil step
quantity"].ToString();
                txtBslideinit.Text = Fvar["//slide init"].ToString();
                txtBslidetime.Text = Fvar["//slide time"].ToString();
                if (Abool["//brink"])
                    chkBF1.Checked = true;
                else
                    chkBF1.Checked = false;
                if (Fbool["//brink"] & !Abool["//brink"])
                    Abool["//brink"] = true;
                else
                    if (Fbool["//brink"] & Abool["//brink"])
                        Abool["//brink"] = false;
                if (Abool["//metro"])
                    chkBF2.Checked = true;
                else
                    chkBF2.Checked = false;
                if (Fbool["//metro"] & !Abool["//metro"])
                    Abool["//metro"] = true;
                else
                    if (Fbool["//metro"] & Abool["//metro"])
                        Abool["//metro"] = false;
                if (Abool["//titanfall"])
                    chkBF3.Checked = true;
                else
                    chkBF3.Checked = false;
                if (Fbool["//titanfall"] & !Abool["//titanfall"])
                    Abool["//titanfall"] = true;
                else
                    if (Fbool["//titanfall"] & Abool["//titanfall"])
                        Abool["//titanfall"] = false;
                if (Abool["//cursor"])
                    chkBF4.Checked = true;
                else
                    chkBF4.Checked = false;
                if (Fbool["//cursor"] & !Abool["//cursor"])
```

```
    Abool["//cursor"] = true;
else
    if (Fbool["//cursor"] & Abool["//cursor"])
        Abool["//cursor"] = false;
if (Abool["//A press I/O"])
    chkBF12.Checked = true;
else
    chkBF12.Checked = false;
if (Fbool["//A press I/O"] & !Abool["//A press I/O"])
    Abool["//A press I/O"] = true;
else
    if (Fbool["//A press I/O"] & Abool["//A press I/O"])
        Abool["//A press I/O"] = false;
if (Abool["//warface"])
    chkBF5.Checked = true;
else
    chkBF5.Checked = false;
if (Fbool["//warface"] & !Abool["//warface"])
    Abool["//warface"] = true;
else
    if (Fbool["//warface"] & Abool["//warface"])
        Abool["//warface"] = false;
if (Abool["//bo3"])
    chkBF6.Checked = true;
else
    chkBF6.Checked = false;
if (Fbool["//bo3"] & !Abool["//bo3"])
    Abool["//bo3"] = true;
else
    if (Fbool["//bo3"] & Abool["//bo3"])
        Abool["//bo3"] = false;
if (Abool["//fake"])
    chkBF7.Checked = true;
else
    chkBF7.Checked = false;
if (Fbool["//fake"] & !Abool["//fake"])
    Abool["//fake"] = true;
else
    if (Fbool["//fake"] & Abool["//fake"])
        Abool["//fake"] = false;
if (Abool["//mw3"])
    chkBF8.Checked = true;
else
    chkBF8.Checked = false;
if (Fbool["//mw3"] & !Abool["//mw3"])
    Abool["//mw3"] = true;
else
    if (Fbool["//mw3"] & Abool["//mw3"])
        Abool["//mw3"] = false;
if (Abool["//wheel script"])
    chkBF9S.Checked = true;
else
    chkBF9S.Checked = false;
if (Fbool["//wheel script"] & !Abool["//wheel script"])
    Abool["//wheel script"] = true;
else
    if (Fbool["//wheel script"] & Abool["//wheel script"])
        Abool["//wheel script"] = false;
if (Abool["//A accuracy"])
    chkBF11.Checked = true;
else
    chkBF11.Checked = false;
if (Fbool["//A accuracy"] & !Abool["//A accuracy"])
    Abool["//A accuracy"] = true;
else
    if (Fbool["//A accuracy"] & Abool["//A accuracy"])
```

```
        Abool["//A accuracy"] = false;
if (Abool["//no roll qe"])
    chkBF10S.Checked = true;
else
    chkBF10S.Checked = false;
if (Fbool["//no roll qe"] & !Abool["//no roll qe"])
    Abool["//no roll qe"] = true;
else
    if (Fbool["//no roll qe"] & Abool["//no roll qe"])
        Abool["//no roll qe"] = false;
if (Abool["//Home f only"])
    chkBF8S.Checked = true;
else
    chkBF8S.Checked = false;
if (Fbool["//Home f only"] & !Abool["//Home f only"])
    Abool["//Home f only"] = true;
else
    if (Fbool["//Home f only"] & Abool["//Home f only"])
        Abool["//Home f only"] = false;
if (Abool["//A view on"])
    chkBF1S.Checked = true;
else
    chkBF1S.Checked = false;
if (Fbool["//A view on"] & !Abool["//A view on"])
    Abool["//A view on"] = true;
else
    if (Fbool["//A view on"] & Abool["//A view on"])
        Abool["//A view on"] = false;
if (Abool["//1 tab I/O"])
    chkBF7S.Checked = true;
else
    chkBF7S.Checked = false;
if (Fbool["//1 tab I/O"] & !Abool["//1 tab I/O"])
    Abool["//1 tab I/O"] = true;
else
    if (Fbool["//1 tab I/O"] & Abool["//1 tab I/O"])
        Abool["//1 tab I/O"] = false;
if (Abool["//A aim plus"])
    chkBF2S.Checked = true;
else
    chkBF2S.Checked = false;
if (Fbool["//A aim plus"] & !Abool["//A aim plus"])
    Abool["//A aim plus"] = true;
else
    if (Fbool["//A aim plus"] & Abool["//A aim plus"])
        Abool["//A aim plus"] = false;
if (Abool["//xaim"])
    chkBF9.Checked = true;
else
    chkBF9.Checked = false;
if (Fbool["//xaim"] & !Abool["//xaim"])
    Abool["//xaim"] = true;
else
    if (Fbool["//xaim"] & Abool["//xaim"])
        Abool["//xaim"] = false;
if (Abool["//Home -> - + +"])
    chkBF11S.Checked = true;
else
    chkBF11S.Checked = false;
if (Fbool["//Home -> - + +"] & !Abool["//Home -> - + +"])
    Abool["//Home -> - + +"] = true;
else
    if (Fbool["//Home -> - + +"] & Abool["//Home -> - + +"])
        Abool["//Home -> - + +"] = false;
if (Abool["//1 and 2 view"])
    chkBF6S.Checked = true;
```

```
else
    chkBF6S.Checked = false;
if (Fbool["//1 and 2 view"] & !Abool["//1 and 2 view"])
    Abool["//1 and 2 view"] = true;
else
    if (Fbool["//1 and 2 view"] & Abool["//1 and 2 view"])
        Abool["//1 and 2 view"] = false;
if (Abool["//nunchuck"])
    chkBF12S.Checked = true;
else
    chkBF12S.Checked = false;
if (Fbool["//nunchuck"] & !Abool["//nunchuck"])
    Abool["//nunchuck"] = true;
else
    if (Fbool["//nunchuck"] & Abool["//nunchuck"])
        Abool["//nunchuck"] = false;
if (Abool["//wheel view"])
    chkBF4S.Checked = true;
else
    chkBF4S.Checked = false;
if (Fbool["//wheel view"] & !Abool["//wheel view"])
    Abool["//wheel view"] = true;
else
    if (Fbool["//wheel view"] & Abool["//wheel view"])
        Abool["//wheel view"] = false;
if (Abool["//stick view"])
    chkBF3S.Checked = true;
else
    chkBF3S.Checked = false;
if (Fbool["//stick view"] & !Abool["//stick view"])
    Abool["//stick view"] = true;
else
    if (Fbool["//stick view"] & Abool["//stick view"])
        Abool["//stick view"] = false;
if (Abool["//cancel reload x"])
    chkBF2C.Checked = true;
else
    chkBF2C.Checked = false;
if (Fbool["//cancel reload x"] & !Abool["//cancel reload x"])
    Abool["//cancel reload x"] = true;
else
    if (Fbool["//cancel reload x"] & Abool["//cancel reload x"])
        Abool["//cancel reload x"] = false;
if (Abool["//double A r"])
    chkBF3C.Checked = true;
else
    chkBF3C.Checked = false;
if (Fbool["//double A r"] & !Abool["//double A r"])
    Abool["//double A r"] = true;
else
    if (Fbool["//double A r"] & Abool["//double A r"])
        Abool["//double A r"] = false;
if (Abool["//1 tab switch"])
    chkBF4C.Checked = true;
else
    chkBF4C.Checked = false;
if (Fbool["//1 tab switch"] & !Abool["//1 tab switch"])
    Abool["//1 tab switch"] = true;
else
    if (Fbool["//1 tab switch"] & Abool["//1 tab switch"])
        Abool["//1 tab switch"] = false;
if (Abool["//1 B swap"])
    chkBF5C.Checked = true;
else
    chkBF5C.Checked = false;
if (Fbool["//1 B swap"] & !Abool["//1 B swap"])
```

```
    Abool["//1 B swap"] = true;
else
    if (Fbool["//1 B swap"] & Abool["//1 B swap"])
        Abool["//1 B swap"] = false;
if (Abool["//A C swap"])
    chkBF6C.Checked = true;
else
    chkBF6C.Checked = false;
if (Fbool["//A C swap"] & !Abool["//A C swap"])
    Abool["//A C swap"] = true;
else
    if (Fbool["//A C swap"] & Abool["//A C swap"])
        Abool["//A C swap"] = false;
if (Abool["//push r 1'"])
    chkBF1C.Checked = true;
else
    chkBF1C.Checked = false;
if (Fbool["//push r 1'"] & !Abool["//push r 1'"])
    Abool["//push r 1'"] = true;
else
    if (Fbool["//push r 1'"] & Abool["//push r 1'"])
        Abool["//push r 1'"] = false;
if (Abool["//A slide A+B"])
    chkBF7C.Checked = true;
else
    chkBF7C.Checked = false;
if (Fbool["//A slide A+B"] & !Abool["//A slide A+B"])
    Abool["//A slide A+B"] = true;
else
    if (Fbool["//A slide A+B"] & Abool["//A slide A+B"])
        Abool["//A slide A+B"] = false;
if (Abool["//dpad view"])
    chkBF5S.Checked = true;
else
    chkBF5S.Checked = false;
if (Fbool["//dpad view"] & !Abool["//dpad view"])
    Abool["//dpad view"] = true;
else
    if (Fbool["//dpad view"] & Abool["//dpad view"])
        Abool["//dpad view"] = false;
if (Abool["//A B switch"])
    chkBF8C.Checked = true;
else
    chkBF8C.Checked = false;
if (Fbool["//A B switch"] & !Abool["//A B switch"])
    Abool["//A B switch"] = true;
else
    if (Fbool["//A B switch"] & Abool["//A B switch"])
        Abool["//A B switch"] = false;
if (Abool["//Z press I/O"])
    chkBF9C.Checked = true;
else
    chkBF9C.Checked = false;
if (Fbool["//Z press I/O"] & !Abool["//Z press I/O"])
    Abool["//Z press I/O"] = true;
else
    if (Fbool["//Z press I/O"] & Abool["//Z press I/O"])
        Abool["//Z press I/O"] = false;
if (Abool["//driver mouse"])
    chkBF10.Checked = true;
else
    chkBF10.Checked = false;
if (Fbool["//driver mouse"] & !Abool["//driver mouse"])
    Abool["//driver mouse"] = true;
else
    if (Fbool["//driver mouse"] & Abool["//driver mouse"])
```

```
                    Abool["//driver mouse"] = false;
                if (Abool["//driver keyboard"])
                    chkBF12C.Checked = true;
                else
                    chkBF12C.Checked = false;
                if (Fbool["//driver keyboard"] & !Abool["//driver keyboard"])
                    Abool["//driver keyboard"] = true;
                else
                    if (Fbool["//driver keyboard"] & Abool["//driver keyboard"])
                        Abool["//driver keyboard"] = false;
                if (Abool["//stick arrows"])
                    chkBF10C.Checked = true;
                else
                    chkBF10C.Checked = false;
                if (Fbool["//stick arrows"] & !Abool["//stick arrows"])
                    Abool["//stick arrows"] = true;
                else
                    if (Fbool["//stick arrows"] & Abool["//stick arrows"])
                        Abool["//stick arrows"] = false;
                if (Abool["//A roll qe"])
                    chkBF11C.Checked = true;
                else
                    chkBF11C.Checked = false;
                if (Fbool["//A roll qe"] & !Abool["//A roll qe"])
                    Abool["//A roll qe"] = true;
                else
                    if (Fbool["//A roll qe"] & Abool["//A roll qe"])
                        Abool["//A roll qe"] = false;
                Assignating();
                clearList();
                setControlsAndOptions();
                reconfiguration = false;
            }
            catch
            {
                saveConfig("default.txt");
                savePathInitFile("default.txt");
            }
        }
        public void button2_Click(object sender, EventArgs e)
        {
            switchwheelfix();
            String charstore;
            System.Windows.Forms.SaveFileDialog saveFileDialog1 = new
System.Windows.Forms.SaveFileDialog();
            saveFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
            saveFileDialog1.FilterIndex = 2;
            saveFileDialog1.RestoreDirectory = true;
            if (saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            {
                charstore = saveFileDialog1.FileName;
                saveConfig(charstore);
                savePathInitFile(charstore);
            }
        }
        public void saveConfig(string charstore)
        {
            reconfiguration = true;
            System.Threading.Thread.Sleep(1000);
            System.IO.StreamWriter file = new System.IO.StreamWriter(charstore);
            if (chkBF8S.Checked)
            {
                cmBWNHOME.Text = cmBWNHOME.Text.Replace("(R)", "");
                cmBWAHOME.Text = cmBWAHOME.Text.Replace("(R)", "");
            }
            else
```

```
            {
                cmBWNHOME.Text = cmBWNHOME.Text.Replace("(R)", "");
                cmBWAHOME.Text = cmBWAHOME.Text.Replace("(R)", "");
                cmBWNHOME.Text = cmBWNHOME.Text + "(R)";
                cmBWAHOME.Text = cmBWAHOME.Text + "(R)";
            }
            if (!chkBF9S.Checked)
                cmBWAWHEELa.Text = " ";
            if (chkBF10S.Checked)
            {
                cmBWNROLLLEFT.Text = " ";
                cmBWNROLLRIGHT.Text = " ";
            }
            if (!chkBF2C.Checked)
                cmBWNCANCELRELOAD.Text = " ";
            if (!chkBF3C.Checked)
                cmBWNDOUBLEA.Text = " ";
            if (cmBWARIGHT.Text == cmBWARIGHT.Items[0].ToString() & cmBWAUP.Text ==
cmBWAUP.Items[0].ToString() & cmBWADOWN.Text == cmBWADOWN.Items[0].ToString() &
cmBWNPLUS.Text == cmBWNPLUS.Items[0].ToString() & cmBWALEFT.Text ==
cmBWALEFT.Items[0].ToString() & cmBWNA.Text == cmBWNA.Items[0].ToString() &
((cmBWAHOME.Text == cmBWAHOME.Items[0].ToString() & !chkBF8S.Checked) | (cmBWAHOME.Text ==
cmBWAHOME.Items[0].ToString() & chkBF8S.Checked)) & cmBWATWO.Text ==
cmBWATWO.Items[0].ToString() & cmBWNTOFRONT.Text == cmBWNTOFRONT.Items[0].ToString() &
((cmBWAWHEELa.Text == cmBWAWHEELa.Items[0].ToString() & chkBF9S.Checked) |
(cmBWAWHEELa.Text == " " & !chkBF9S.Checked)) & cmBWAROLLLEFT.Text ==
cmBWAROLLLEFT.Items[0].ToString() & cmBWNB.Text == cmBWNB.Items[0].ToString() &
cmBWNMINUS.Text == cmBWNMINUS.Items[0].ToString() & cmBWAONE.Text ==
cmBWAONE.Items[0].ToString() & cmBWAAB.Text == cmBWAAB.Items[0].ToString() &
cmBWNTOFRONT.Text == cmBWNTOFRONT.Items[0].ToString() & cmBWNUP.Text ==
cmBWNUP.Items[0].ToString() & cmBWNLEFT.Text == cmBWNLEFT.Items[0].ToString() &
cmBWNRIGHT.Text == cmBWNRIGHT.Items[0].ToString() & cmBWNDOWN.Text ==
cmBWNDOWN.Items[0].ToString() & cmBWNC.Text == cmBWNC.Items[0].ToString() & cmBWNSTICK.Text
== cmBWNSTICK.Items[0].ToString() & cmBWNB.Text == cmBWNB.Items[0].ToString() & cmBWNA.Text
== cmBWNA.Items[0].ToString() & cmBWNCZ.Text == cmBWNCZ.Items[0].ToString() & cmBWNZ.Text
== cmBWNZ.Items[0].ToString() & cmBWNMINUS.Text == cmBWNMINUS.Items[0].ToString() &
((cmBWNHOME.Text == cmBWNHOME.Items[0].ToString() & !chkBF8S.Checked) | (cmBWNHOME.Text ==
cmBWNHOME.Items[0].ToString() & chkBF8S.Checked)) & cmBWNPLUS.Text ==
cmBWNPLUS.Items[0].ToString() & cmBWNTODOWN1.Text == cmBWNTODOWN1.Items[0].ToString() &
cmBWNTODOWN2.Text == cmBWNTODOWN2.Items[0].ToString() & cmBWNTODOWN3.Text ==
cmBWNTODOWN3.Items[0].ToString() & cmBWNONE.Text == cmBWNONE.Items[0].ToString() &
((cmBWNROLLLEFT.Text == cmBWNROLLLEFT.Items[0].ToString() & !chkBF10S.Checked) |
(cmBWNROLLLEFT.Text == " " & chkBF10S.Checked)) & ((cmBWNROLLRIGHT.Text ==
cmBWNROLLRIGHT.Items[0].ToString() & !chkBF10S.Checked) | (cmBWNROLLRIGHT.Text == " " &
chkBF10S.Checked)) & cmBWNTWO.Text == cmBWNTWO.Items[0].ToString() &
((cmBWNCANCELRELOAD.Text == cmBWNCANCELRELOAD.Items[0].ToString() & cmBWNCANCELRELOAD.Text
== cmBWNCANCELRELOAD.Items[0].ToString() & chkBF2C.Checked) | (cmBWNCANCELRELOAD.Text == "
" & !chkBF2C.Checked)) & ((cmBWNDOUBLEA.Text == cmBWNDOUBLEA.Items[0].ToString() &
chkBF3C.Checked) | (cmBWNDOUBLEA.Text == " " & !chkBF3C.Checked)) & cmBWN1SWITCH.Text ==
cmBWN1SWITCH.Items[0].ToString() & cmBWN1SWITCH.Text == cmBWN1SWITCH.Items[0].ToString())
                Fbool["//rebind keys"] = false;
            else
                Fbool["//rebind keys"] = true;
            action["//wiimote alone wheel"] = cmBWAWHEELa.Text;
            action["//1 tab switch"] = cmBWN1SWITCH.Text;
            action["//double A r"] = cmBWNDOUBLEA.Text;
            action["//cancel reload x"] = cmBWNCANCELRELOAD.Text;
            action["//wiimote b"] = cmBWNB.Text;
            action["//wiimote a"] = cmBWNA.Text;
            action["//wiimote plus"] = cmBWNPLUS.Text;
            action["//wiimote minus"] = cmBWNMINUS.Text;
            action["//wiimote to front"] = cmBWNTOFRONT.Text;
            action["//nunchuck stick"] = cmBWNSTICK.Text;
            action["//wiimote alone up"] = cmBWAUP.Text;
            action["//wiimote alone down"] = cmBWADOWN.Text;
            action["//wiimote alone roll left"] = cmBWAROLLLEFT.Text;
```

```
            action["//wiimote alone home"] = cmBWAHOME.Text.Replace("(R)", "");
            action["//wiimote alone a and b"] = cmBWAAB.Text;
            action["//wiimote alone left"] = cmBWALEFT.Text;
            action["//wiimote alone right"] = cmBWARIGHT.Text;
            action["//wiimote alone 1"] = cmBWAONE.Text;
            action["//wiimote alone 2"] = cmBWATWO.Text;
            action["//wiimote nunchuck roll right"] = cmBWNROLLRIGHT.Text;
            action["//wiimote nunchuck roll left"] = cmBWNROLLLEFT.Text;
            action["//nunchuck to down"] = cmBWNTODOWN1.Text;
            action["//nunchuck to down 2"] = cmBWNTODOWN2.Text;
            action["//nunchuck to down 3"] = cmBWNTODOWN3.Text;
            action["//nunchuck z and c"] = cmBWNCZ.Text;
            action["//nunchuck c"] = cmBWNC.Text;
            action["//nunchuck z"] = cmBWNZ.Text;
            action["//wiimote nunchuck down"] = cmBWNDOWN.Text;
            action["//wiimote nunchuck home"] = cmBWNHOME.Text.Replace("(R)", "");
            action["//wiimote nunchuck left"] = cmBWNLEFT.Text;
            action["//wiimote nunchuck right"] = cmBWNRIGHT.Text;
            action["//wiimote nunchuck up"] = cmBWNUP.Text;
            action["//wiimote nunchuck 1"] = cmBWNONE.Text;
            action["//wiimote nunchuck 2"] = cmBWNTWO.Text;
            Fvar["//angleinit"] = Convert.ToInt32(Regex.Replace(txtBangleinit.Text, "[^0-9-
]", ""));
            Fvar["//irxinit"] = Convert.ToInt32(Regex.Replace(txtBirxinit.Text, "[^0-9-]",
""));
            Fvar["//iryinit"] = Convert.ToInt32(Regex.Replace(txtBiryinit.Text, "[^0-9-]",
""));
            Fvar["//wiimote to front push r time extra setting"] =
Convert.ToInt32(Regex.Replace(txtBwiimotetofrontpush.Text, "[^0-9-]", ""));
            Fvar["//double A push r time extra setting"] =
Convert.ToInt32(Regex.Replace(txtBdoubleapush.Text, "[^0-9-]", ""));
            Fvar["//cancel reload waiting A or C time extra setting"] =
Convert.ToInt32(Regex.Replace(txtBcancelreload.Text, "[^0-9-]", ""));
            Fvar["//second nunchuck to down push v time extra setting"] =
Convert.ToInt32(Regex.Replace(txtBsecondnunchucktodown.Text, "[^0-9-]", ""));
            Fvar["//brink or titanfall time extra setting"] =
Convert.ToInt32(Regex.Replace(txtBbrinkortitanfall.Text, "[^0-9-]", ""));
            Fvar["//bo3 time extra setting"] = Convert.ToInt32(Regex.Replace(txtBbo3.Text,
"[^0-9-]", ""));
            Fvar["//smooth time extra setting"] =
Convert.ToInt32(Regex.Replace(txtBsmooth.Text, "[^0-9-]", ""));
            Fvar["//aim plus latency time extra setting"] =
Convert.ToInt32(Regex.Replace(txtBaimpluslatency.Text, "[^0-9-]", ""));
            Fvar["//aim plus quantity extra setting"] =
Convert.ToInt32(Regex.Replace(txtBaimplusquantity.Text, "[^0-9-]", ""));
            Fvar["//anti-tearing outer size"] =
Convert.ToInt32(Regex.Replace(txtBantitearingoutersize.Text, "[^0-9-]", ""));
            Fvar["//hardness quantity"] =
Convert.ToInt32(Regex.Replace(txtBhardnessquantity.Text, "[^0-9-]", ""));
            Fvar["//aim speed axis x quantity"] =
Convert.ToInt32(Regex.Replace(txtBaimspeedaxisxquantity.Text, "[^0-9-]", ""));
            Fvar["//aim speed axis y quantity"] =
Convert.ToInt32(Regex.Replace(txtBaimspeedaxisyquantity.Text, "[^0-9-]", ""));
            Fvar["//aim speed accuracy size of center axis x extra setting"] =
Convert.ToInt32(Regex.Replace(txtBaimspeedaccuracysizex.Text, "[^0-9-]", ""));
            Fvar["//aim speed accuracy multipler of center axis x extra setting"] =
Convert.ToInt32(Regex.Replace(txtBaimspeedaccuracymultiplerx.Text, "[^0-9-]", ""));
            Fvar["//aim speed accuracy size of center axis y extra setting"] =
Convert.ToInt32(Regex.Replace(txtBaimspeedaccuracysizey.Text, "[^0-9-]", ""));
            Fvar["//aim speed accuracy multipler of center axis y extra setting"] =
Convert.ToInt32(Regex.Replace(txtBaimspeedaccuracymultiplery.Text, "[^0-9-]", ""));
            Fvar["//no recoil quantity extra setting"] =
Convert.ToInt32(Regex.Replace(txtBnorecoilquantity.Text, "[^0-9-]", ""));
            Fvar["//1 tab switch interval time extra setting"] =
Convert.ToInt32(Regex.Replace(txtB1tabswitchinterval.Text, "[^0-9-]", ""));
```

```
            Fvar["//1 tab switch press delay time extra setting"] =
Convert.ToInt32(Regex.Replace(txtB1tabswitchpressdelay.Text, "[^0-9-]", ""));
            Fvar["//tick time"] = Convert.ToInt32(Regex.Replace(txtBticktime.Text, "[^0-9-
]", ""));
            Fvar["//wheel script stick limit in"] =
Convert.ToInt32(Regex.Replace(txtBwheelscriptsticklimitin.Text, "[^0-9-]", ""));
            Fvar["//wheel script stick limit out"] =
Convert.ToInt32(Regex.Replace(txtBwheelscriptsticklimitout.Text, "[^0-9-]", ""));
            Fvar["//wheel script gyroscope limit in"] =
Convert.ToInt32(Regex.Replace(txtBwheelscriptgyroscopelimitin.Text, "[^0-9-]", ""));
            Fvar["//wheel script gyroscope limit out"] =
Convert.ToInt32(Regex.Replace(txtBwheelscriptgyroscopelimitout.Text, "[^0-9-]", ""));
            Fvar["//zoning quantity"] =
Convert.ToInt32(Regex.Replace(txtBzoningquantity.Text, "[^0-9-]", ""));
            Fvar["//zoning hardness quantity"] =
Convert.ToInt32(Regex.Replace(txtBzoninghardnessquantity.Text, "[^0-9-]", ""));
            Fvar["//no recoil step quantity"] =
Convert.ToInt32(Regex.Replace(txtBnorecoilstepquantity.Text, "[^0-9-]", ""));
            Fvar["//slide init"] = Convert.ToInt32(Regex.Replace(txtBslideinit.Text, "[^0-
9-]", ""));
            Fvar["//slide time"] = Convert.ToInt32(Regex.Replace(txtBslidetime.Text, "[^0-
9-]", ""));
            file.WriteLine(charstore);
            file.WriteLine("//brink");
            file.WriteLine(chkBF1.Checked);
            file.WriteLine("//metro");
            file.WriteLine(chkBF2.Checked);
            file.WriteLine("//titanfall");
            file.WriteLine(chkBF3.Checked);
            file.WriteLine("//cursor");
            file.WriteLine(chkBF4.Checked);
            file.WriteLine("//warface");
            file.WriteLine(chkBF5.Checked);
            file.WriteLine("//bo3");
            file.WriteLine(chkBF6.Checked);
            file.WriteLine("//fake");
            file.WriteLine(chkBF7.Checked);
            file.WriteLine("//mw3");
            file.WriteLine(chkBF8.Checked);
            file.WriteLine("//xaim");
            file.WriteLine(chkBF9.Checked);
            file.WriteLine("//A press I/O");
            file.WriteLine(chkBF12.Checked);
            file.WriteLine("//A accuracy");
            file.WriteLine(chkBF11.Checked);
            file.WriteLine("//no roll qe");
            file.WriteLine(chkBF10S.Checked);
            file.WriteLine("//Home f only");
            file.WriteLine(chkBF8S.Checked);
            file.WriteLine("//wheel script");
            file.WriteLine(chkBF9S.Checked);
            file.WriteLine("//A view on");
            file.WriteLine(chkBF1S.Checked);
            file.WriteLine("//1 tab I/O");
            file.WriteLine(chkBF7S.Checked);
            file.WriteLine("//A aim plus");
            file.WriteLine(chkBF2S.Checked);
            file.WriteLine("//Home -> - + +");
            file.WriteLine(chkBF11S.Checked);
            file.WriteLine("//stick view");
            file.WriteLine(chkBF3S.Checked);
            file.WriteLine("//wheel view");
            file.WriteLine(chkBF4S.Checked);
            file.WriteLine("//1 and 2 view");
            file.WriteLine(chkBF6S.Checked);
            file.WriteLine("//nunchuck");
```

```
file.WriteLine(chkBF12S.Checked);
file.WriteLine("//rebind keys");
file.WriteLine(Fbool["//rebind keys"]);
file.WriteLine("//lock features and options");
file.WriteLine(Fbool["//lock features and options"]);
file.WriteLine("//1 B swap");
file.WriteLine(chkBF5C.Checked);
file.WriteLine("//A C swap");
file.WriteLine(chkBF6C.Checked);
file.WriteLine("//push r 1'");
file.WriteLine(chkBF1C.Checked);
file.WriteLine("//cancel reload x");
file.WriteLine(chkBF2C.Checked);
file.WriteLine("//double A r");
file.WriteLine(chkBF3C.Checked);
file.WriteLine("//1 tab switch");
file.WriteLine(chkBF4C.Checked);
file.WriteLine("//A slide A+B");
file.WriteLine(chkBF7C.Checked);
file.WriteLine("//dpad view");
file.WriteLine(chkBF5S.Checked);
file.WriteLine("//A B switch");
file.WriteLine(chkBF8C.Checked);
file.WriteLine("//Z press I/O");
file.WriteLine(chkBF9C.Checked);
file.WriteLine("//driver mouse");
file.WriteLine(chkBF10.Checked);
file.WriteLine("//driver keyboard");
file.WriteLine(chkBF12C.Checked);
file.WriteLine("//stick arrows");
file.WriteLine(chkBF10C.Checked);
file.WriteLine("//A roll qe");
file.WriteLine(chkBF11C.Checked);
file.WriteLine("//cancel reload x");
file.WriteLine(action["//cancel reload x"]);
file.WriteLine("//double A r");
file.WriteLine(action["//double A r"]);
file.WriteLine("//1 tab switch");
file.WriteLine(action["//1 tab switch"]);
file.WriteLine("//nunchuck stick");
file.WriteLine(action["//nunchuck stick"]);
file.WriteLine("//wiimote alone wheel");
file.WriteLine(action["//wiimote alone wheel"]);
file.WriteLine("//wiimote alone up");
file.WriteLine(action["//wiimote alone up"]);
file.WriteLine("//wiimote alone down");
file.WriteLine(action["//wiimote alone down"]);
file.WriteLine("//wiimote b");
file.WriteLine(action["//wiimote b"]);
file.WriteLine("//wiimote a");
file.WriteLine(action["//wiimote a"]);
file.WriteLine("//wiimote plus");
file.WriteLine(action["//wiimote plus"]);
file.WriteLine("//wiimote minus");
file.WriteLine(action["//wiimote minus"]);
file.WriteLine("//wiimote alone roll left");
file.WriteLine(action["//wiimote alone roll left"]);
file.WriteLine("//wiimote alone home");
file.WriteLine(action["//wiimote alone home"].Replace("(R)", ""));
file.WriteLine("//wiimote alone a and b");
file.WriteLine(action["//wiimote alone a and b"]);
file.WriteLine("//wiimote alone left");
file.WriteLine(action["//wiimote alone left"]);
file.WriteLine("//wiimote alone right");
file.WriteLine(action["//wiimote alone right"]);
file.WriteLine("//wiimote alone 1");
```

```
            file.WriteLine(action["//wiimote alone 1"]);
            file.WriteLine("//wiimote alone 2");
            file.WriteLine(action["//wiimote alone 2"]);
            file.WriteLine("//wiimote nunchuck roll right");
            file.WriteLine(action["//wiimote nunchuck roll right"]);
            file.WriteLine("//wiimote nunchuck roll left");
            file.WriteLine(action["//wiimote nunchuck roll left"]);
            file.WriteLine("//wiimote to front");
            file.WriteLine(action["//wiimote to front"]);
            file.WriteLine("//nunchuck to down");
            file.WriteLine(action["//nunchuck to down"]);
            file.WriteLine("//nunchuck to down");
            file.WriteLine(action["//nunchuck to down 2"]);
            file.WriteLine("//nunchuck to down");
            file.WriteLine(action["//nunchuck to down 3"]);
            file.WriteLine("//nunchuck z and c");
            file.WriteLine(action["//nunchuck z and c"]);
            file.WriteLine("//nunchuck c");
            file.WriteLine(action["//nunchuck c"]);
            file.WriteLine("//nunchuck z");
            file.WriteLine(action["//nunchuck z"]);
            file.WriteLine("//wiimote nunchuck down");
            file.WriteLine(action["//wiimote nunchuck down"]);
            file.WriteLine("//wiimote nunchuck home");
            file.WriteLine(action["//wiimote nunchuck home"].Replace("(R)", ""));
            file.WriteLine("//wiimote nunchuck left");
            file.WriteLine(action["//wiimote nunchuck left"]);
            file.WriteLine("//wiimote nunchuck right");
            file.WriteLine(action["//wiimote nunchuck right"]);
            file.WriteLine("//wiimote nunchuck up");
            file.WriteLine(action["//wiimote nunchuck up"]);
            file.WriteLine("//wiimote nunchuck 1");
            file.WriteLine(action["//wiimote nunchuck 1"]);
            file.WriteLine("//wiimote nunchuck 2");
            file.WriteLine(action["//wiimote nunchuck 2"]);
            file.WriteLine("//angleinit");
            file.WriteLine((Convert.ToInt32(Fvar["//angleinit"])).ToString());
            file.WriteLine("//irxinit");
            file.WriteLine((Convert.ToInt32(Fvar["//irxinit"])).ToString());
            file.WriteLine("//iryinit");
            file.WriteLine((Convert.ToInt32(Fvar["//iryinit"])).ToString());
            file.WriteLine("//wiimote to front push r time extra setting");
            file.WriteLine((Convert.ToInt32(Fvar["//wiimote to front push r time extra
setting"])).ToString());
            file.WriteLine("//double A push r time extra setting");
            file.WriteLine((Convert.ToInt32(Fvar["//double A push r time extra
setting"])).ToString());
            file.WriteLine("//cancel reload waiting A or C time extra setting");
            file.WriteLine((Convert.ToInt32(Fvar["//cancel reload waiting A or C time extra
setting"])).ToString());
            file.WriteLine("//second nunchuck to down push v time extra setting");
            file.WriteLine((Convert.ToInt32(Fvar["//second nunchuck to down push v time
extra setting"])).ToString());
            file.WriteLine("//brink or titanfall time extra setting");
            file.WriteLine((Convert.ToInt32(Fvar["//brink or titanfall time extra
setting"])).ToString());
            file.WriteLine("//bo3 time extra setting");
            file.WriteLine((Convert.ToInt32(Fvar["//bo3 time extra setting"])).ToString());
            file.WriteLine("//smooth time extra setting");
            file.WriteLine((Convert.ToInt32(Fvar["//smooth time extra
setting"])).ToString());
            file.WriteLine("//aim plus latency time extra setting");
            file.WriteLine((Convert.ToInt32(Fvar["//aim plus latency time extra
setting"])).ToString());
            file.WriteLine("//aim plus quantity extra setting");
```

```
            file.WriteLine((Convert.ToInt32(Fvar["//aim plus quantity extra
setting"])).ToString());
            file.WriteLine("//anti-tearing outer size");
            file.WriteLine((Convert.ToInt32(Fvar["//anti-tearing outer
size"])).ToString());
            file.WriteLine("//hardness quantity");
            file.WriteLine((Convert.ToInt32(Fvar["//hardness quantity"])).ToString());
            file.WriteLine("//aim speed axis x quantity");
            file.WriteLine((Convert.ToInt32(Fvar["//aim speed axis x
quantity"])).ToString());
            file.WriteLine("//aim speed axis y quantity");
            file.WriteLine((Convert.ToInt32(Fvar["//aim speed axis y
quantity"])).ToString());
            file.WriteLine("//aim speed accuracy size of center axis x extra setting");
            file.WriteLine((Convert.ToInt32(Fvar["//aim speed accuracy size of center axis
x extra setting"])).ToString());
            file.WriteLine("//aim speed accuracy multipler of center axis x extra
setting");
            file.WriteLine((Convert.ToInt32(Fvar["//aim speed accuracy multipler of center
axis x extra setting"])).ToString());
            file.WriteLine("//aim speed accuracy size of center axis y extra setting");
            file.WriteLine((Convert.ToInt32(Fvar["//aim speed accuracy size of center axis
y extra setting"])).ToString());
            file.WriteLine("//aim speed accuracy multipler of center axis y extra
setting");
            file.WriteLine((Convert.ToInt32(Fvar["//aim speed accuracy multipler of center
axis y extra setting"])).ToString());
            file.WriteLine("//no recoil quantity extra setting");
            file.WriteLine((Convert.ToInt32(Fvar["//no recoil quantity extra
setting"])).ToString());
            file.WriteLine("//1 tab switch interval time extra setting");
            file.WriteLine((Convert.ToInt32(Fvar["//1 tab switch interval time extra
setting"])).ToString());
            file.WriteLine("//1 tab switch press delay time extra setting");
            file.WriteLine((Convert.ToInt32(Fvar["//1 tab switch press delay time extra
setting"])).ToString());
            file.WriteLine("//tick time");
            file.WriteLine((Convert.ToInt32(Fvar["//tick time"])).ToString());
            file.WriteLine("//wheel script stick limit in");
            file.WriteLine((Convert.ToInt32(Fvar["//wheel script stick limit
in"])).ToString());
            file.WriteLine("//wheel script stick limit out");
            file.WriteLine((Convert.ToInt32(Fvar["//wheel script stick limit
out"])).ToString());
            file.WriteLine("//wheel script gyroscope limit in");
            file.WriteLine((Convert.ToInt32(Fvar["//wheel script gyroscope limit
in"])).ToString());
            file.WriteLine("//wheel script gyroscope limit out");
            file.WriteLine((Convert.ToInt32(Fvar["//wheel script gyroscope limit
out"])).ToString());
            file.WriteLine("//zoning quantity");
            file.WriteLine((Convert.ToInt32(Fvar["//zoning quantity"])).ToString());
            file.WriteLine("//zoning hardness quantity");
            file.WriteLine((Convert.ToInt32(Fvar["//zoning hardness
quantity"])).ToString());
            file.WriteLine("//no recoil step quantity");
            file.WriteLine((Convert.ToInt32(Fvar["//no recoil step
quantity"])).ToString());
            file.WriteLine("//slide init");
            file.WriteLine((Convert.ToInt32(Fvar["//slide init"])).ToString());
            file.WriteLine("//slide time");
            file.WriteLine((Convert.ToInt32(Fvar["//slide time"])).ToString());
            file.WriteLine("//List of possible entries:");
            file.WriteLine("WASD, ZQSD, arrow keys, AD, QD, left right arrow keys, A, B, C,
D, E, ..., X, Y, Z, 0, 1, 2, ..., 8, 9, F1, F2, F3, ..., F11,");
```

```
            file.WriteLine("F12, Capslock, Alt, Back, Apostrophe, left, right, up, down,
Escape, Control, LControl, RControl, Shift, LShift, RShift,");
            file.WriteLine("Enter, Space, Tab, wheel down, wheel up, middle click, left
click, right click, 0-4, 5-9, Enter/Tab.");
            file.Close();
            if ((Fbool["//brink"] & !chkBF1.Checked) | (!Fbool["//brink"] &
chkBF1.Checked))
                Abool["//brink"] = true;
            if ((Fbool["//metro"] & !chkBF2.Checked) | (!Fbool["//metro"] &
chkBF2.Checked))
                Abool["//metro"] = true;
            if ((Fbool["//titanfall"] & !chkBF3.Checked) | (!Fbool["//titanfall"] &
chkBF3.Checked))
                Abool["//titanfall"] = true;
            if ((Fbool["//cursor"] & !chkBF4.Checked) | (!Fbool["//cursor"] &
chkBF4.Checked))
                Abool["//cursor"] = true;
            if ((Fbool["//A press I/O"] & !chkBF12.Checked) | (!Fbool["//A press I/O"] &
chkBF12.Checked))
                Abool["//A press I/O"] = true;
            if ((Fbool["//warface"] & !chkBF5.Checked) | (!Fbool["//warface"] &
chkBF5.Checked))
                Abool["//warface"] = true;
            if ((Fbool["//bo3"] & !chkBF6.Checked) | (!Fbool["//bo3"] & chkBF6.Checked))
                Abool["//bo3"] = true;
            if ((Fbool["//fake"] & !chkBF7.Checked) | (!Fbool["//fake"] & chkBF7.Checked))
                Abool["//fake"] = true;
            if ((Fbool["//mw3"] & !chkBF8.Checked) | (!Fbool["//mw3"] & chkBF8.Checked))
                Abool["//mw3"] = true;
            if ((Fbool["//wheel script"] & !chkBF9S.Checked) | (!Fbool["//wheel script"] &
chkBF9S.Checked))
                Abool["//wheel script"] = true;
            if ((Fbool["//A accuracy"] & !chkBF11.Checked) | (!Fbool["//A accuracy"] &
chkBF11.Checked))
                Abool["//A accuracy"] = true;
            if ((Fbool["//no roll qe"] & !chkBF10S.Checked) | (!Fbool["//no roll qe"] &
chkBF10S.Checked))
                Abool["//no roll qe"] = true;
            if ((Fbool["//Home f only"] & !chkBF8S.Checked) | (!Fbool["//Home f only"] &
chkBF8S.Checked))
                Abool["//Home f only"] = true;
            if ((Fbool["//A view on"] & !chkBF1S.Checked) | (!Fbool["//A view on"] &
chkBF1S.Checked))
                Abool["//A view on"] = true;
            if ((Fbool["//1 tab I/O"] & !chkBF7S.Checked) | (!Fbool["//1 tab I/O"] &
chkBF7S.Checked))
                Abool["//1 tab I/O"] = true;
            if ((Fbool["//A aim plus"] & !chkBF2S.Checked) | (!Fbool["//A aim plus"] &
chkBF2S.Checked))
                Abool["//A aim plus"] = true;
            if ((Fbool["//xaim"] & !chkBF9.Checked) | (!Fbool["//xaim"] & chkBF9.Checked))
                Abool["//xaim"] = true;
            if ((Fbool["//Home -> - + +"] & !chkBF11S.Checked) | (!Fbool["//Home -> - + +"]
& chkBF11S.Checked))
                Abool["//Home -> - + +"] = true;
            if ((Fbool["//1 and 2 view"] & !chkBF6S.Checked) | (!Fbool["//1 and 2 view"] &
chkBF6S.Checked))
                Abool["//1 and 2 view"] = true;
            if ((Fbool["//nunchuck"] & !chkBF12S.Checked) | (!Fbool["//nunchuck"] &
chkBF12S.Checked))
                Abool["//nunchuck"] = true;
            if ((Fbool["//wheel view"] & !chkBF4S.Checked) | (!Fbool["//wheel view"] &
chkBF4S.Checked))
                Abool["//wheel view"] = true;
            if ((Fbool["//stick view"] & !chkBF3S.Checked) | (!Fbool["//stick view"] &
chkBF3S.Checked))
```

```
                Abool["//stick view"] = true;
            if ((Fbool["//cancel reload x"] & !chkBF2C.Checked) | (!Fbool["//cancel reload
x"] & chkBF2C.Checked))
                Abool["//cancel reload x"] = true;
            if ((Fbool["//double A r"] & !chkBF3C.Checked) | (!Fbool["//double A r"] &
chkBF3C.Checked))
                Abool["//double A r"] = true;
            if ((Fbool["//1 tab switch"] & !chkBF4C.Checked) | (!Fbool["//1 tab switch"] &
chkBF4C.Checked))
                Abool["//1 tab switch"] = true;
            if ((Fbool["//1 B swap"] & !chkBF5C.Checked) | (!Fbool["//1 B swap"] &
chkBF5C.Checked))
                Abool["//1 B swap"] = true;
            if ((Fbool["//A C swap"] & !chkBF6C.Checked) | (!Fbool["//A C swap"] &
chkBF6C.Checked))
                Abool["//A C swap"] = true;
            if ((Fbool["//push r 1'"] & !chkBF1C.Checked) | (!Fbool["//push r 1'"] &
chkBF1C.Checked))
                Abool["//push r 1'"] = true;
            if ((Fbool["//A slide A+B"] & !chkBF7C.Checked) | (!Fbool["//A slide A+B"] &
chkBF7C.Checked))
                Abool["//A slide A+B"] = true;
            if ((Fbool["//dpad view"] & !chkBF5S.Checked) | (!Fbool["//dpad view"] &
chkBF5S.Checked))
                Abool["//dpad view"] = true;
            if ((Fbool["//A B switch"] & !chkBF8C.Checked) | (!Fbool["//A B switch"] &
chkBF8C.Checked))
                Abool["//A B switch"] = true;
            if ((Fbool["//Z press I/O"] & !chkBF9C.Checked) | (!Fbool["//Z press I/O"] &
chkBF9C.Checked))
                Abool["//Z press I/O"] = true;
            if ((Fbool["//driver mouse"] & !chkBF10.Checked) | (!Fbool["//driver mouse"] &
chkBF10.Checked))
                Abool["//driver mouse"] = true;
            if ((Fbool["//driver keyboard"] & !chkBF12C.Checked) | (!Fbool["//driver
keyboard"] & chkBF12C.Checked))
                Abool["//driver keyboard"] = true;
            if ((Fbool["//stick arrows"] & !chkBF10C.Checked) | (!Fbool["//stick arrows"] &
chkBF10C.Checked))
                Abool["//stick arrows"] = true;
            if ((Fbool["//A roll qe"] & !chkBF11C.Checked) | (!Fbool["//A roll qe"] &
chkBF11C.Checked))
                Abool["//A roll qe"] = true;
            Assignating();
            clearList();
            setControlsAndOptions();
            reconfiguration = false;
        }
        public void savePathInitFile(string pathtoinitfile)
        {
            System.IO.StreamWriter initfile = new System.IO.StreamWriter("initfile.txt");
            initfile.WriteLine("//path to last open or save file");
            initfile.WriteLine(pathtoinitfile);
            initfile.WriteLine("//enable autoload of last open or save file");
            initfile.WriteLine(enableautoloadoflastfile);
            initfile.Close();
        }
        public void button3_Click(object sender, EventArgs e)
        {
            const string message = "• Start the program with Wiimote couch or hold
horizontally for calibration, and press 1 and 2 buttons for pairing it, and also with
administrative privilege.\n\r• Let 1 meter between sensor barre and Wiimote.\n\r• Free
sensor barre all around it.\n\r• Adapt the mouse sensitivities and DPI in game
options.\n\r• Press 1+2, Wiimote seeing sensor bar, to center the cursor position if screen
size changed.\n\r• Press 1+2, Wiimote seeing sensor bar and during 2 seconds, to take
Wiimote default angle position under the roll axis and Wiimote IR position as new centered
```

```
position.\n\r• Press Z+C+Up to switch IR mouse view control to stick mouse view
control.\n\r• Press Z+C+Down to switch IR mouse view control to wheel mouse view
control.\n\r• Press Z+1 to switch Wiimote with nunchuck to Wiimote alone with wheel script
enable.\n\r• Press C+1 to switch Wiimote with nunchuck to Wiimote alone with wheel script
disable.\n\r• Press F1, F2, ..., F12, Shift+F1, Shift+F2, ..., or Shift+F12, Control+F1,
Control+F2, ..., or Control+F4 for enable features and options.\n\r• Press
Shift+Control+Capslock to lock change of features and options.\n\r• Check A press I/O and A
view on for enable A view I/O.\n\r• Check both wheel view and dpad view for control view
with gyroscope and right and left buttons.\n\r• Check wheel view and 1 and 2 view for
control view with gyroscope and 1 and 2 buttons.\n\r• Check stick view and 1 and 2 view for
control view with stick and A and B buttons.\n\r• Check dpad view and press B button for
control half view.\n\r• Check 1 tab I/O and 1 tab switch for enable 1 tab switch I/O.\n\r•
Check 1 B swap for using above options with B.\n\r• Check push r 1' and double A r for
enable double A giving r pushed 1'.\n\r• Check A C swap for using C instead of A in A
accuracy, A press I/O, A view on, A aim plus, cancel reload x, double A r.\n\r• Set
negative numbers for extra settings of no recoil quantity to have recoil when firing, of
aim plus quantity to lower speed aim while aiming, of aim speed accuracy multipler of
center extra setting with positive number of aim speed accuracy size of center extra
setting to have a deadzone, or of anti-tearing outer size to have tearing outer.\n\r• Check
A aim plus if you check A accuracy for remove deadzone progressively.\n\r• Change hardness
quantity or zoning hardness quantity only for metro or xaim or fake.\n\r• Check A B switch,
1 tab switch, and 1 B swap for enable switch of B when holding A.\n\r• Lean 1' the Wiimote
to the left for enable stick sending arrows instead of WASD.\n\r• Press A while lean
nunchuck for roll QE.\n\r• Press decimal to unlock controls.\n\r• Change controls and
options with click on checkboxes.\n\r• Save in a file for enable changes.";
            const string caption = "WiimoteTheory Legend";
            MessageBox.Show(message, caption, MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }
        public void button4_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        public void button5_Click(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Minimized;
        }
        private void FormClose()
        {
            disconnect();
        }
        private void Form1_FormClosed(object sender, FormClosedEventArgs e)
        {
            notpressing1and2 = true;
            runningoff = true;
            switchwheelfix();
            TimeEndPeriod(1);
            System.Threading.Thread.Sleep(100);
            threadstart = new ThreadStart(FormClose);
            thread = new Thread(threadstart);
            thread.Start();
        }
        private void button6_Click(object sender, EventArgs e)
        {
            if (!Fbool["//lock features and options"])
                Fbool["//lock features and options"] = true;
            else
                if (Fbool["//lock features and options"])
                    Fbool["//lock features and options"] = false;
        }
        private void button7_Click(object sender, EventArgs e)
        {
            if (tabControl.SelectedTab == tabControl.TabPages[0])
                tabControl.SelectTab(1);
            else
                tabControl.SelectTab(0);
```

```
        }
    }
}
```

7. Use and Agreement Contract

Owner: Michael Andre Franiatte.

Contact: michael.franiatte@gmail.com.

Owning: All works from scratch of the owner.

Proof of Owning: Works published, and writings/speakings all over.

Requirements of Use: Pay the owner, quote the owner, agreement of the owner.

Availability of Works: Only under the shapes of the owner built, only for personal use.

Subjects of Claims: Works published by the owner on Google Play and Google Books.

Concerning Author Rights: Equations and codes from scratch of the owner, softwares built from it, all things of people arising from it.

End User License Agreement: A commercial license is required to use in personal manner. Do not redistributing in any manner, including by computer media, a file server, an email attachment, etc. Do not embedding in or linking it to another programs, source codes and assistances including internal applications, scripts, batch files, etc. Do not use for any kind of technical support including on customer or retailer computer, hardware or software development, research, discovery, teachery, talk, speech, write, etc. Do not use for win money or for commercialisation of any products arising from my programs, source codes and assistances. Do not use and do not copy the way it run in other programs, source codes and assistances. Do not use without pay me, quote me and my agreement. Do not steal or copy or reproduce or modify or peer or share. Do not use in other manner than personal. It stand for my programs, source codes and assistances or programs, source codes and assistances stealing or copying or reproducing or modifying or peering or sharing my programs, source codes, and assistances. If you aren't agree you shall not use.

Terms of License and Price: The present contract acceptance is required to use works of the owner and built from it in all kind of manner. The price for each user shall be defined with the owner by contacting him and this for each subject of works the owner claims. Each user shall contact the owner for asking his agreement. It can be refused by the owner depending who asking and the price defined. People don’t respecting the present contract shall not use the works of the owner.