EBOOK

Michael Andre Franiatte

C# Codes for Joycons to Play PC Games

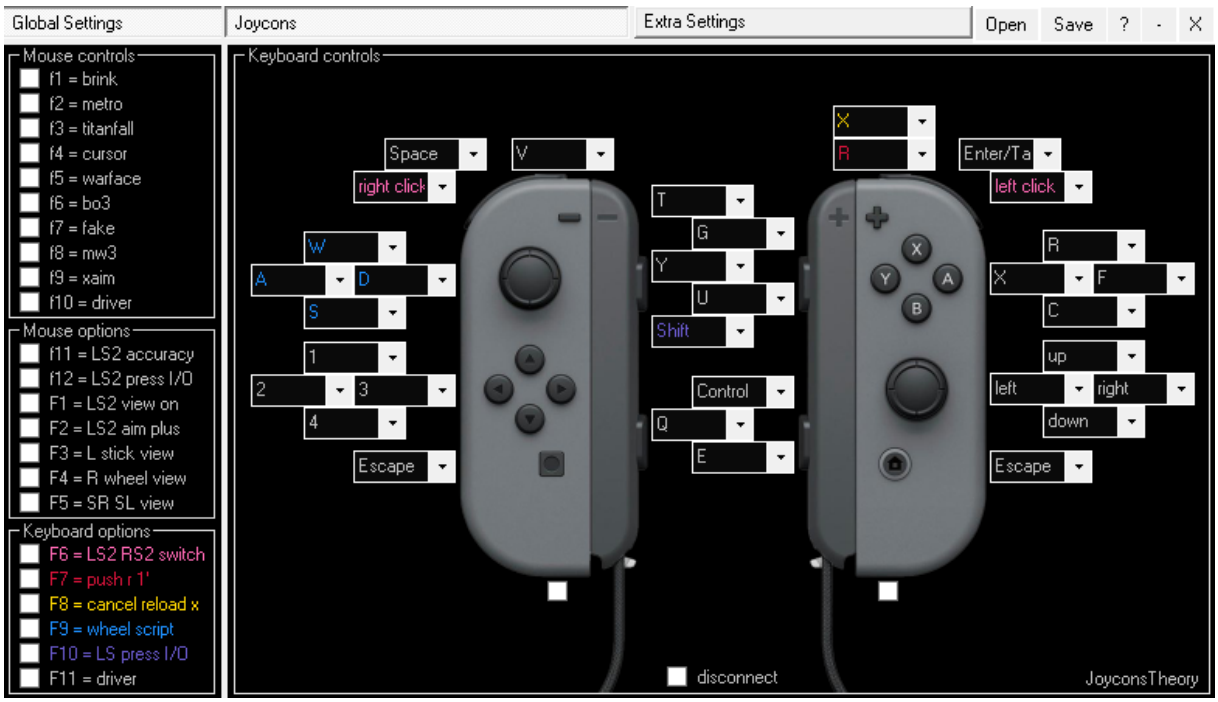
JoyconsTheory.exe

EBOOK

C# Codes for Joycons to Play PC Games

JoyconsTheory.exe

Michael Franiatte
03/02/2019



The C# codes presented can simulate keyboard and mouse events to play very well PC games using Joycons 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 Joycons to Play PC Games

Michael Franiatte*

Abstract

With these C# codes, Joycons 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 Joycons. This paper gives 10 years of works on coding Joycons 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. Joycons 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, Joycons*

* Author correspondence: michael.franiatte@gmail.com

1. Win32 C++ DLL JoyconsPairing 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 btdir;
BLUETOOTH_DEVICE_INFO btdil;
bool joyconlfound = false;
bool joyconrfound = false;
HBLUETOOTH_DEVICE_FIND hFind = NULL;
#pragma warning(disable : 4995)
extern "C"
{
	__declspec(dllexport) int 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;
		btdir.dwSize = sizeof(btdir);
		btdil.dwSize = sizeof(btdil);
		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"Joy-Con (R)") |
!wcscmp(btdi.szName, L"Joy-Con (L)"))
						{
```

```
                                                        if (!wcscmp(btdi.szName, L"Joy-Con (R)"))
                                                        {
                                                                btdir = btdi;
                                                                joyconrfound = true;
                                                        }
                                                        if (!wcscmp(btdi.szName, L"Joy-Con (L)"))
                                                        {
                                                                btdil = btdi;
                                                                joyconlfound = true;
                                                        }
                                                        WCHAR pass[6];
                                                        DWORD pcServices = 16;
                                                        GUID guids[16];
                                                        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);
                                                }
                                                nPaired++;
                                                Sleep(1);
                                        } while (BluetoothFindNextDevice(hFind, &btdi));
                                        Sleep(1);
                                }
                                if (!joyconrfound & joyconlfound)
                                        return 1;
                                if (joyconrfound & joyconlfound)
                                        return 2;
                                if (joyconrfound & !joyconlfound)
                                        return 3;
                        }
                }
                return 0;
        }
        __declspec(dllexport) bool disconnect()
        {
                BluetoothRemoveDevice(&btdil.Address);
                BluetoothRemoveDevice(&btdir.Address);
                return true;
        }
}
```

2. hidread Win32 C++ DLL hidapi.h header file

```
#include <wchar.h>
#include <windows.h>
#include <setupapi.h>
#include <winioctl.h>
#include <stdio.h>
#include <stdlib.h>
#pragma warning(disable:4996)
struct hid_device_ {
                HANDLE device_handle;
                BOOL blocking;
                USHORT output_report_length;
                size_t input_report_length;
                void *last_error_str;
                DWORD last_error_num;
                BOOL read_pending;
```

```
		char *read_buf;
		OVERLAPPED ol;
};
typedef struct hid_device_ hid_device;
typedef struct _HIDD_ATTRIBUTES{
	ULONG Size;
	USHORT VendorID;
	USHORT ProductID;
	USHORT VersionNumber;
} HIDD_ATTRIBUTES, *PHIDD_ATTRIBUTES;
typedef USHORT USAGE;
typedef struct _HIDP_CAPS {
	USAGE Usage;
	USAGE UsagePage;
	USHORT InputReportByteLength;
	USHORT OutputReportByteLength;
	USHORT FeatureReportByteLength;
	USHORT Reserved[17];
	USHORT fields_not_used_by_hidapi[10];
} HIDP_CAPS, *PHIDP_CAPS;
typedef void* PHIDP_PREPARSED_DATA;
typedef LONG NTSTATUS;
typedef NTSTATUS (__stdcall *HidP_GetCaps_)(PHIDP_PREPARSED_DATA preparsed_data, HIDP_CAPS
*caps);
static HMODULE lib_handle = NULL;
typedef BOOLEAN (__stdcall *HidD_GetAttributes_)(HANDLE device, PHIDD_ATTRIBUTES attrib);
typedef BOOLEAN (__stdcall *HidD_GetSerialNumberString_)(HANDLE device, PVOID buffer, ULONG
buffer_len);
typedef BOOLEAN (__stdcall *HidD_GetManufacturerString_)(HANDLE handle, PVOID buffer, ULONG
buffer_len);
typedef BOOLEAN (__stdcall *HidD_GetProductString_)(HANDLE handle, PVOID buffer, ULONG
buffer_len);
typedef BOOLEAN (__stdcall *HidD_SetFeature_)(HANDLE handle, PVOID data, ULONG length);
typedef BOOLEAN (__stdcall *HidD_GetFeature_)(HANDLE handle, PVOID data, ULONG length);
typedef BOOLEAN (__stdcall *HidD_GetIndexedString_)(HANDLE handle, ULONG string_index,
PVOID buffer, ULONG buffer_len);
typedef BOOLEAN (__stdcall *HidD_GetPreparsedData_)(HANDLE handle, PHIDP_PREPARSED_DATA
*preparsed_data);
typedef BOOLEAN (__stdcall *HidD_FreePreparsedData_)(PHIDP_PREPARSED_DATA preparsed_data);
typedef NTSTATUS (__stdcall *HidP_GetCaps_)(PHIDP_PREPARSED_DATA preparsed_data, HIDP_CAPS
*caps);
typedef BOOLEAN (__stdcall *HidD_SetNumInputBuffers_)(HANDLE handle, ULONG number_buffers);
static HidD_GetAttributes_ HidD_GetAttributes;
static HidD_GetSerialNumberString_ HidD_GetSerialNumberString;
static HidD_GetManufacturerString_ HidD_GetManufacturerString;
static HidD_GetProductString_ HidD_GetProductString;
static HidD_SetFeature_ HidD_SetFeature;
static HidD_GetFeature_ HidD_GetFeature;
static HidD_GetIndexedString_ HidD_GetIndexedString;
static HidD_GetPreparsedData_ HidD_GetPreparsedData;
static HidD_FreePreparsedData_ HidD_FreePreparsedData;
static HidP_GetCaps_ HidP_GetCaps;
static HidD_SetNumInputBuffers_ HidD_SetNumInputBuffers;
int hid_exit(void)
{
	FreeLibrary(lib_handle);
	return 0;
}
static hid_device *new_hid_device()
{
	hid_device *dev = (hid_device*)calloc(1, sizeof(hid_device));
	dev->device_handle = INVALID_HANDLE_VALUE;
	dev->blocking = FALSE;
	dev->output_report_length = 0;
	dev->input_report_length = 0;
	dev->last_error_str = NULL;
```

```
	dev->last_error_num = 0;
	dev->read_pending = FALSE;
	dev->read_buf = NULL;
	dev->ol.hEvent = NULL;
	return dev;
}
static int lookup_functions()
{
	lib_handle = LoadLibraryA("hid.dll");
	if (lib_handle) {
#define RESOLVE(x) x = (x##_)GetProcAddress(lib_handle, #x); if (!x) return -1;
		RESOLVE(HidD_GetAttributes);
		RESOLVE(HidD_GetSerialNumberString);
		RESOLVE(HidD_GetManufacturerString);
		RESOLVE(HidD_GetProductString);
		RESOLVE(HidD_SetFeature);
		RESOLVE(HidD_GetFeature);
		RESOLVE(HidD_GetIndexedString);
		RESOLVE(HidD_GetPreparsedData);
		RESOLVE(HidD_FreePreparsedData);
		RESOLVE(HidP_GetCaps);
		RESOLVE(HidD_SetNumInputBuffers);
#undef RESOLVE
	}
	else
		return -1;
	return 0;
}
static void free_hid_device(hid_device *dev)
{
	CloseHandle(dev->ol.hEvent);
	CloseHandle(dev->device_handle);
	LocalFree(dev->last_error_str);
	free(dev->read_buf);
	free(dev);
}
```

3. *hidread Win32 C++ DLL hidread.cpp source file*

```
#include "stdafx.h"
#include <windows.h>
#include <setupapi.h>
#include <winioctl.h>
#include <stdio.h>
#include <stdlib.h>
#include <cstdlib>
#include "hidapi.h"
DWORD Lbytes_read = 0;
DWORD Rbytes_read = 0;
#pragma warning(disable:4996)
#pragma comment(lib, "setupapi.lib")
extern "C"
{
	__declspec(dllexport) void Lhid_read_timeout(hid_device *dev, unsigned char *data,
size_t length)
	{
		ReadFile(dev->device_handle, dev->read_buf, dev->input_report_length,
&Lbytes_read, NULL);
		memcpy(data, dev->read_buf, length);
		CancelIoEx(dev->device_handle, NULL);
	}
	__declspec(dllexport) void Rhid_read_timeout(hid_device *dev, unsigned char *data,
size_t length)
	{
		ReadFile(dev->device_handle, dev->read_buf, dev->input_report_length,
&Rbytes_read, NULL);
		memcpy(data, dev->read_buf, length);
```

```
		CancelIoEx(dev->device_handle, NULL);
	}
	__declspec(dllexport) void hid_write(hid_device *dev, const unsigned char *data,
size_t length)
	{
		unsigned char *buf = (unsigned char *)malloc(dev->output_report_length);
		memcpy(buf, data, length);
		WriteFile(dev->device_handle, buf, dev->output_report_length, NULL, NULL);
		CancelIoEx(dev->device_handle, NULL);
	}
	__declspec(dllexport) hid_device *hid_open_path(HANDLE handle)
	{
		hid_device *dev;
		HIDP_CAPS caps;
		PHIDP_PREPARSED_DATA pp_data = NULL;
		lookup_functions();
		dev = new_hid_device();
		dev->device_handle = handle;
		HidD_GetPreparsedData(dev->device_handle, &pp_data);
		HidP_GetCaps(pp_data, &caps);
		dev->output_report_length = caps.OutputReportByteLength;
		dev->input_report_length = caps.InputReportByteLength;
		dev->read_buf = (char*)malloc(dev->input_report_length);
		hid_exit();
		return dev;
	}
	__declspec(dllexport) void hid_close(hid_device *dev)
	{
		CancelIoEx(dev->device_handle, NULL);
		free_hid_device(dev);
	}
}
```

4. C# Windows Form JoyconsTheory Code

```
using Microsoft.Win32.SafeHandles;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Threading;
using System.Diagnostics;
using System.Text.RegularExpressions;
using System.Numerics;
namespace JoyconsTheory
{
  public partial class Form1 : Form
  {
    public Form1()
    {
      InitializeComponent();
    }
    public static double WidthS, HeightS, keys123, keys456, keysEnterTab, irx2e, iry2e, irx3e, iry3e, irxe, irye, irx, iry, irxc,
iryc, iryn, irxpp, irypp, mousexi, mouseyi, mousexn, mouseyn, mousexpn, mouseypn, mousexbn, mouseybn, mousexp,
mouseyp, mousexpm, mouseypm, mousexpp, mouseypp, Rand1swps, Rand1swms, Rand2swyps, Rand2swyms, Rand2swps,
Rand2swms, signirx, signiry, absirx, absiry, aimpluscount, centercursorposcount, stickviewxinit, stickviewyinit,
readingfilecount, brinktitanfalltimecount, bo3timecount, calibrationinit, Acceleration, Breaking, pushrcount,
cancelreloadcount, rapidfirecount, keystodown, pushtodown, norecoilcount, antistcount, ticktimecount, watchK = 50,
```

```
watchK1 = 2, watchK2 = 0, watchM = 50, watchM1 = 2, watchM2 = 0;
    public static bool bool2swyps, bool2swyms, bool2swps, bool2swms, bool1swps, bool1swms, runningoff, Setrecenter,
_getstate, _Getstate, getstate, Getstate, mWSButtonStateAio, cancelreloadbool, foraorciso n, enableautoloadoflastfile,
notpressing1and2, reconfiguration, randA, randZ, readingfile, _value, lockchangefeaturesandoptions;
    public static bool[] _Valuechanged = new bool[95], _valuechanged = new bool[95], _Value = new bool[95];
    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 taskDLeft, taskDRight, taskM, taskK;
    public static uint CurrentResolution = 0;
    public static ThreadStart threadstart;
    public static Thread thread;
    private static bool ISLEFT, ISRIGHT;
    public static double RolljoyconrightAngle, _rolljoyconrightanglechanged;
    public static double mousex, mousey, keys5678;
    public static bool LeftButtonSHOULDER_2io, LeftButtonSTICKio;
    private double signchangewheelZ1, signchangewheelZ2;
    private bool signchangewheelZ;
    private static Stopwatch diffM = new Stopwatch(), diffK = new Stopwatch();
    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("JoyconsPairing.dll", EntryPoint = "connect")]
    public static unsafe extern int connect();
    [DllImport("JoyconsPairing.dll", EntryPoint = "disconnect")]
    public static unsafe extern bool disconnect();
    [DllImport("hid.dll")]
    public static unsafe extern void HidD_GetHidGuid(out Guid gHid);
    [DllImport("hid.dll")]
    public static unsafe extern 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 IntPtr CreateFile(string fileName, System.IO.FileAccess fileAccess, System.IO.FileShare
fileShare, IntPtr securityAttributes, System.IO.FileMode creationDisposition, EFileAttributes flags, IntPtr template);
    [DllImport("hidread.dll", CallingConvention = CallingConvention.Cdecl, EntryPoint = "Lhid_read_timeout")]
    public static unsafe extern int Lhid_read_timeout(SafeFileHandle dev, byte[] data, UIntPtr length);
    [DllImport("hidread.dll", CallingConvention = CallingConvention.Cdecl, EntryPoint = "Rhid_read_timeout")]
    public static unsafe extern int Rhid_read_timeout(SafeFileHandle dev, byte[] data, UIntPtr length);
    [DllImport("hidread.dll", CallingConvention = CallingConvention.Cdecl, EntryPoint = "hid_write")]
    public static unsafe extern int hid_write(SafeFileHandle device, byte[] data, UIntPtr length);
    [DllImport("hidread.dll", CallingConvention = CallingConvention.Cdecl, EntryPoint = "hid_open_path")]
    public static unsafe extern SafeFileHandle hid_open_path(IntPtr handle);
    [DllImport("hidread.dll", CallingConvention = CallingConvention.Cdecl, EntryPoint = "hid_close")]
    public static unsafe extern void hid_close(SafeFileHandle device);
```

```
    [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("user32.dll")]
    public static extern bool GetAsyncKeyState(System.Windows.Forms.Keys vKey);
    [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();
[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);
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();
}
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);
    txtBjoyconrighttofrontpush.Text = "1000";
    txtBcancelreload.Text = "1800";
    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";
    txtBRS2switchinterval.Text = "6";
    txtBRS2switchpressdelay.Text = "18";
    txtBticktime.Text = "6";
    txtBwheelscriptsticklimitin.Text = "33";
    txtBwheelscriptsticklimitout.Text = "2000";
    txtBzoningquantity.Text = "100";
    txtBzoninghardnessquantity.Text = "100";
    txtBnorecoilstepquantity.Text = "0";
    Fvar.Add("//joycon right to front push r time extra setting", 1000);
    Fvar.Add("//cancel reload waiting LS2 time extra setting", 1800);
    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("//RS2 switch interval time extra setting", 6);
    Fvar.Add("//RS2 switch press delay time extra setting", 18);
    Fvar.Add("//tick time", 6);
    Fvar.Add("//wheel script stick limit in", 33);
    Fvar.Add("//wheel script stick limit out", 2000);
    Fvar.Add("//zoning quantity", 100);
    Fvar.Add("//zoning hardness quantity", 100);
    Fvar.Add("//no recoil step quantity", 0);
    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("//LS2 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("//LS2 accuracy", false);
```

```
Fbool.Add("//LS2 view on", false);
Fbool.Add("//LS2 aim plus", false);
Fbool.Add("//xaim", false);
Fbool.Add("//RSR RSL view", false);
Fbool.Add("//wheel view", false);
Fbool.Add("//stick view", false);
Fbool.Add("//cancel reload x", false);
Fbool.Add("//push r 1'", false);
Fbool.Add("//LS2 RS2 switch", false);
Fbool.Add("//LS press I/O", false);
Fbool.Add("//driver mouse", false);
Fbool.Add("//driver keyboard", false);
Fbool.Add("//swap", false);
Abool.Add("//brink", false);
Abool.Add("//metro", false);
Abool.Add("//titanfall", false);
Abool.Add("//cursor", false);
Abool.Add("//LS2 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("//LS2 accuracy", false);
Abool.Add("//LS2 view on", false);
Abool.Add("//LS2 aim plus", false);
Abool.Add("//xaim", false);
Abool.Add("//RSR RSL view", false);
Abool.Add("//wheel view", false);
Abool.Add("//stick view", false);
Abool.Add("//cancel reload x", false);
Abool.Add("//push r 1'", false);
Abool.Add("//LS2 RS2 switch", false);
Abool.Add("//LS press I/O", false);
Abool.Add("//driver mouse", false);
Abool.Add("//driver keyboard", false);
Abool.Add("//swap", false);
action.Add("//cancel reload x", "");
action.Add("//joycon left stick", "");
action.Add("//joycon left up", "");
action.Add("//joycon left down", "");
action.Add("//joycon right S2", "");
action.Add("//joycon left S2", "");
action.Add("//joycon plus", "");
action.Add("//joycon minus", "");
action.Add("//joycon right stick", "");
action.Add("//joycon right home", "");
action.Add("//joycon right S1", "");
action.Add("//joycon left left", "");
action.Add("//joycon left right", "");
action.Add("//joycon left SL", "");
action.Add("//joycon right SR", "");
action.Add("//joycon right SL", "");
action.Add("//joycon left SR", "");
action.Add("//joycon right to front", "");
action.Add("//joycon left to front", "");
action.Add("//joycon left stick up", "");
action.Add("//joycon left S1", "");
action.Add("//joycon right stick down", "");
action.Add("//joycon left stick down", "");
action.Add("//joycon right stick left", "");
action.Add("//joycon right stick right", "");
action.Add("//joycon right stick up", "");
```

```
action.Add("//joycon left stick left", "");
action.Add("//joycon left stick right", "");
action.Add("//joycon right left", "");
action.Add("//joycon right right", "");
action.Add("//joycon right up", "");
action.Add("//joycon right down", "");
action.Add("//joycon left capture", "");
actionassign.Add("//cancel reload x", new Point2D());
actionassign.Add("//joycon left stick", new Point2D());
actionassign.Add("//joycon left up", new Point2D());
actionassign.Add("//joycon left down", new Point2D());
actionassign.Add("//joycon right S2", new Point2D());
actionassign.Add("//joycon left S2", new Point2D());
actionassign.Add("//joycon plus", new Point2D());
actionassign.Add("//joycon minus", new Point2D());
actionassign.Add("//joycon right stick", new Point2D());
actionassign.Add("//joycon right home", new Point2D());
actionassign.Add("//joycon right S1", new Point2D());
actionassign.Add("//joycon left left", new Point2D());
actionassign.Add("//joycon left right", new Point2D());
actionassign.Add("//joycon left SL", new Point2D());
actionassign.Add("//joycon right SR", new Point2D());
actionassign.Add("//joycon right SL", new Point2D());
actionassign.Add("//joycon left SR", new Point2D());
actionassign.Add("//joycon right to front", new Point2D());
actionassign.Add("//joycon left to front", new Point2D());
actionassign.Add("//joycon left stick up", new Point2D());
actionassign.Add("//joycon left S1", new Point2D());
actionassign.Add("//joycon right stick down", new Point2D());
actionassign.Add("//joycon left stick down", new Point2D());
actionassign.Add("//joycon right stick left", new Point2D());
actionassign.Add("//joycon right stick right", new Point2D());
actionassign.Add("//joycon right stick up", new Point2D());
actionassign.Add("//joycon left stick left", new Point2D());
actionassign.Add("//joycon left stick right", new Point2D());
actionassign.Add("//joycon right left", new Point2D());
actionassign.Add("//joycon right right", new Point2D());
actionassign.Add("//joycon right up", new Point2D());
actionassign.Add("//joycon right down", new Point2D());
actionassign.Add("//joycon left capture", new Point2D());
do
{
  Thread.Sleep(1);
  leftandright = connect();
}
while (leftandright != 1 & leftandright != 2 & leftandright != 3 & !notpressing1and2);
if (!notpressing1and2)
{
  if (leftandright == 3 | leftandright == 2)
    do
      Thread.Sleep(1);
    while (!ScanRight());
  if (leftandright == 1 | leftandright == 2)
    do
      Thread.Sleep(1);
    while (!ScanLeft());
  if (leftandright == 3 | leftandright == 2)
  {
    checkBox2.Checked = true;
    taskDRight = new Task(Joycon_thrDRight);
    taskDRight.Start();
  }
  if (leftandright == 1 | leftandright == 2)
```

```
            {
                checkBox1.Checked = true;
                taskDLeft = new Task(Joycon_thrDLeft);
                taskDLeft.Start();
            }
            System.Threading.Thread.Sleep(2000);
            if (leftandright == 3 | leftandright == 2)
            {
                stick_rawRight[0] = report_bufaRight[6 + (!ISRIGHT ? 0 : 3)];
                stick_rawRight[1] = report_bufaRight[7 + (!ISRIGHT ? 0 : 3)];
                stick_rawRight[2] = report_bufaRight[8 + (!ISRIGHT ? 0 : 3)];
                stick_calibrationRight[0] = (UInt16)(stick_rawRight[0] | ((stick_rawRight[1] & 0xf) << 8));
                stick_calibrationRight[1] = (UInt16)((stick_rawRight[1] >> 4) | (stick_rawRight[2] << 4));
                acc_gcalibrationRightX = (int)(avg((Int16)(report_bufRight[13 + 0 * 12] | ((report_bufRight[14 + 0 * 12] << 8) &
0xff00)), (Int16)(report_bufRight[13 + 1 * 12] | ((report_bufRight[14 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufRight[13 +
2 * 12] | ((report_bufRight[14 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f);
                acc_gcalibrationRightY = -(int)(avg((Int16)(report_bufRight[15 + 0 * 12] | ((report_bufRight[16 + 0 * 12] << 8) &
0xff00)), (Int16)(report_bufRight[15 + 1 * 12] | ((report_bufRight[16 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufRight[15 +
2 * 12] | ((report_bufRight[16 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f);
                acc_gcalibrationRightZ = -(int)(avg((Int16)(report_bufRight[17 + 0 * 12] | ((report_bufRight[18 + 0 * 12] << 8) &
0xff00)), (Int16)(report_bufRight[17 + 1 * 12] | ((report_bufRight[18 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufRight[17 +
2 * 12] | ((report_bufRight[18 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f);
                gyr_gcalibrationRightX = (int)(avg((int)((Int16)((report_bufRight[19 + 0 * 12] | ((report_bufRight[20 + 0 * 12] <<
8) & 0xff00)))), (int)((Int16)((report_bufRight[19 + 1 * 12] | ((report_bufRight[20 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufRight[19 + 2 * 12] | ((report_bufRight[20 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f);
                gyr_gcalibrationRightY = -(int)(avg((int)((Int16)((report_bufRight[21 + 0 * 12] | ((report_bufRight[22 + 0 * 12] <<
8) & 0xff00)))), (int)((Int16)((report_bufRight[21 + 1 * 12] | ((report_bufRight[22 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufRight[21 + 2 * 12] | ((report_bufRight[22 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f);
                gyr_gcalibrationRightZ = -(int)(avg((int)((Int16)((report_bufRight[23 + 0 * 12] | ((report_bufRight[24 + 0 * 12] <<
8) & 0xff00)))), (int)((Int16)((report_bufRight[23 + 1 * 12] | ((report_bufRight[24 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufRight[23 + 2 * 12] | ((report_bufRight[24 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f);
            }
            if (leftandright == 1 | leftandright == 2)
            {
                stick_rawLeft[0] = report_bufaLeft[6 + (ISLEFT ? 0 : 3)];
                stick_rawLeft[1] = report_bufaLeft[7 + (ISLEFT ? 0 : 3)];
                stick_rawLeft[2] = report_bufaLeft[8 + (ISLEFT ? 0 : 3)];
                stick_calibrationLeft[0] = (UInt16)(stick_rawLeft[0] | ((stick_rawLeft[1] & 0xf) << 8));
                stick_calibrationLeft[1] = (UInt16)((stick_rawLeft[1] >> 4) | (stick_rawLeft[2] << 4));
                acc_gcalibrationLeftX = (int)(avg((Int16)(report_bufLeft[13 + 0 * 12] | ((report_bufLeft[14 + 0 * 12] << 8) &
0xff00)), (Int16)(report_bufLeft[13 + 1 * 12] | ((report_bufLeft[14 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufLeft[13 + 2 *
12] | ((report_bufLeft[14 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f);
                acc_gcalibrationLeftY = (int)(avg((Int16)(report_bufLeft[15 + 0 * 12] | ((report_bufLeft[16 + 0 * 12] << 8) &
0xff00)), (Int16)(report_bufLeft[15 + 1 * 12] | ((report_bufLeft[16 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufLeft[15 + 2 *
12] | ((report_bufLeft[16 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f);
                acc_gcalibrationLeftZ = (int)(avg((Int16)(report_bufLeft[17 + 0 * 12] | ((report_bufLeft[18 + 0 * 12] << 8) &
0xff00)), (Int16)(report_bufLeft[17 + 1 * 12] | ((report_bufLeft[18 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufLeft[17 + 2 *
12] | ((report_bufLeft[18 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f);
                gyr_gcalibrationLeftX = (int)(avg((int)((Int16)((report_bufLeft[19 + 0 * 12] | ((report_bufLeft[20 + 0 * 12] << 8) &
0xff00)))), (int)((Int16)((report_bufLeft[19 + 1 * 12] | ((report_bufLeft[20 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufLeft[19 + 2 * 12] | ((report_bufLeft[20 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f);
                gyr_gcalibrationLeftY = (int)(avg((int)((Int16)((report_bufLeft[21 + 0 * 12] | ((report_bufLeft[22 + 0 * 12] << 8) &
0xff00)))), (int)((Int16)((report_bufLeft[21 + 1 * 12] | ((report_bufLeft[22 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufLeft[21 + 2 * 12] | ((report_bufLeft[22 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f);
                gyr_gcalibrationLeftZ = (int)(avg((int)((Int16)((report_bufLeft[23 + 0 * 12] | ((report_bufLeft[24 + 0 * 12] << 8) &
0xff00)))), (int)((Int16)((report_bufLeft[23 + 1 * 12] | ((report_bufLeft[24 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufLeft[23 + 2 * 12] | ((report_bufLeft[24 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f);
            }
            try
            {
                System.IO.StreamReader file = new System.IO.StreamReader("joyconsinitfile.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("joyconsinitfile.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(Joycons_thrK);
        taskK.Start();
        diffK.Start();
        taskM = new Task(Joycons_thrM);
        taskM.Start();
        diffM.Start();
    }
}
private double avg(double val1, double val2, double val3)
{
    return (new double[] { val1, val2, val3 }).Average();
}
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()
  {
    int countchange = 0;
    foreach (bool Value in _Value)
    {
      countchange++;
      if (Value & countchange <= 89)
      {
        _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["//joycon right left"].X,actionassign["//joycon right left"].Y);
        doSimulateKeyUp(actionassign["//joycon right right"].X, actionassign["//joycon right right"].Y);
        doSimulateKeyUp(actionassign["//joycon right up"].X, actionassign["//joycon right up"].Y);
        doSimulateKeyUp(actionassign["//joycon right down"].X, actionassign["//joycon right down"].Y);
        doSimulateKeyUp(actionassign["//joycon left left"].X, actionassign["//joycon left left"].Y);
        doSimulateKeyUp(actionassign["//joycon left right"].X,actionassign["//joycon left right"].Y);
        doSimulateKeyUp(actionassign["//joycon left up"].X, actionassign["//joycon left up"].Y);
        doSimulateKeyUp(actionassign["//joycon left down"].X, actionassign["//joycon left down"].Y);
        doSimulateKeyUp(actionassign["//joycon right S2"].X, actionassign["//joycon right S2"].Y);
        doSimulateKeyUp(actionassign["//joycon left S2"].X, actionassign["//joycon left S2"].Y);
        doSimulateKeyUp(actionassign["//joycon plus"].X, actionassign["//joycon plus"].Y);
        doSimulateKeyUp(actionassign["//joycon minus"].X, actionassign["//joycon minus"].Y);
        doSimulateKeyUp(actionassign["//joycon right stick"].X, actionassign["//joycon right stick"].Y);
        doSimulateKeyUp(actionassign["//joycon right home"].X, actionassign["//joycon right home"].Y);
        doSimulateKeyUp(actionassign["//joycon right S1"].X, actionassign["//joycon right S1"].Y);
        doSimulateKeyUp(actionassign["//joycon left SL"].X, actionassign["//joycon left SL"].Y);
        doSimulateKeyUp(actionassign["//joycon right SR"].X, actionassign["//joycon right SR"].Y);
        doSimulateKeyUp(actionassign["//joycon right SL"].X, actionassign["//joycon right SL"].Y);
        doSimulateKeyUp(actionassign["//joycon left SR"].X, actionassign["//joycon left SR"].Y);
        doSimulateKeyUp(actionassign["//joycon right to front"].X, actionassign["//joycon right to front"].Y);
        doSimulateKeyUp(actionassign["//joycon left to front"].X, actionassign["//joycon left to front"].Y);
        doSimulateKeyUp(actionassign["//joycon left stick up"].X, actionassign["//joycon left stick up"].Y);
        doSimulateKeyUp(actionassign["//joycon left S1"].X, actionassign["//joycon left S1"].Y);
        doSimulateKeyUp(actionassign["//joycon left stick"].X, actionassign["//joycon left stick"].Y);
        doSimulateKeyUp(actionassign["//joycon right stick down"].X, actionassign["//joycon right stick down"].Y);
        doSimulateKeyUp(actionassign["//joycon left stick down"].X, actionassign["//joycon left stick down"].Y);
        doSimulateKeyUp(actionassign["//joycon right stick left"].X, actionassign["//joycon right stick left"].Y);
        doSimulateKeyUp(actionassign["//joycon right stick right"].X, actionassign["//joycon rightstick right"].Y);
        doSimulateKeyUp(actionassign["//joycon right stick up"].X, actionassign["//joycon right stick up"].Y);
        doSimulateKeyUp(actionassign["//joycon left stick left"].X, actionassign["//joycon left stick left"].Y);
        doSimulateKeyUp(actionassign["//joycon left stick right"].X, actionassign["//joycon leftstick right"].Y);
        doSimulateKeyUp(actionassign["//cancel reload x"].X, actionassign["//cancel reload x"].Y);
        doSimulateKeyUp(actionassign["//joycon left capture"].X, actionassign["//joycon left capture"].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["//joycon right to front"] = cmBRTOFRONT.Items[0].ToString();
    action["//joycon plus"] = cmBRPLUS.Items[0].ToString();
    action["//joycon minus"] = cmBLMINUS.Items[0].ToString();
    action["//cancel reload x"] = cmBRCANCELRELOAD.Items[0].ToString();
    action["//joycon right home"] = cmBRHOME.Items[0].ToString().Replace("(R)", "");
    action["//joycon right S1"] = cmBRS1.Items[0].ToString();
    action["//joycon left SL"] = cmBLSL.Items[0].ToString();
    action["//joycon right SR"] = cmBRSR.Items[0].ToString();
    action["//joycon left stick left"] = cmBLSLEFT.Items[0].ToString();
    action["//joycon left stick"] = cmBLS.Items[0].ToString();
    action["//joycon left stick down"] = cmBLSDOWN.Items[0].ToString().Replace("(R)", "");
    action["//joycon left stick up"] = cmBLSUP.Items[0].ToString();
    action["//joycon left stick right"] = cmBLSRIGHT.Items[0].ToString();
    action["//joycon right stick"] = cmBRS.Items[0].ToString();
    action["//joycon right SL"] = cmBRSL.Items[0].ToString();
    action["//joycon left SR"] = cmBLSR.Items[0].ToString();
    action["//joycon left to front"] = cmBLTOFRONT.Items[0].ToString();
    action["//joycon right S2"] = cmBRS2.Items[0].ToString();
    action["//joycon left S2"] = cmBLS2.Items[0].ToString();
    action["//joycon left up"] = cmBLUP.Items[0].ToString();
    action["//joycon left down"] = cmBLDOWN.Items[0].ToString();
    action["//joycon left left"] = cmBLLEFT.Items[0].ToString();
    action["//joycon left right"] = cmBLRIGHT.Items[0].ToString();
    action["//joycon left S1"] = cmBLS1.Items[0].ToString();
    action["//joycon right stick down"] = cmBRSDOWN.Items[0].ToString();
    action["//joycon left stick"] = cmBLS.Items[0].ToString();
    action["//joycon right stick right"] = cmBRSRIGHT.Items[0].ToString();
    action["//joycon right stick left"] = cmBRSLEFT.Items[0].ToString();
    action["//joycon right stick up"] = cmBRSUP.Items[0].ToString();
    action["//joycon right left"] = cmBRLEFT.Items[0].ToString();
    action["//joycon right right"] = cmBRRIGHT.Items[0].ToString();
    action["//joycon right down"] = cmBRDOWN.Items[0].ToString();
    action["//joycon right up"] = cmBRUP.Items[0].ToString();
    action["//joycon left capture"] = cmBLCAPTURE.Items[0].ToString();
}
cmBRTOFRONT.Text = action["//joycon right to front"];
cmBRPLUS.Text =action["//joycon plus"];
cmBLMINUS.Text = action["//joycon minus"];
cmBRS1.Text =action["//joycon right S1"];
cmBLSL.Text = action["//joycon left SL"];
cmBRSR.Text = action["//joycon right SR"];
cmBLSUP.Text = action["//joycon left stick up"];
cmBLS.Text = action["//joycon left stick"];
cmBLSLEFT.Text = action["//joycon leftstick left"];
cmBLSRIGHT.Text =action["//joycon leftstick right"];
cmBLSDOWN.Text = action["//joycon left stick down"];
cmBRS.Text =action["//joycon right stick"];
cmBLTOFRONT.Text = action["//joycon left to front"];
cmBRS2.Text =action["//joycon right S2"];
cmBLS2.Text = action["//joycon left S2"];
cmBLUP.Text = action["//joycon left up"];
cmBLDOWN.Text = action["//joycon left down"];
cmBLLEFT.Text = action["//joycon left left"];
cmBLRIGHT.Text = action["//joycon left right"];
cmBLS1.Text = action["//joycon left S1"];
cmBLS.Text = action["//joycon left stick"];
cmBRSRIGHT.Text = action["//joycon rightstick right"];
cmBRSLEFT.Text = action["//joycon rightstick left"];
cmBRSLEFT.Text = action["//joycon rightstick left"];
cmBRSDOWN.Text = action["//joycon right stick down"];
cmBRHOME.Text =action["//joycon right home"];
cmBLSR.Text =action["//joycon left SR"];
cmBRSL.Text =action["//joycon right SL"];
```

```
    if (!chkBF9S.Checked)
        cmBRCANCELRELOAD.Text = " ";
    else
        cmBRCANCELRELOAD.Text = action["//cancel reload x"];
    cmBRLEFT.Text = action["//joycon right left"];
    cmBRRIGHT.Text =action["//joycon right right"];
    cmBRDOWN.Text = action["//joycon right down"];
    cmBRUP.Text = action["//joycon right up"];
    cmBLCAPTURE.Text = action["//joycon left capture"];
    actionassign["//joycon right to front"] = assign(action["//joycon right to front"]);
    actionassign["//joycon plus"] =assign(action["//joycon plus"]);
    actionassign["//joycon minus"] = assign(action["//joycon minus"]);
    actionassign["//cancel reload x"] =assign(action["//cancel reload x"]);
    actionassign["//joycon right home"] =assign(action["//joycon right home"]);
    actionassign["//joycon right S1"] = assign(action["//joycon right S1"]);
    actionassign["//joycon left SL"] = assign(action["//joycon left SL"]);
    actionassign["//joycon right SR"] = assign(action["//joycon right SR"]);
    actionassign["//joycon left stick up"] = assign(action["//joycon left stick up"]);
    actionassign["//joycon left stick"] = assign(action["//joycon leftstick"]);
    actionassign["//joycon left stick down"] = assign(action["//joycon leftstick down"]);
    actionassign["//joycon right stick up"] =assign(action["//joycon right stick up"]);
    actionassign["//joycon left stick right"] = assign(action["//joycon left stick right"]);
    actionassign["//joycon right stick"] = assign(action["//joycon rightstick"]);
    actionassign["//joycon right SL"] = assign(action["//joycon right SL"]);
    actionassign["//joycon left SR"] = assign(action["//joycon left SR"]);
    actionassign["//joycon left to front"] = assign(action["//joycon left to front"]);
    actionassign["//joycon right S2"] = assign(action["//joycon right S2"]);
    actionassign["//joycon left S2"] = assign(action["//joycon left S2"]);
    actionassign["//joycon left up"] = assign(action["//joycon left up"]);
    actionassign["//joycon left down"] = assign(action["//joycon left down"]);
    actionassign["//joycon left left"] = assign(action["//joycon left left"]);
    actionassign["//joycon left right"] =assign(action["//joycon left right"]);
    actionassign["//joycon left S1"] = assign(action["//joycon left S1"]);
    actionassign["//joycon right stick down"] =assign(action["//joycon right stick down"]);
    actionassign["//joycon left stick left"] = assign(action["//joycon leftstick left"]);
    actionassign["//joycon right stick right"] = assign(action["//joycon right stick right"]);
    actionassign["//joycon right stick left"] = assign(action["//joycon rightstick left"]);
    actionassign["//joycon right left"] =assign(action["//joycon right left"]);
    actionassign["//joycon right right"] = assign(action["//joycon right right"]);
    actionassign["//joycon right down"] = assign(action["//joycon right down"]);
    actionassign["//joycon right up"] = assign(action["//joycon right up"]);
    actionassign["//joycon left capture"] = assign(action["//joycon left capture"]);
}
public void SelectOptions()
{
    if (Fbool["//LS2 view on"])
    {
        this[3] = LeftButtonSHOULDER_2io | (LeftButtonSHOULDER_2 & 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;
        if (leftandright == 1 | leftandright == 2)
            checkBox1.Checked = true;
        if (leftandright == 3 | leftandright == 2)
            checkBox2.Checked = true;
    }
    else
        if (_Valuechanged[89] & this[89] & Getstate)
        {
            Getstate = false;
            checkBox1.Checked = false;
            checkBox2.Checked = false;
        }
    Setrecenter = !Getstate | (LeftButtonSHOULDER_1 & LeftButtonSHOULDER_2 & chkBF6S.Checked) |
(RightButtonSHOULDER_1 & RightButtonSHOULDER_2 & !chkBF6S.Checked);
    if (!Fbool["//lock features and options"] | lockchangefeaturesandoptions)
    {
        this[4] = GetAsyncKeyState(System.Windows.Forms.Keys.F1) & !_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;
        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;
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;
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;
if ((_Valuechanged[8] & !this[8]) | Abool["//LS2 press I/O"])
{
    if (Fbool["//LS2 press I/O"] == false)
    {
        Fbool["//LS2 press I/O"] = true;
        chkBF12.Checked = true;
    }
    else
    {
        Fbool["//LS2 press I/O"] = false;
        chkBF12.Checked = false;
    }
    Abool["//LS2 press I/O"] = false;
}
this[9] = GetAsyncKeyState(System.Windows.Forms.Keys.F5) & !_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;
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;
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;
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.F10) & _getstate;
if ((_Valuechanged[14] & !this[14]) | Abool["//wheel script"])
{
    if (Fbool["//wheel script"] == false)
    {
        Fbool["//wheel script"] = true;
        chkBF10S.Checked = true;
    }
    else
    {
        Fbool["//wheel script"] = false;
        chkBF10S.Checked = false;
    }
    Abool["//wheel script"] = false;
}
```

```
this[15] = GetAsyncKeyState(System.Windows.Forms.Keys.F11) & !_getstate;
if ((_Valuechanged[15] & !this[15]) | Abool["//LS2 accuracy"])
{
   if (Fbool["//LS2 accuracy"] == false)
   {
      Fbool["//LS2 accuracy"] = true;
      chkBF11.Checked = true;
   }
   else
   {
      Fbool["//LS2 accuracy"] = false;
      chkBF11.Checked = false;
   }
   Abool["//LS2 accuracy"] = false;
}
this[16] = GetAsyncKeyState(System.Windows.Forms.Keys.F7) & _getstate;
if ((_Valuechanged[16] & !this[16]) | Abool["//LS2 RS2 switch"])
{
   if (Fbool["//LS2 RS2 switch"] == false)
   {
      Fbool["//LS2 RS2 switch"] = true;
      chkBF7S.Checked = true;
   }
   else
   {
      Fbool["//LS2 RS2 switch"] = false;
      chkBF7S.Checked = false;
   }
   Abool["//LS2 RS2 switch"] = false;
}
this[17] = GetAsyncKeyState(System.Windows.Forms.Keys.F9) & _getstate;
if ((_Valuechanged[17] & !this[17]) | Abool["//cancel reload x"])
{
   if (Fbool["//cancel reload x"] == false)
   {
      Fbool["//cancel reload x"] = true;
      chkBF9S.Checked = true;
   }
   else
   {
      Fbool["//cancel reload x"] = false;
      chkBF9S.Checked = false;
   }
   Abool["//cancel reload x"] = false;
}
this[18] = GetAsyncKeyState(System.Windows.Forms.Keys.F1) & _getstate;
if ((_Valuechanged[18] & !this[18]) | Abool["//LS2 view on"])
{
   if (Fbool["//LS2 view on"] == false)
   {
      Fbool["//LS2 view on"] = true;
      chkBF1S.Checked = true;
   }
   else
   {
      Fbool["//LS2 view on"] = false;
      chkBF1S.Checked = false;
   }
   Abool["//LS2 view on"] = false;
}
this[20] = GetAsyncKeyState(System.Windows.Forms.Keys.F2) & _getstate;
if ((_Valuechanged[20] & !this[20]) | Abool["//LS2 aim plus"])
{
```

```
    if (Fbool["//LS2 aim plus"] == false)
    {
        Fbool["//LS2 aim plus"] = true;
        chkBF2S.Checked = true;
    }
    else
    {
        Fbool["//LS2 aim plus"] = false;
        chkBF2S.Checked = false;
    }
    Abool["//LS2 aim plus"] = false;
}
this[21] = GetAsyncKeyState(System.Windows.Forms.Keys.F9) & !_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;
if ((_Valuechanged[23] & !this[23]) | Abool["//LS press I/O"])
{
    if (Fbool["//LS press I/O"] == false)
    {
        Fbool["//LS press I/O"] = true;
        chkBF11S.Checked = true;
    }
    else
    {
        Fbool["//LS press I/O"] = false;
        chkBF11S.Checked = false;
    }
    Abool["//LS press I/O"] = false;
}
this[24] = GetAsyncKeyState(System.Windows.Forms.Keys.F5) & _getstate;
if ((_Valuechanged[24] & !this[24]) | Abool["//RSR RSL view"])
{
    if (Fbool["//RSR RSL view"] == false)
    {
        Fbool["//RSR RSL view"] = true;
        chkBF5S.Checked = true;
    }
    else
    {
        Fbool["//RSR RSL view"] = false;
        chkBF5S.Checked = false;
    }
    Abool["//RSR RSL view"] = false;
}
this[25] = GetAsyncKeyState(System.Windows.Forms.Keys.F8) & _getstate;
if ((_Valuechanged[25] & !this[25]) | Abool["//push r 1'"])
{
    if (Fbool["//push r 1'"] == false)
    {
        Fbool["//push r 1'"] = true;
```

```
        chkBF8S.Checked = true;
    }
    else
    {
        Fbool["//push r 1'"] = false;
        chkBF8S.Checked = false;
    }
    Abool["//push r 1'"] = false;
}
this[27] = GetAsyncKeyState(System.Windows.Forms.Keys.F4) & _getstate;
if ((_Valuechanged[27] & !this[27]) | Abool["//wheel view"])
{
    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[29] = GetAsyncKeyState(System.Windows.Forms.Keys.F3) & _getstate;
if ((_Valuechanged[29] & !this[29]) | Abool["//stick view"])
{
    if (Fbool["//stick view"] == false)
    {
        Fbool["//stick view"] = true;
        chkBF3S.Checked = true;
    }
    else
    {
        Fbool["//stick view"] = false;
        chkBF3S.Checked = false;
    }
    Abool["//stick view"] = false;
    switchwheelfix();
}
this[85] = GetAsyncKeyState(System.Windows.Forms.Keys.F12) & _getstate;
if ((_Valuechanged[85] & !this[85]) | Abool["//driver keyboard"])
{
    if (Fbool["//driver keyboard"] == false)
    {
        Fbool["//driver keyboard"] = true;
        chkBF12S.Checked = true;
    }
    else
    {
        Fbool["//driver keyboard"] = false;
        chkBF12S.Checked = false;
    }
    Abool["//driver keyboard"] = false;
}
this[63] = GetAsyncKeyState(System.Windows.Forms.Keys.F10) & !_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[90] = GetAsyncKeyState(System.Windows.Forms.Keys.F6) & _getstate;
        if ((_Valuechanged[90] & !this[90]) | Abool["//swap"])
        {
            if (Fbool["//swap"] == false)
            {
                Fbool["//swap"] = true;
                chkBF6S.Checked = true;
            }
            else
            {
                Fbool["//swap"] = false;
                chkBF6S.Checked = false;
            }
            Abool["//swap"] = false;
        }
        if (lockchangefeaturesandoptions)
            lockchangefeaturesandoptions = false;
    }
}
private void Joycons()
{
    readingfilecount += watchK;
    if (readingfilecount > 100f)
    {
        SelectOptions();
        readingfilecount = 0;
    }
    ticktimecount++;
    if (!reconfiguration & ticktimecount >= (Fvar["//tick time"] < 1 ? 1 : Fvar["//tick time"]))
    {
        foraorcison = (LeftButtonMINUS | RightButtonPLUS | RightButtonHOME | (GetAccelRight().X > 1.5f |
GetAccelRight().X < -1.5f) | (GetAccelLeft().X > 1.5f | GetAccelLeft().X < -1.5f) | (GetStickLeft()[1] > 0.33f & LeftButtonSTICK)
| LeftButtonDPAD_UP | LeftButtonDPAD_DOWN | LeftButtonDPAD_LEFT | LeftButtonDPAD_RIGHT);
        if (!Fbool["//LS2 press I/O"])
            LeftButtonSHOULDER_2io = LeftButtonSHOULDER_2;
        else
        {
            this[64] = LeftButtonSHOULDER_2;
            if (_Valuechanged[64] & this[64])
                if (!randA)
                {
                    LeftButtonSHOULDER_2io = true;
                    randA = true;
                }
                else
                    if (randA)
                    {
                        LeftButtonSHOULDER_2io = false;
                        randA = false;
                    }
            if (LeftButtonSHOULDER_2io & foraorcison & !Fbool["//LS2 view on"])
            {
                LeftButtonSHOULDER_2io = false;
                randA = false;
            }
```

```
        }
        if (!Fbool["//LS press I/O"])
            LeftButtonSTICKio = LeftButtonSTICK;
        else
        {
            this[88] = LeftButtonSTICK;
            if (_Valuechanged[88] & this[88])
                if (!randZ)
                {
                    LeftButtonSTICKio = true;
                    randZ = true;
                }
                else
                    if (randZ)
                    {
                        LeftButtonSTICKio = false;
                        randZ = false;
                    }
        }
        if (RightButtonSHOULDER_2 & LeftButtonSHOULDER_2 & Fbool["//LS2 RS2 switch"])
        {
            rapidfirecount++;
            if (rapidfirecount == 1)
                miceevent(actionassign["//joycon right S2"].X, actionassign["//joycon right S2"].Y);
            if (rapidfirecount == (int)(Fvar["//RS2 switch press delay time extra setting"] / (Fvar["//tick time"] < 1f ? watchK :
Fvar["//tick time"] * watchK)))
                miceeventf(actionassign["//joycon right S2"].X, actionassign["//joycon right S2"].Y);
            if (rapidfirecount >= (int)(Fvar["//RS2 switch press delay time extra setting"] / (Fvar["//tick time"] < 1f ? watchK :
Fvar["//tick time"] * watchK) + Fvar["//RS2 switch interval time extra setting"] / (Fvar["//tick time"] < 1f ? watchK :
Fvar["//tick time"] * watchK)))
                rapidfirecount = 0;
        }
        this[76] = RightButtonSHOULDER_2 & LeftButtonSHOULDER_2 & Fbool["//LS2 RS2 switch"];
        if (_Valuechanged[76] & !this[76])
        {
            miceeventf(actionassign["//joycon right S2"].X, actionassign["//joycon right S2"].Y);
            rapidfirecount = 0;
        }
        this[30] = RightButtonSHOULDER_2;
        if (_Valuechanged[30] & this[30])
            miceevent(actionassign["//joycon right S2"].X, actionassign["//joycon right S2"].Y);
        if (_Valuechanged[30] & !this[30])
            miceeventf(actionassign["//joycon right S2"].X, actionassign["//joycon right S2"].Y);
        this[66] = (LeftButtonSHOULDER_2 & foraorcison) | LeftButtonSHOULDER_2io;
        if (_Valuechanged[66] & this[66])
            miceevent(actionassign["//joycon left S2"].X, actionassign["//joycon left S2"].Y);
        if (_Valuechanged[66] & !this[66])
            miceeventf(actionassign["//joycon left S2"].X, actionassign["//joycon left S2"].Y);
        if (Fbool["//push r 1'"] & pushrcount >= 1)
        {
            pushrcount++;
            if (pushrcount == 2)
                miceevent(actionassign["//joycon right to front"].X,actionassign["//joycon right to front"].Y);
            if (pushrcount >= Fvar["//joycon right to front push r time extra setting"] / (Fvar["//tick time"] < 1f ? watchK :
Fvar["//tick time"] * watchK))
            {
                miceeventf(actionassign["//joycon right to front"].X, actionassign["//joycon right to front"].Y);
                pushrcount = 0;
            }
        }
        this[32] = (GetAccelRight().X > 1.5f | GetAccelRight().X < -1.5f);
        if (_Valuechanged[32] & this[32])
            miceevent(actionassign["//joycon right to front"].X, actionassign["//joycon right to front"].Y);
```

```
            if (_Valuechanged[32] & !this[32])
            {
               miceeventf(actionassign["//joycon right to front"].X, actionassign["//joycon right to front"].Y);
               if (Fbool["//push r 1'"])
                  pushrcount = 1;
            }
            this[33] = RightButtonPLUS;
            if (_Valuechanged[33] & this[33])
               miceevent(actionassign["//joycon plus"].X, actionassign["//joycon plus"].Y);
            if (_Valuechanged[33] & !this[33])
               miceeventf(actionassign["//joycon plus"].X, actionassign["//joycon plus"].Y);
            this[34] = LeftButtonMINUS;
            if (_Valuechanged[34] & this[34])
               miceevent(actionassign["//joycon minus"].X, actionassign["//joycon minus"].Y);
            if (_Valuechanged[34] & !this[34])
               miceeventf(actionassign["//joycon minus"].X, actionassign["//joycon minus"].Y);
            if (Fbool["//cancel reload x"])
            {
               if (GetAccelRight().X > 1.5f | GetAccelRight().X < -1.5f)
                  cancelreloadcount = 1;
               if (cancelreloadcount >= 1)
                  cancelreloadcount++;
               if (LeftButtonSHOULDER_2 & cancelreloadcount >= 1)
                  cancelreloadbool = true;
               else
                  cancelreloadbool = false;
               if (cancelreloadcount >= Fvar["//cancel reload waiting LS2 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;
               }
            }
            this[35] = LeftButtonDPAD_UP;
            if (_Valuechanged[35] & this[35])
               miceevent(actionassign["//joycon left up"].X, actionassign["//joycon left up"].Y);
            if (_Valuechanged[35] & !this[35])
               miceeventf(actionassign["//joycon left up"].X, actionassign["//joycon left up"].Y);
            this[36] = LeftButtonDPAD_DOWN;
            if (_Valuechanged[36] & this[36])
               miceevent(actionassign["//joycon left down"].X, actionassign["//joycon left down"].Y);
            if (_Valuechanged[36] & !this[36])
               miceeventf(actionassign["//joycon left down"].X, actionassign["//joycon left down"].Y);
            this[39] = LeftButtonDPAD_LEFT;
            if (_Valuechanged[39] & this[39])
               miceevent(actionassign["//joycon left left"].X, actionassign["//joycon left left"].Y);
            if (_Valuechanged[39] & !this[39])
               miceeventf(actionassign["//joycon left left"].X, actionassign["//joycon left left"].Y);
            this[40] = LeftButtonDPAD_RIGHT;
            if (_Valuechanged[40] & this[40])
               miceevent(actionassign["//joycon left right"].X, actionassign["//joycon left right"].Y);
            if (_Valuechanged[40] & !this[40])
               miceeventf(actionassign["//joycon left right"].X, actionassign["//joycon left right"].Y);
            this[37] = RightButtonHOME;
            if (_Valuechanged[37] & this[37])
               miceevent(actionassign["//joycon right home"].X, actionassign["//joycon right home"].Y);
            if (_Valuechanged[37] & !this[37])
               miceeventf(actionassign["//joycon right home"].X, actionassign["//joycon right home"].Y);
```

```
            this[38] = RightButtonSHOULDER_1;
            if (_Valuechanged[38] & this[38])
               miceevent(actionassign["//joycon right S1"].X, actionassign["//joycon right S1"].Y);
            if (_Valuechanged[38] & !this[38])
               miceeventf(actionassign["//joycon right S1"].X, actionassign["//joycon right S1"].Y);
            this[41] = LeftButtonSL;
            if (_Valuechanged[41] & this[41])
               miceevent(actionassign["//joycon left SL"].X, actionassign["//joycon left SL"].Y);
            if (_Valuechanged[41] & !this[41])
               miceeventf(actionassign["//joycon left SL"].X, actionassign["//joycon left SL"].Y);
            this[42] = RightButtonSR;
            if (_Valuechanged[42] & this[42])
               miceevent(actionassign["//joycon right SR"].X, actionassign["//joycon right SR"].Y);
            if (_Valuechanged[42] & !this[42])
               miceeventf(actionassign["//joycon right SR"].X, actionassign["//joycon right SR"].Y);
            this[44] = LeftButtonSHOULDER_1;
            if (_Valuechanged[44] & this[44])
               miceevent(actionassign["//joycon left S1"].X, actionassign["//joycon left S1"].Y);
            if (_Valuechanged[44] & !this[44])
               miceeventf(actionassign["//joycon left S1"].X, actionassign["//joycon left S1"].Y);
            this[45] = (LeftButtonSTICK | LeftButtonSTICKio);
            if (_Valuechanged[45] & this[45])
               miceevent(actionassign["//joycon left stick"].X, actionassign["//joycon left stick"].Y);
            if (_Valuechanged[45] & !this[45])
               miceeventf(actionassign["//joycon left stick"].X, actionassign["//joycon left stick"].Y);
            this[46] = GetStickRight()[1] < -0.33f;
            if (_Valuechanged[46] & this[46])
               miceevent(actionassign["//joycon right stick down"].X, actionassign["//joycon right stick down"].Y);
            if (_Valuechanged[46] & !this[46])
               miceeventf(actionassign["//joycon right stick down"].X, actionassign["//joycon right stick down"].Y);
            this[48] = GetStickRight()[0] < -0.33f;
            if (_Valuechanged[48] & this[48])
               miceevent(actionassign["//joycon right stick left"].X, actionassign["//joycon right stick left"].Y);
            if (_Valuechanged[48] & !this[48])
               miceeventf(actionassign["//joycon right stick left"].X, actionassign["//joycon right stick left"].Y);
            this[49] = GetStickRight()[0] > 0.33f;
            if (_Valuechanged[49] & this[49])
               miceevent(actionassign["//joycon right stick right"].X, actionassign["//joycon right stick right"].Y);
            if (_Valuechanged[49] & !this[49])
               miceeventf(actionassign["//joycon right stick right"].X, actionassign["//joycon right stick right"].Y);
            this[50] = GetStickRight()[1] > 0.33f;
            if (_Valuechanged[50] & this[50])
               miceevent(actionassign["//joycon right stick up"].X, actionassign["//joycon right stick up"].Y);
            if (_Valuechanged[50] & !this[50])
               miceeventf(actionassign["//joycon right stick up"].X, actionassign["//joycon right stick up"].Y);
            this[55] = RightButtonSTICK;
            if (_Valuechanged[55] & this[55])
               miceevent(actionassign["//joycon right stick"].X, actionassign["//joycon right stick"].Y);
            if (_Valuechanged[55] & !this[55])
               miceeventf(actionassign["//joycon right stick"].X, actionassign["//joycon right stick"].Y);
            if (Fbool["//wheel script"])
            {
               if (GetStickLeft()[0] > 0.15f)
                  Rand2swps = Rand2swps + GetStickLeft()[0];
               if (GetStickLeft()[0] < -0.15f)
                  Rand2swms = Rand2swms + GetStickLeft()[0];
               if (GetStickLeft()[1] > 0.15f)
                  Rand2swyps = Rand2swyps + GetStickLeft()[1];
               if (GetStickLeft()[1] < -0.15f)
                  Rand2swyms = Rand2swyms + GetStickLeft()[1];
               if (Rand2swps >= Fvar["//wheel script stick limit out"] / 100f / (Fvar["//tick time"] < 1f ? watchK : Fvar["//tick
time"] * watchK))
               {
```

```
                bool2swps = true;
                Rand2swps = 0;
            }
            else
                bool2swps = false;
            if (Rand2swms <= -Fvar["//wheel script stick limit out"] / 100f / (Fvar["//tick time"] < 1f ? watchK : Fvar["//tick
time"] * watchK))
            {
                bool2swms = true;
                Rand2swms = 0;
            }
            else
                bool2swms = false;
            if (Rand2swyps >= Fvar["//wheel script stick limit out"] / 100f / (Fvar["//tick time"] < 1f ? watchK : Fvar["//tick
time"] * watchK))
            {
                bool2swyps = true;
                Rand2swyps = 0;
            }
            else
                bool2swyps = false;
            if (Rand2swyms <= -Fvar["//wheel script stick limit out"] / 100f / (Fvar["//tick time"] < 1f ? watchK : Fvar["//tick
time"] * watchK))
            {
                bool2swyms = true;
                Rand2swyms = 0;
            }
            else
                bool2swyms = false;
            this[56] = (GetStickLeft()[0] > 0.33f & !Fbool["//wheel script"]) | ((GetStickLeft()[0] >= Fvar["//wheel script stick
limit in"] / 100f | bool2swps) & Fbool["//wheel script"]);
            this[57] = (GetStickLeft()[0] < -0.33f & !Fbool["//wheel script"]) | ((GetStickLeft()[0] <= -Fvar["//wheel script stick
limit in"] / 100f | bool2swms) & Fbool["//wheel script"]);
            this[58] = (GetStickLeft()[1] > 0.33f & !Fbool["//wheel script"]) | ((GetStickLeft()[1] >= Fvar["//wheel script stick
limit in"] / 100f | bool2swyps) & Fbool["//wheel script"]);
            this[59] = (GetStickLeft()[1] < -0.33f & !Fbool["//wheel script"]) | ((GetStickLeft()[1] <= -Fvar["//wheel script stick
limit in"] / 100f | bool2swyms) & Fbool["//wheel script"]);
            if (_Value changed[56] & this[56])
                micevent(actionassign["//joycon left stick right"].X, actionassign["//joycon left stick right"].Y);
            if (_Value changed[56] & !this[56])
                miceventf(actionassign["//joycon left stick right"].X, actionassign["//joycon left stick right"].Y);
            if (_Value changed[57] & this[57])
                micevent(actionassign["//joycon left stick left"].X, actionassign["//joycon left stick left"].Y);
            if (_Value changed[57] & !this[57])
                miceventf(actionassign["//joycon left stick left"].X, actionassign["//joycon left stick left"].Y);
            if (_Value changed[58] & this[58])
                micevent(actionassign["//joycon left stick up"].X, actionassign["//joycon left stick up"].Y);
            if (_Value changed[58] & !this[58])
                miceventf(actionassign["//joycon left stick up"].X, actionassign["//joycon left stick up"].Y);
            if (_Value changed[59] & this[59])
                micevent(actionassign["//joycon left stick down"].X, actionassign["//joycon left stick down"].Y);
            if (_Value changed[59] & !this[59])
                miceventf(actionassign["//joycon left stick down"].X, actionassign["//joycon left stick down"].Y);
        }
        else
        {
            this[51] = GetStickLeft()[0] < -0.33f;
            if (_Value changed[51] & this[51])
                micevent(actionassign["//joycon left stick left"].X, actionassign["//joycon left stick left"].Y);
            if (_Value changed[51] & !this[51])
                miceventf(actionassign["//joycon left stick left"].X, actionassign["//joycon left stick left"].Y);
            this[52] = GetStickLeft()[0] > 0.33f;
            if (_Value changed[52] & this[52])
```

```
                miceevent(actionassign["//joycon left stick right"].X, actionassign["//joycon left stick right"].Y);
            if (_Valuechanged[52] & !this[52])
                miceeventf(actionassign["//joycon left stick right"].X, actionassign["//joycon left stick right"].Y);
            this[47] = GetStickLeft()[1] < -0.33f;
            if (_Valuechanged[47] & this[47])
                miceevent(actionassign["//joycon left stick down"].X, actionassign["//joycon left stick down"].Y);
            if (_Valuechanged[47] & !this[47])
                miceeventf(actionassign["//joycon left stick down"].X, actionassign["//joycon left stick down"].Y);
            this[43] = GetStickLeft()[1] > 0.33f;
            if (_Valuechanged[43] & this[43])
                miceevent(actionassign["//joycon left stick up"].X, actionassign["//joycon left stick up"].Y);
            if (_Valuechanged[43] & !this[43])
                miceeventf(actionassign["//joycon left stick up"].X, actionassign["//joycon left stick up"].Y);
        }
        this[62] = (GetAccelLeft().X > 1.5f | GetAccelLeft().X < -1.5f) & !(GetAccelRight().X > 1.5f | GetAccelRight().X < -
1.5f);
        if (_Valuechanged[62] & this[62])
            miceevent(actionassign["//joycon left to front"].X, actionassign["//joycon left to front"].Y);
        if (_Valuechanged[62] & !this[62])
            miceeventf(actionassign["//joycon left to front"].X, actionassign["//joycon left to front"].Y);
        this[60] = RightButtonSL;
        if (_Valuechanged[60] & this[60])
            miceevent(actionassign["//joycon right SL"].X, actionassign["//joycon right SL"].Y);
        if (_Valuechanged[60] & !this[60])
            miceeventf(actionassign["//joycon right SL"].X, actionassign["//joycon right SL"].Y);
        this[61] = LeftButtonSR;
        if (_Valuechanged[61] & this[61])
            miceevent(actionassign["//joycon left SR"].X, actionassign["//joycon left SR"].Y);
        if (_Valuechanged[61] & !this[61])
            miceeventf(actionassign["//joycon left SR"].X, actionassign["//joycon left SR"].Y);
        this[22] = RightButtonDPAD_UP;
        if (_Valuechanged[22] & this[22])
            miceevent(actionassign["//joycon right up"].X, actionassign["//joycon right up"].Y);
        if (_Valuechanged[22] & !this[22])
            miceeventf(actionassign["//joycon right up"].X, actionassign["//joycon right up"].Y);
        this[26] = RightButtonDPAD_DOWN;
        if (_Valuechanged[26] & this[26])
            miceevent(actionassign["//joycon right down"].X, actionassign["//joycon right down"].Y);
        if (_Valuechanged[26] & !this[26])
            miceeventf(actionassign["//joycon right down"].X, actionassign["//joycon right down"].Y);
        this[28] = RightButtonDPAD_LEFT;
        if (_Valuechanged[28] & this[28])
            miceevent(actionassign["//joycon right left"].X, actionassign["//joycon right left"].Y);
        if (_Valuechanged[28] & !this[28])
            miceeventf(actionassign["//joycon right left"].X, actionassign["//joycon right left"].Y);
        this[31] = RightButtonDPAD_RIGHT;
        if (_Valuechanged[31] & this[31])
            miceevent(actionassign["//joycon right right"].X, actionassign["//joycon right right"].Y);
        if (_Valuechanged[31] & !this[31])
            miceeventf(actionassign["//joycon right right"].X, actionassign["//joycon right right"].Y);
        this[65] = LeftButtonCAPTURE;
        if (_Valuechanged[65] & this[65])
            miceevent(actionassign["//joycon left capture"].X, actionassign["//joycon left capture"].Y);
        if (_Valuechanged[65] & !this[65])
            miceeventf(actionassign["//joycon left capture"].X, actionassign["//joycon left capture"].Y);
        ticktimecount = 0;
    }
}
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 JoyconsIR()
{
    if (!Fbool["//swap"])
    {
        if (Fbool["//stick view"] | Fbool["//wheel view"])
        {
            if (Fbool["//stick view"] & !Fbool["//wheel view"] & !Fbool["//RSR RSL view"])
            {
                irxpp = -GetStickLeft()[0] * 1024f;
                irypp = -GetStickLeft()[1] * 1024f;
            }
            if (Fbool["//stick view"] & !Fbool["//wheel view"] & Fbool["//RSR RSL view"])
                irxpp = GetStickLeft()[1] * 1024f;
            if (Fbool["//wheel view"] & !Fbool["//stick view"])
            {
                if (!signchangewheelZ)
                    signchangewheelZ1 = signchangewheelZ2;
                signchangewheelZ2 = (double)DirectAnglesRight.X;
                if (DirectAnglesRight.X > 0.8f | DirectAnglesRight.X < -0.8f)
                {
                    if (Math.Sign(signchangewheelZ1) != Math.Sign(DirectAnglesRight.X))
                    {
                        signchangewheelZ2 = signchangewheelZ1;
                        signchangewheelZ = true;
                    }
                    else
                    {
                        signchangewheelZ2 = (double)DirectAnglesRight.X;
                        signchangewheelZ = false;
                    }
                }
                if (!signchangewheelZ)
                    irxpp = (DirectAnglesRight.X * 1024f) / 1f;
                if (!Fbool["//RSR RSL view"])
                    irypp = -(DirectAnglesRight.Y * 1024f) / 0.5f;
            }
            if (Fbool["//RSR RSL view"] & Fbool["//wheel view"] & !Fbool["//stick view"])
            {
                if (RightButtonSR & Acceleration >= -1024f)
                    Acceleration -= 3f * watchM;
                if (!RightButtonSR & Acceleration <= 0)
                    Acceleration += 3f * watchM;
                if (RightButtonSL & Breaking <= 1024f)
                    Breaking += 3f * watchM;
                if (!RightButtonSL & Breaking >= 0)
                    Breaking -= 3f * watchM;
                if (Acceleration >= 0)
                    Acceleration = 0;
                if (Breaking <= 0)
                    Breaking = 0;
                if (RightButtonSR & !RightButtonSL)
                    Breaking = 0;
                if (!RightButtonSR & RightButtonSL)
                    Acceleration = 0;
                irypp = Acceleration + Breaking;
            }
```

```
    if (Fbool["//RSR RSL view"] & !Fbool["//wheel view"] & Fbool["//stick view"])
    {
      if (LeftButtonSR & Acceleration >= -1024f)
        Acceleration -= 3f * watchM;
      if (!LeftButtonSR & Acceleration <= 0)
        Acceleration += 3f * watchM;
      if (LeftButtonSL & Breaking <= 1024f)
        Breaking += 3f * watchM;
      if (!LeftButtonSL & Breaking >= 0)
        Breaking -= 3f * watchM;
      if (Acceleration >= 0)
        Acceleration = 0;
      if (Breaking <= 0)
        Breaking = 0;
      if (LeftButtonSR & !LeftButtonSL)
        Breaking = 0;
      if (!LeftButtonSR & LeftButtonSL)
        Acceleration = 0;
      irypp = Acceleration + Breaking;
    }
  }
  else
  {
    irxe = -(EulerAnglesRight.X * 1024f) / 0.5f;
    irye = -(EulerAnglesRight.Z * 1024f) / 0.5f;
  }
}
else
{
  if (Fbool["//stick view"] | Fbool["//wheel view"])
  {
    if (Fbool["//stick view"] & !Fbool["//wheel view"] & !Fbool["//RSR RSL view"])
    {
      irxpp = -GetStickRight()[0] * 1024f;
      irypp = -GetStickRight()[1] * 1024f;
    }
    if (Fbool["//stick view"] & !Fbool["//wheel view"] & Fbool["//RSR RSL view"])
      irxpp = -GetStickRight()[1] * 1024f;
    if (Fbool["//wheel view"] & !Fbool["//stick view"])
    {
      if (!signchangewheelZ)
        signchangewheelZ1 = signchangewheelZ2;
      signchangewheelZ2 = (double)DirectAnglesLeft.X;
      if (DirectAnglesLeft.X > 0.8f | DirectAnglesLeft.X < -0.8f)
      {
        if (Math.Sign(signchangewheelZ1) != Math.Sign(DirectAnglesLeft.X))
        {
          signchangewheelZ2 = signchangewheelZ1;
          signchangewheelZ = true;
        }
        else
        {
          signchangewheelZ2 = (double)DirectAnglesLeft.X;
          signchangewheelZ = false;
        }
      }
      if (!signchangewheelZ)
        irxpp = -(DirectAnglesLeft.X * 1024f) / 1f;
      if (!Fbool["//RSR RSL view"])
        irypp = (DirectAnglesLeft.Y * 1024f) / 0.5f;
    }
    if (Fbool["//RSR RSL view"] & Fbool["//wheel view"] & !Fbool["//stick view"])
    {
```

```
            if (LeftButtonSR & Acceleration >= -1024f)
              Acceleration -= 3f * watchM;
            if (!LeftButtonSR & Acceleration <= 0)
              Acceleration += 3f * watchM;
            if (LeftButtonSL & Breaking <= 1024f)
              Breaking += 3f * watchM;
            if (!LeftButtonSL & Breaking >= 0)
              Breaking -= 3f * watchM;
            if (Acceleration >= 0)
              Acceleration = 0;
            if (Breaking <= 0)
              Breaking = 0;
            if (LeftButtonSR & !LeftButtonSL)
              Breaking = 0;
            if (!LeftButtonSR & LeftButtonSL)
              Acceleration = 0;
            irypp = Acceleration + Breaking;
          }
          if (Fbool["//RSR RSL view"] & !Fbool["//wheel view"] & Fbool["//stick view"])
          {
            if (RightButtonSR & Acceleration >= -1024f)
              Acceleration -= 3f * watchM;
            if (!RightButtonSR & Acceleration <= 0)
              Acceleration += 3f * watchM;
            if (RightButtonSL & Breaking <= 1024f)
              Breaking += 3f * watchM;
            if (!RightButtonSL & Breaking >= 0)
              Breaking -= 3f * watchM;
            if (Acceleration >= 0)
              Acceleration = 0;
            if (Breaking <= 0)
              Breaking = 0;
            if (RightButtonSR & !RightButtonSL)
              Breaking = 0;
            if (!RightButtonSR & RightButtonSL)
              Acceleration = 0;
            irypp = Acceleration + Breaking;
          }
        }
        else
        {
          irxe = -(EulerAnglesLeft.X * 1024f) / 0.5f;
          irye = -(EulerAnglesLeft.Z * 1024f) / 0.5f;
        }
      }
      if (!Fbool["//stick view"] & !Fbool["//wheel view"])
      {
        if ((Fbool["//LS2 aim plus"] | Fbool["//LS2 accuracy"]) & (LeftButtonSHOULDER_2io | (LeftButtonSHOULDER_2 &
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 >= 0 ? Scale(irxe, 0f, 1024f, (Fvar["//aim speed accuracy multipler of center axis x extra setting"] *
```

```
(Fbool["//LS2 accuracy"] ? aimpluscount / Fvar["//aim plus latency time extra setting"] : 1)) / 100f, 1024f) : Scale(irxe, -1024f, 0f, -1024f, -(Fvar["//aim speed accuracy multipler of center axis x extra setting"] * (Fbool["//LS2 accuracy"] ? aimpluscount / Fvar["//aim plus latency time extra setting"] : 1)) / 100f);
            else
                irx = irxe;
            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 >= 0 ? Scale(irye, 0f, 1024f, (Fvar["//aim speed accuracy multipler of center axis y extra setting"] * (Fbool["//LS2 accuracy"] ? aimpluscount / Fvar["//aim plus latency time extra setting"] : 1)) / 100f, 1024f) : Scale(irye, -1024f, 0f, -1024f, -(Fvar["//aim speed accuracy multipler of center axis y extra setting"] * (Fbool["//LS2 accuracy"] ? aimpluscount / Fvar["//aim plus latency time extra setting"] : 1)) / 100f);
            else
                iry = irye;
            if (Fvar["//no recoil quantity extra setting"] != 0 & Fvar["//no recoil step quantity"] != 0)
            {
                if (RightButtonSHOULDER_2)
                {
                    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["//LS2 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;
            }
        }
        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 outersize"] / 100f) / 1024f
+ (100f - Fvar["//anti-tearing outersize"]) / 100f);
        if (Fvar["//anti-tearing outer size"] < 0)
            mousexi = mousexbn * (((mousexbn > 0 ? mousexbn : -mousexbn) * -Fvar["//anti-tearing outersize"] / 100f) /
1024f + (100f + Fvar["//anti-tearing outersize"]) / 100f);
        if (Fvar["//anti-tearing outer size"] == 0)
            mousexi = mousexbn;
        if (!(Fbool["//RSR RSL 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 outersize"] / 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 outersize"] / 100f) /
1024f + (100f + Fvar["//anti-tearing outersize"]) / 100f);
            if (Fvar["//anti-tearing outer size"] == 0)
                mouseyi = mouseybn;
        }
        else
            mouseyi = irypp;
        mousexpn = (double)(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 = (double)(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 = (double)(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 = (double)(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"]))
            {
                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 ((!((EulerAnglesRight.X <= 0.5f & EulerAnglesRight.X >= -0.5f) | (EulerAnglesRight.Z <= 0.5f & EulerAnglesRight.Z
>= -0.5f)) & !Fbool["//swap"]) | (!((EulerAnglesLeft.X <= 0.5f & EulerAnglesLeft.X >= -0.5f) | (EulerAnglesLeft.Z <= 0.5f &
EulerAnglesLeft.Z >= -0.5f)) & Fbool["//swap"]))
            {
                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 Joycons_thrK()
    {
        for (; ; )
        {
            if (runningoff)
                return;
            watchK2 = (double)diffK.ElapsedTicks / (Stopwatch.Frequency / (1000L * 1000L));
            watchK = (watchK2 - watchK1) / 1000f;
            watchK1 = watchK2;
            if (Getstate)
                Joycons();
            else
            {
                signchangewheelZ = false;
                SelectOptions();
            }
            Thread.Sleep(5);
        }
    }
    private void Joycons_thrM()
    {
        for (; ; )
        {
            if (runningoff)
                return;
            watchM2 = (double)diffM.ElapsedTicks / (Stopwatch.Frequency / (1000L * 1000L));
            watchM = (watchM2 - watchM1) / 1000f;
            watchM1 = watchM2;
            if (Getstate)
                JoyconsIR();
            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; }
}
private void Joycon_thrDLeft()
{
  for (; ; )
  {
    if (runningoff)
      return;
    try
    {
      ReceiveRawLeft();
    }
    catch { }
  }
}
private void Joycon_thrDRight()
{
  for (; ; )
  {
    if (runningoff)
      return;
    try
    {
      ReceiveRawRight();
    }
    catch { }
  }
}
private void button4_Click(object sender, EventArgs e)
{
  this.Close();
}
private void button5_Click(object sender, EventArgs e)
{
  this.WindowState = FormWindowState.Minimized;
}
private void button3_Click(object sender, EventArgs e)
{
    const string message = "• Start the program after pressing sync buttons of Joycons for pairing it (press sync button of right Joycon first if you want both Joycons), and also with administrative privilege.\n\r• Adapt the mouse sensitivities and DPI in game options.\n\r• Press F1, F2, ..., F12, Shift+F1, Shift+F2, ..., or Shift+F12 for enable features and options.\n\r• Press Shift+Control+Capslock to lock change of features and options.\n\r• Check LS2 press I/O and LS2 view on for enable LS2 view I/O.\n\r• Check wheel view and RSR and RSL view for control view with gyroscope and RSR and RSL buttons.\n\r• Check stick view and RSR and RSL view for control view with stick and RSR and RSL buttons.\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 LS2 aim plus if you check LS2 accuracy for remove deadzone progressively.\n\r• Check swap for use left Joycon instead of right Joycon for cancel reload, LS2 RS2 switch, Joycon to front, shoulder 2, normal view, stick view and wheel view.\n\r• Change hardness quantity or zoning hardness quantity only for metro or xaim or fake.\n\r• Press decimal key to unlock controls.\n\r• Press at same time both shoulder buttons of enabled Joycon for center mouse.\n\r• Change controls and options with click on checkboxes.\n\r• Save in a file for enable changes.";
    const string caption = "JoyconsTheory Legend";
    MessageBox.Show(message, caption, MessageBoxButtons.OK, MessageBoxIcon.Information);
}
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["//LS2 press I/O"] = bool.Parse(file.ReadLine());
        file.ReadLine();
        Abool["//LS2 accuracy"] = bool.Parse(file.ReadLine());
        file.ReadLine();
        Abool["//wheel script"] = bool.Parse(file.ReadLine());
        file.ReadLine();
        Abool["//LS2 view on"] = bool.Parse(file.ReadLine());
        file.ReadLine();
        Abool["//LS2 aim plus"] = 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["//RSR RSL view"] = 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["//push r 1'"] = bool.Parse(file.ReadLine());
        file.ReadLine();
        Abool["//cancel reload x"] = bool.Parse(file.ReadLine());
        file.ReadLine();
        Abool["//LS2 RS2 switch"] = bool.Parse(file.ReadLine());
        file.ReadLine();
```

```
Abool["//LS 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["//swap"] = bool.Parse(file.ReadLine());
file.ReadLine();
action["//cancel reload x"] = file.ReadLine();
file.ReadLine();
action["//joycon left stick"] = file.ReadLine();
file.ReadLine();
action["//joycon left up"] = file.ReadLine();
file.ReadLine();
action["//joycon left down"] = file.ReadLine();
file.ReadLine();
action["//joycon right S2"] = file.ReadLine();
file.ReadLine();
action["//joycon left S2"] = file.ReadLine();
file.ReadLine();
action["//joycon plus"] = file.ReadLine();
file.ReadLine();
action["//joycon minus"] = file.ReadLine();
file.ReadLine();
action["//joycon right stick"] = file.ReadLine();
file.ReadLine();
action["//joycon right home"] = file.ReadLine();
file.ReadLine();
action["//joycon right S1"] = file.ReadLine();
file.ReadLine();
action["//joycon left left"] = file.ReadLine();
file.ReadLine();
action["//joycon left right"] = file.ReadLine();
file.ReadLine();
action["//joycon left SL"] = file.ReadLine();
file.ReadLine();
action["//joycon right SR"] = file.ReadLine();
file.ReadLine();
action["//joycon right SL"] = file.ReadLine();
file.ReadLine();
action["//joycon left SR"] = file.ReadLine();
file.ReadLine();
action["//joycon right to front"] = file.ReadLine();
file.ReadLine();
action["//joycon left to front"] = file.ReadLine();
file.ReadLine();
action["//joycon left stick up"] = file.ReadLine();
file.ReadLine();
action["//joycon left S1"] = file.ReadLine();
file.ReadLine();
action["//joycon left stick"] = file.ReadLine();
file.ReadLine();
action["//joycon right stick down"] = file.ReadLine();
file.ReadLine();
action["//joycon left stick down"] = file.ReadLine();
file.ReadLine();
action["//joycon right stick left"] = file.ReadLine();
file.ReadLine();
action["//joycon right stick right"] = file.ReadLine();
file.ReadLine();
action["//joycon right stick up"] = file.ReadLine();
file.ReadLine();
action["//joycon left stick left"] = file.ReadLine();
```

```
            file.ReadLine();
            action["//joycon left stick right"] = file.ReadLine();
            file.ReadLine();
            action["//joycon right up"] = file.ReadLine();
            file.ReadLine();
            action["//joycon right down"] = file.ReadLine();
            file.ReadLine();
            action["//joycon right left"] = file.ReadLine();
            file.ReadLine();
            action["//joycon right right"] = file.ReadLine();
            file.ReadLine();
            action["//joycon left capture"] = file.ReadLine();
            file.ReadLine();
            Fvar["//joycon right to front push r time extra setting"] = Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]",
"));
            file.ReadLine();
            Fvar["//cancel reload waiting LS2 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["//RS2 switch interval time extra setting"] = Convert.ToSingle(Regex.Replace(file.ReadLine(), "[^0-9-]", ""));
            file.ReadLine();
            Fvar["//RS2 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["//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.Close();
        cmBRTOFRONT.Text = action["//joycon right to front"];
        cmBRPLUS.Text = action["//joycon plus"];
        cmBLMINUS.Text =action["//joycon minus"];
        cmBRS1.Text =action["//joycon right S1"];
        cmBLSL.Text = action["//joycon left SL"];
        cmBRSR.Text = action["//joycon right SR"];
        cmBLSUP.Text = action["//joycon left stick up"];
        cmBLS.Text = action["//joycon left stick"];
        cmBLSLEFT.Text = action["//joycon left stick left"];
        cmBLSRIGHT.Text = action["//joycon leftstick right"];
        cmBLSDOWN.Text =action["//joycon left stick down"];
        cmBRS.Text = action["//joycon right stick"];
        cmBLTOFRONT.Text = action["//joycon left to front"];
        cmBRS2.Text =action["//joycon right S2"];
        cmBLS2.Text = action["//joycon left S2"];
        cmBLUP.Text = action["//joycon left up"];
        cmBLDOWN.Text = action["//joycon left down"];
        cmBLLEFT.Text =action["//joycon left left"];
        cmBLRIGHT.Text = action["//joycon left right"];
        cmBLS1.Text = action["//joycon left S1"];
        cmBLS.Text = action["//joycon left stick"];
        cmBRSRIGHT.Text = action["//joycon rightstick right"];
        cmBRSLEFT.Text =action["//joycon right stick left"];
        cmBRSLEFT.Text =action["//joycon right stick left"];
        cmBRSDOWN.Text = action["//joycon right stick down"];
        cmBRHOME.Text = action["//joycon right home"];
        cmBLSR.Text = action["//joycon left SR"];
        cmBRSL.Text = action["//joycon right SL"];
        cmBRLEFT.Text = action["//joycon right left"];
        cmBRRIGHT.Text = action["//joycon right right"];
        cmBRDOWN.Text =action["//joycon right down"];
        cmBRUP.Text = action["//joycon right up"];
        if (!Abool["//cancel reload x"])
           cmBRCANCELRELOAD.Text = " ";
        else
           cmBRCANCELRELOAD.Text = action["//cancel reload x"];
        cmBLCAPTURE.Text = action["//joycon left capture"];
        txtBjoyconrighttofrontpush.Text = Fvar["//joycon right to front push r time extra setting"].ToString();
        txtBcancelreload.Text = Fvar["//cancel reload waiting LS2 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();
        txtBRS2switchinterval.Text = Fvar["//RS2 switch interval time extra setting"].ToString();
        txtBRS2switchpressdelay.Text = Fvar["//RS2 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();
txtBzoningquantity.Text = Fvar["//zoning quantity"].ToString();
txtBzoninghardnessquantity.Text = Fvar["//zoning hardness quantity"].ToString();
txtBnorecoilstepquantity.Text = Fvar["//no recoil step quantity"].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["//LS2 press I/O"])
    chkBF12.Checked = true;
else
    chkBF12.Checked = false;
if (Fbool["//LS2 press I/O"] & !Abool["//LS2 press I/O"])
    Abool["//LS2 press I/O"] = true;
else
    if (Fbool["//LS2 press I/O"] & Abool["//LS2 press I/O"])
        Abool["//LS2 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"])
   chkBF10S.Checked = true;
else
   chkBF10S.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["//LS2 accuracy"])
   chkBF11.Checked = true;
else
   chkBF11.Checked = false;
if (Fbool["//LS2 accuracy"] & !Abool["//LS2 accuracy"])
   Abool["//LS2 accuracy"] = true;
else
   if (Fbool["//LS2accuracy"] & Abool["//LS2 accuracy"])
      Abool["//LS2 accuracy"] = false;
if (Abool["//LS2 view on"])
   chkBF1S.Checked = true;
else
   chkBF1S.Checked = false;
if (Fbool["//LS2 view on"] & !Abool["//LS2 view on"])
   Abool["//LS2 view on"] = true;
else
   if (Fbool["//LS2 view on"] & Abool["//LS2 view on"])
      Abool["//LS2 view on"] = false;
if (Abool["//LS2 aim plus"])
   chkBF2S.Checked = true;
else
   chkBF2S.Checked = false;
if (Fbool["//LS2 aim plus"] & !Abool["//LS2 aim plus"])
   Abool["//LS2 aim plus"] = true;
else
   if (Fbool["//LS2aim plus"] & Abool["//LS2 aim plus"])
      Abool["//LS2 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["//RSR RSL view"])
   chkBF5S.Checked = true;
else
   chkBF5S.Checked = false;
if (Fbool["//RSR RSL view"] & !Abool["//RSR RSL view"])
   Abool["//RSR RSL view"] = true;
else
   if (Fbool["//RSR RSL view"] & Abool["//RSR RSL view"])
      Abool["//RSR RSL view"] = 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"])
   chkBF9S.Checked = true;
else
   chkBF9S.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["//push r 1'"])
   chkBF8S.Checked = true;
else
   chkBF8S.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["//LS2 RS2 switch"])
   chkBF7S.Checked = true;
else
   chkBF7S.Checked = false;
if (Fbool["//LS2 RS2 switch"] & !Abool["//LS2 RS2 switch"])
   Abool["//LS2 RS2 switch"] = true;
else
   if (Fbool["//LS2 RS2 switch"] & Abool["//LS2 RS2 switch"])
      Abool["//LS2 RS2 switch"] = false;
if (Abool["//LS press I/O"])
   chkBF11S.Checked = true;
else
   chkBF11S.Checked = false;
```

```
        if (Fbool["//LS press I/O"] & !Abool["//LS press I/O"])
            Abool["//LS press I/O"] = true;
        else
            if (Fbool["//LS press I/O"] & Abool["//LS press I/O"])
                Abool["//LS 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"])
            chkBF12S.Checked = true;
        else
            chkBF12S.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["//swap"])
            chkBF6S.Checked = true;
        else
            chkBF6S.Checked = false;
        if (Fbool["//swap"] & !Abool["//swap"])
            Abool["//swap"] = true;
        else
            if (Fbool["//swap"] & Abool["//swap"])
                Abool["//swap"] = false;
        Assignating();
        clearList();
        setControlsAndOptions();
        reconfiguration = false;
    }
    catch
    {
        saveConfig("joyconsdefault.txt");
        savePathInitFile("joyconsdefault.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 (!chkBF9S.Checked)
            cmBRCANCELRELOAD.Text = " ";
        if (cmBRSUP.Text == cmBRSUP.Items[0].ToString() & cmBRSRIGHT.Text == cmBRSRIGHT.Items[0].ToString() &
cmBRSLEFT.Text == cmBRSLEFT.Items[0].ToString() & cmBRSDOWN.Text == cmBRSDOWN.Items[0].ToString() &
cmBLSUP.Text == cmBLSUP.Items[0].ToString() & cmBLSRIGHT.Text == cmBLSRIGHT.Items[0].ToString() & cmBLSLEFT.Text
== cmBLSLEFT.Items[0].ToString() & cmBLSDOWN.Text == cmBLSDOWN.Items[0].ToString() & cmBLRIGHT.Text ==
cmBLRIGHT.Items[0].ToString() & cmBLUP.Text == cmBLUP.Items[0].ToString() & cmBLDOWN.Text ==
cmBLDOWN.Items[0].ToString() & cmBRPLUS.Text == cmBRPLUS.Items[0].ToString() & cmBLLEFT.Text ==
cmBLLEFT.Items[0].ToString() & cmBLS2.Text == cmBLS2.Items[0].ToString() & cmBRHOME.Text ==
cmBRHOME.Items[0].ToString() & chkBF9S.Checked & cmBRSR.Text == cmBRSR.Items[0].ToString() & cmBRTOFRONT.Text
== cmBRTOFRONT.Items[0].ToString() & cmBRS.Text == cmBRS.Items[0].ToString() & cmBRS2.Text ==
cmBRS2.Items[0].ToString() & cmBLMINUS.Text == cmBLMINUS.Items[0].ToString() & cmBLSL.Text ==
cmBLSL.Items[0].ToString() & cmBRS1.Text == cmBRS1.Items[0].ToString() & cmBLSUP.Text == cmBLSUP.Items[0].ToString()
& cmBRSLEFT.Text == cmBRSLEFT.Items[0].ToString() & cmBRSRIGHT.Text == cmBRSRIGHT.Items[0].ToString() &
cmBRSDOWN.Text == cmBRSDOWN.Items[0].ToString() & cmBLS1.Text == cmBLS1.Items[0].ToString() & cmBLS.Text ==
cmBLS.Items[0].ToString() & cmBLSUP.Text == cmBLSUP.Items[0].ToString() & cmBLS.Text == cmBLS.Items[0].ToString() &
cmBLSDOWN.Text == cmBLSDOWN.Items[0].ToString() & cmBLSDOWN.Text == cmBLSDOWN.Items[0].ToString() &
cmBLTOFRONT.Text == cmBLTOFRONT.Items[0].ToString() & cmBLSLEFT.Text == cmBLSLEFT.Items[0].ToString() &
cmBLSR.Text == cmBLSR.Items[0].ToString() & cmBRSL.Text == cmBRSL.Items[0].ToString() & cmBLSRIGHT.Text ==
cmBLSRIGHT.Items[0].ToString() & ((cmBRCANCELRELOAD.Text == cmBRCANCELRELOAD.Items[0].ToString() &
chkBF9S.Checked) | (cmBRCANCELRELOAD.Text == " " & !chkBF9S.Checked)))
            Fbool["//rebind keys"] = false;
        else
            Fbool["//rebind keys"] = true;
        action["//joycon right to front"] = cmBRTOFRONT.Text;
        action["//joycon plus"] = cmBRPLUS.Text;
        action["//joycon minus"] = cmBLMINUS.Text;
        action["//joycon right S1"] = cmBRS1.Text;
        action["//joycon left SL"] = cmBLSL.Text;
        action["//joycon right SR"] = cmBRSR.Text;
        action["//joycon left stick up"] = cmBLSUP.Text;
        action["//joycon left stick"] = cmBLS.Text;
        action["//joycon left stick left"] = cmBLSLEFT.Text;
        action["//joycon left stick right"] = cmBLSRIGHT.Text;
        action["//joycon left stick down"] = cmBLSDOWN.Text;
        action["//joycon right stick"] = cmBRS.Text;
        action["//joycon left to front"] = cmBLTOFRONT.Text;
        action["//joycon right S2"] = cmBRS2.Text;
        action["//joycon left S2"] = cmBLS2.Text;
        action["//joycon left up"] = cmBLUP.Text;
        action["//joycon left down"] = cmBLDOWN.Text;
        action["//joycon left left"] = cmBLLEFT.Text;
        action["//joycon left right"] = cmBLRIGHT.Text;
        action["//joycon left S1"] = cmBLS1.Text;
        action["//joycon left stick"] = cmBLS.Text;
        action["//joycon right stick right"] = cmBRSRIGHT.Text;
        action["//joycon right stick left"] = cmBRSLEFT.Text;
        action["//joycon right stick left"] = cmBRSLEFT.Text;
        action["//joycon right stick down"] = cmBRSDOWN.Text;
        action["//joycon right home"] = cmBRHOME.Text;
        action["//joycon left SR"] = cmBLSR.Text;
        action["//joycon right SL"] = cmBRSL.Text;
        action["//joycon right left"] = cmBRLEFT.Text;
        action["//joycon right right"] = cmBRRIGHT.Text;
        action["//joycon right down"] = cmBRDOWN.Text;
        action["//joycon right up"] = cmBRUP.Text;
        action["//cancel reload x"] = cmBRCANCELRELOAD.Text;
        action["//joycon left capture"] = cmBLCAPTURE.Text;
        Fvar["//joycon right to front push r time extra setting"] =
Convert.ToInt32(Regex.Replace(txtBjoyconrighttofrontpush.Text, "[^0-9-]", ""));
        Fvar["//cancel reload waiting LS2 time extra setting"] = Convert.ToInt32(Regex.Replace(txtBcancelreload.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["//RS2 switch interval time extra setting"] = Convert.ToInt32(Regex.Replace(txtBRS2switchinterval.Text, "[^0-9-
]", ""));
        Fvar["//RS2 switch press delay time extra setting"] = Convert.ToInt32(Regex.Replace(txtBRS2switchpressdelay.Text,
"[^0-9-]", ""));
        Fvar["//tick time"] = Convert.ToInt32(Regex.Replace(txtBticktime.Text, "[^0-9-]", ""));
        Fvar["//wheel scripts tick limit in"] = Convert.ToInt32(Regex.Replace(txtBwheelscriptsticklimitin.Text, "[^0-9-]", ""));
        Fvar["//wheel scripts tick limit out"] = Convert.ToInt32(Regex.Replace(txtBwheelscriptsticklimitout.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-]", ""));
        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("//LS2 press I/O");
        file.WriteLine(chkBF12.Checked);
        file.WriteLine("//LS2 accuracy");
        file.WriteLine(chkBF11.Checked);
        file.WriteLine("//wheel script");
        file.WriteLine(chkBF10S.Checked);
        file.WriteLine("//LS2 view on");
        file.WriteLine(chkBF1S.Checked);
        file.WriteLine("//LS2 aim plus");
        file.WriteLine(chkBF2S.Checked);
        file.WriteLine("//stick view");
        file.WriteLine(chkBF3S.Checked);
        file.WriteLine("//wheel view");
```

```
file.WriteLine(chkBF4S.Checked);
file.WriteLine("//RSR RSL view");
file.WriteLine(chkBF5S.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("//push r 1'");
file.WriteLine(chkBF8S.Checked);
file.WriteLine("//cancel reload x");
file.WriteLine(chkBF9S.Checked);
file.WriteLine("//LS2 RS2 switch");
file.WriteLine(chkBF7S.Checked);
file.WriteLine("//LS press I/O");
file.WriteLine(chkBF11S.Checked);
file.WriteLine("//driver mouse");
file.WriteLine(chkBF10.Checked);
file.WriteLine("//driver keyboard");
file.WriteLine(chkBF12S.Checked);
file.WriteLine("//swap");
file.WriteLine(chkBF6S.Checked);
file.WriteLine("//cancel reload x");
file.WriteLine(action["//cancel reload x"]);
file.WriteLine("//joycon left stick");
file.WriteLine(action["//joycon left stick"]);
file.WriteLine("//joycon left up");
file.WriteLine(action["//joycon left up"]);
file.WriteLine("//joycon left down");
file.WriteLine(action["//joycon left down"]);
file.WriteLine("//joycon right S2");
file.WriteLine(action["//joycon right S2"]);
file.WriteLine("//joycon left S2");
file.WriteLine(action["//joycon left S2"]);
file.WriteLine("//joycon plus");
file.WriteLine(action["//joycon plus"]);
file.WriteLine("//joycon minus");
file.WriteLine(action["//joycon minus"]);
file.WriteLine("//joycon right stick");
file.WriteLine(action["//joycon right stick"]);
file.WriteLine("//joycon right home");
file.WriteLine(action["//joycon right home"]);
file.WriteLine("//joycon right S1");
file.WriteLine(action["//joycon right S1"]);
file.WriteLine("//joycon left left");
file.WriteLine(action["//joycon left left"]);
file.WriteLine("//joycon left right");
file.WriteLine(action["//joycon left right"]);
file.WriteLine("//joycon left SL");
file.WriteLine(action["//joycon left SL"]);
file.WriteLine("//joycon right SR");
file.WriteLine(action["//joycon right SR"]);
file.WriteLine("//joycon right SL");
file.WriteLine(action["//joycon right SL"]);
file.WriteLine("//joycon left SR");
file.WriteLine(action["//joycon left SR"]);
file.WriteLine("//joycon right to front");
file.WriteLine(action["//joycon right to front"]);
file.WriteLine("//joycon left to front");
file.WriteLine(action["//joycon left to front"]);
file.WriteLine("//joycon left stick up");
file.WriteLine(action["//joycon left stick up"]);
file.WriteLine("//joycon left S1");
file.WriteLine(action["//joycon left S1"]);
```

```
file.WriteLine("//joycon left stick");
file.WriteLine(action["//joycon left stick"]);
file.WriteLine("//joycon right stick down");
file.WriteLine(action["//joycon right stick down"]);
file.WriteLine("//joycon left stick down");
file.WriteLine(action["//joycon left stick down"]);
file.WriteLine("//joycon right stick left");
file.WriteLine(action["//joycon right stick left"]);
file.WriteLine("//joycon right stick right");
file.WriteLine(action["//joycon right stick right"]);
file.WriteLine("//joycon right stick up");
file.WriteLine(action["//joycon right stick up"]);
file.WriteLine("//joycon left stick left");
file.WriteLine(action["//joycon left stick left"]);
file.WriteLine("//joycon left stick right");
file.WriteLine(action["//joycon left stick right"]);
file.WriteLine("//joycon right up");
file.WriteLine(action["//joycon right up"]);
file.WriteLine("//joycon right down");
file.WriteLine(action["//joycon right down"]);
file.WriteLine("//joycon right left");
file.WriteLine(action["//joycon right left"]);
file.WriteLine("//joycon right right");
file.WriteLine(action["//joycon right right"]);
file.WriteLine("//joycon left capture");
file.WriteLine(action["//joycon left capture"]);
file.WriteLine("//joycon right to front push r time extra setting");
file.WriteLine((Convert.ToInt32(Fvar["//joycon right to front push r time extra setting"])).ToString());
file.WriteLine("//cancel reload waiting LS2 time extra setting");
file.WriteLine((Convert.ToInt32(Fvar["//cancel reload waiting LS2 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("//RS2 switch interval time extra setting");
file.WriteLine((Convert.ToInt32(Fvar["//RS2 switch interval time extra setting"])).ToString());
file.WriteLine("//RS2 switch press delay time extra setting");
file.WriteLine((Convert.ToInt32(Fvar["//RS2 switch press delay time extra setting"])).ToString());
file.WriteLine("//tick time");
```

```
        file.WriteLine((Convert.ToInt32(Fvar["//tick time"])).ToString());
        file.WriteLine("//wheel scriptstick limit in");
        file.WriteLine((Convert.ToInt32(Fvar["//wheel scriptstick limit in"])).ToString());
        file.WriteLine("//wheel scriptstick limit out");
        file.WriteLine((Convert.ToInt32(Fvar["//wheel scriptstick 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("//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["//LS2 press I/O"] & !chkBF12.Checked) | (!Fbool["//LS2 press I/O"] & chkBF12.Checked))
          Abool["//LS2 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"] & !chkBF10S.Checked) | (!Fbool["//wheel script"] & chkBF10S.Checked))
          Abool["//wheel script"] = true;
        if ((Fbool["//LS2 accuracy"] & !chkBF11.Checked) | (!Fbool["//LS2 accuracy"] & chkBF11.Checked))
          Abool["//LS2 accuracy"] = true;
        if ((Fbool["//LS2 view on"] & !chkBF1S.Checked) | (!Fbool["//LS2 view on"] & chkBF1S.Checked))
          Abool["//LS2 view on"] = true;
        if ((Fbool["//LS2 aim plus"] & !chkBF2S.Checked) | (!Fbool["//LS2 aim plus"] & chkBF2S.Checked))
          Abool["//LS2 aim plus"] = true;
        if ((Fbool["//xaim"] & !chkBF9.Checked) | (!Fbool["//xaim"] & chkBF9.Checked))
          Abool["//xaim"] = true;
        if ((Fbool["//RSR RSL view"] & !chkBF5S.Checked) | (!Fbool["//RSR RSL view"] & chkBF5S.Checked))
          Abool["//RSR RSL view"] = 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"] & !chkBF9S.Checked) | (!Fbool["//cancel reload x"] & chkBF9S.Checked))
          Abool["//cancel reload x"] = true;
        if ((Fbool["//push r 1'"] & !chkBF8S.Checked) | (!Fbool["//push r 1'"] & chkBF8S.Checked))
          Abool["//push r 1'"] = true;
        if ((Fbool["//LS2 RS2 switch"] & !chkBF7S.Checked) | (!Fbool["//LS2 RS2 switch"] & chkBF7S.Checked))
          Abool["//LS2 RS2 switch"] = true;
        if ((Fbool["//LS press I/O"] & !chkBF11S.Checked) | (!Fbool["//LS press I/O"] & chkBF11S.Checked))
          Abool["//LS press I/O"] = true;
        if ((Fbool["//driver mouse"] & !chkBF10.Checked) | (!Fbool["//driver mouse"] & chkBF10.Checked))
          Abool["//driver mouse"] = true;
        if ((Fbool["//driver keyboard"] & !chkBF12S.Checked) | (!Fbool["//driver keyboard"] & chkBF12S.Checked))
```

```
        Abool["//driver keyboard"] = true;
    if ((Fbool["//swap"] & !chkBF6S.Checked) | (!Fbool["//swap"] & chkBF6S.Checked))
        Abool["//swap"] = true;
    Assignating();
    clearList();
    setControlsAndOptions();
    reconfiguration = false;
}
public void savePathInitFile(string pathtoinitfile)
{
    System.IO.StreamWriter initfile = new System.IO.StreamWriter("joyconsinitfile.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();
}
private void FormClose()
{
    disconnect();
}
private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
    notpressing1and2 = true;
    runningoff = true;
    TimeEndPeriod(1);
    Thread.Sleep(100);
    switchwheelfix();
    try
    {
        hid_close(handleRight);
    }
    catch { }
    try
    {
        hid_close(handleLeft);
    }
    catch { }
    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);
}
private const string vendor_id = "57e", vendor_id_ = "057e", product_l = "2006", product_r = "2007";
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.ByValTStr, SizeConst
= 256)]
      public string DevicePath;
    }
    private int leftandright;
    private static double[] stickLeft = { 0, 0 };
    private static SafeFileHandle handleLeft;
    private static byte[] stick_rawLeft = { 0, 0, 0 };
    private static UInt16[] stick_calibrationLeft = { 0, 0 };
    private static UInt16[] stick_precalLeft = { 0, 0 };
    private static Vector3 gyr_gLeft = new Vector3();
    private static Vector3 acc_gLeft = new Vector3();
    private static Vector3 gyr_rLeft = new Vector3();
    private static Vector3 acc_rLeft = new Vector3();
    private const uint report_lenLeft = 49;
    public static Vector3 i_aLeft = new Vector3(1, 0, 0);
    public static Vector3 j_aLeft = new Vector3(0, 1, 0);
    public static Vector3 k_aLeft = new Vector3(0, 0, 1);
    public static Vector3 i_bLeft = new Vector3(1, 0, 0);
    public static Vector3 j_bLeft = new Vector3(0, 1, 0);
    public static Vector3 k_bLeft = new Vector3(0, 0, 1);
    public static Vector3 i_cLeft = new Vector3(1, 0, 0);
    public static Vector3 j_cLeft = new Vector3(0, 1, 0);
    public static Vector3 k_cLeft = new Vector3(0, 0, 1);
    private static Vector3 InitDirectAnglesLeft, DirectAnglesLeft;
    private static Vector3 InitEulerAnglesaLeft, EulerAnglesaLeft, InitEulerAnglesbLeft, EulerAnglesbLeft,
InitEulerAnglescLeft, EulerAnglescLeft, EulerAnglesLeft;
    private static bool LeftButtonSHOULDER_1, LeftButtonSHOULDER_2, LeftButtonSR, LeftButtonSL,
LeftButtonDPAD_DOWN, LeftButtonDPAD_RIGHT, LeftButtonDPAD_UP, LeftButtonDPAD_LEFT, LeftButtonMINUS,
LeftButtonSTICK, LeftButtonCAPTURE;
    private static byte[] report_bufLeft, report_bufaLeft = new byte[report_lenLeft];
    private static double[] arrayLeft;
    private static byte[] buf_Left = new byte[report_lenLeft];
    public static uint hDevInfoLeft;
    public static double indexLeft = 0;
    private static float roundLeft = 16000f;
    public static float acc_gcalibrationLeftX, acc_gcalibrationLeftY, acc_gcalibrationLeftZ, gyr_gcalibrationLeftX,
gyr_gcalibrationLeftY, gyr_gcalibrationLeftZ;
    private bool ScanLeft()
    {
      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_l))
                {
                    ISLEFT = true;
                    AttachJoyLeft(diDetail.DevicePath);
                    AttachJoyLeft(diDetail.DevicePath);
                    AttachJoyLeft(diDetail.DevicePath);
                    return true;
                }
            }
            index++;
        }
        return false;
    }
    public double[] GetStickLeft()
    {
        return stickLeft;
    }
    public Vector3 GetAccelLeft()
    {
        if (!Fbool["//swap"])
            return acc_gLeft;
        else
            return acc_gRight;
    }
    public Quaternion GetVectoraLeft()
    {
        Vector3 v1 = new Vector3(j_aLeft.X, i_aLeft.X, k_aLeft.X);
        Vector3 v2 = -(new Vector3(j_aLeft.Z, i_aLeft.Z, k_aLeft.Z));
        return QuaternionLookRotationLeft(v1, v2);
    }
    public Quaternion GetVectorbLeft()
    {
        Vector3 v1 = new Vector3(j_bLeft.X, i_bLeft.X, k_bLeft.X);
        Vector3 v2 = -(new Vector3(j_bLeft.Z, i_bLeft.Z, k_bLeft.Z));
        return QuaternionLookRotationLeft(v1, v2);
    }
    public Quaternion GetVectorcLeft()
    {
        Vector3 v1 = new Vector3(j_cLeft.X, i_cLeft.X, k_cLeft.X);
        Vector3 v2 = -(new Vector3(j_cLeft.Z, i_cLeft.Z, k_cLeft.Z));
        return QuaternionLookRotationLeft(v1, v2);
    }
    private static Quaternion QuaternionLookRotationLeft(Vector3 forward, Vector3 up)
    {
        Vector3 vector = Vector3.Normalize(forward);
        Vector3 vector2 = Vector3.Normalize(Vector3.Cross(up, vector));
        Vector3 vector3 = Vector3.Cross(vector, vector2);
        var m00 = vector2.X;
        var m01 = vector2.Y;
        var m02 = vector2.Z;
        var m10 = vector3.X;
        var m11 = vector3.Y;
        var m12 = vector3.Z;
        var m20 = vector.X;
        var m21 = vector.Y;
        var m22 = vector.Z;
        double num8 = (m00 + m11) + m22;
        var quaternion = new Quaternion();
        if (num8 > 0f)
        {
            var num = (double)Math.Sqrt(num8 + 1f);
```

```
            quaternion.W = (float)num * 0.5f;
            num = 0.5f / num;
            quaternion.X = (float)(m12 - m21) * (float)num;
            quaternion.Y = (float)(m20 - m02) * (float)num;
            quaternion.Z = (float)(m01 - m10) * (float)num;
            return quaternion;
        }
        if ((m00 >= m11) && (m00 >= m22))
        {
            var num7 = (double)Math.Sqrt(((1f + m00) - m11) - m22);
            var num4 = 0.5f / num7;
            quaternion.X = 0.5f * (float)num7;
            quaternion.Y = (float)(m01 + m10) * (float)num4;
            quaternion.Z = (float)(m02 + m20) * (float)num4;
            quaternion.W = (float)(m12 - m21) * (float)num4;
            return quaternion;
        }
        if (m11 > m22)
        {
            var num6 = (double)Math.Sqrt(((1f + m11) - m00) - m22);
            var num3 = 0.5f / num6;
            quaternion.X = (float)(m10 + m01) * (float)num3;
            quaternion.Y = 0.5f * (float)num6;
            quaternion.Z = (float)(m21 + m12) * (float)num3;
            quaternion.W = (float)(m20 - m02) * (float)num3;
            return quaternion;
        }
        var num5 = (double)Math.Sqrt(((1f + m22) - m00) - m11);
        var num2 = 0.5f / num5;
        quaternion.X = (float)(m20 + m02) * (float)num2;
        quaternion.Y = (float)(m21 + m12) * (float)num2;
        quaternion.Z = 0.5f * (float)num5;
        quaternion.W = (float)(m01 - m10) * (float)num2;
        return quaternion;
    }
    public static Vector3 ToEulerAnglesLeft(Quaternion q)
    {
        Vector3 pitchYawRoll = new Vector3();
        double sqw = q.W * q.W;
        double sqx = q.X * q.X;
        double sqy = q.Y * q.Y;
        double sqz = q.Z * q.Z;
        double unit = sqx + sqy + sqz + sqw;
        double test = q.X * q.Y + q.Z * q.W;
        if (test > 0.4999f * unit)                // 0.4999f OR 0.5f - EPSILON
        {
            pitchYawRoll.Y = 2f * (float)Math.Atan2(q.X, q.W);  // Yaw
            pitchYawRoll.X = (float)Math.PI * 0.5f;             // Pitch
            pitchYawRoll.Z = 0f;                  // Roll
            return pitchYawRoll;
        }
        else if (test < -0.4999f * unit)          // -0.4999f OR -0.5f + EPSILON
        {
            pitchYawRoll.Y = -2f * (float)Math.Atan2(q.X, q.W); // Yaw
            pitchYawRoll.X = -(float)Math.PI * 0.5f;            // Pitch
            pitchYawRoll.Z = 0f;                  // Roll
            return pitchYawRoll;
        }
        else
        {
            pitchYawRoll.Y = (float)Math.Atan2(2f * q.Y * q.W - 2f * q.X * q.Z, sqx - sqy - sqz + sqw);      // Yaw
            pitchYawRoll.X = (float)Math.Asin(2f * test / unit);                                             // Pitch
            pitchYawRoll.Z = (float)Math.Atan2(2f * q.X * q.W - 2f * q.Y * q.Z, -sqx + sqy - sqz + sqw);     // Roll
```

```
        }
        return pitchYawRoll;
    }
    public void AttachJoyLeft(string path)
    {
        do
        {
            IntPtr handle = CreateFile(path, System.IO.FileAccess.ReadWrite, System.IO.FileShare.ReadWrite, new
System.IntPtr(), System.IO.FileMode.Open, EFileAttributes.Normal, new System.IntPtr());
            handleLeft = hid_open_path(handle);
            SubcommandLeft(0x3, new byte[] { 0x30 }, 1);
            SubcommandLeft(0x40, new byte[] { 0x1 }, 1);
        }
        while (handleLeft.IsInvalid);
    }
    private void ReceiveRawLeft()
    {
        Lhid_read_timeout(handleLeft, report_bufaLeft, (UIntPtr)49);
    }
    private void ProcessButtonsAndStickLeft()
    {
        LeftButtonSHOULDER_1 = (report_bufLeft[3 + (ISLEFT ? 2 : 0)] & 0x40) != 0;
        if (!Fbool["//swap"])
            LeftButtonSHOULDER_2 = (report_bufLeft[3 + (ISLEFT ? 2 : 0)] & 0x80) != 0;
        else
            RightButtonSHOULDER_2 = (report_bufLeft[3 + (ISLEFT ? 2 : 0)] & 0x80) != 0;
        LeftButtonSR = (report_bufLeft[3 + (ISLEFT ? 2 : 0)] & 0x10) != 0;
        LeftButtonSL = (report_bufLeft[3 + (ISLEFT ? 2 : 0)] & 0x20) != 0;
        LeftButtonDPAD_DOWN = (report_bufLeft[3 + (ISLEFT ? 2 : 0)] & (ISLEFT ? 0x01 : 0x04)) != 0;
        LeftButtonDPAD_RIGHT = (report_bufLeft[3 + (ISLEFT ? 2 : 0)] & (ISLEFT ? 0x04 : 0x08)) != 0;
        LeftButtonDPAD_UP = (report_bufLeft[3 + (ISLEFT ? 2 : 0)] & (ISLEFT ? 0x02 : 0x02)) != 0;
        LeftButtonDPAD_LEFT = (report_bufLeft[3 + (ISLEFT ? 2 : 0)] & (ISLEFT ? 0x08 : 0x01)) != 0;
        LeftButtonMINUS = ((report_bufLeft[4] & 0x01) != 0);
        LeftButtonCAPTURE = ((report_bufLeft[4] & 0x20) != 0);
        LeftButtonSTICK = ((report_bufLeft[4] & (ISLEFT ? 0x08 : 0x04)) != 0);
        stick_rawLeft[0] = report_bufLeft[6 + (ISLEFT ? 0 : 3)];
        stick_rawLeft[1] = report_bufLeft[7 + (ISLEFT ? 0 : 3)];
        stick_rawLeft[2] = report_bufLeft[8 + (ISLEFT ? 0 : 3)];
        stick_precalLeft[0] = (UInt16)(stick_rawLeft[0] | ((stick_rawLeft[1] & 0xf) << 8));
        stick_precalLeft[1] = (UInt16)((stick_rawLeft[1] >> 4) | (stick_rawLeft[2] << 4));
        stickLeft = CenterSticksLeft(stick_precalLeft);
    }
    private void ExtractIMUValuesLeft()
    {
        acc_gLeft.X = (int)(averageLeft((Int16)(report_bufLeft[13 + 0 * 12] | ((report_bufLeft[14 + 0 * 12] << 8) & 0xff00)),
(Int16)(report_bufLeft[13 + 1 * 12] | ((report_bufLeft[14 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufLeft[13 + 2 * 12] |
((report_bufLeft[14 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f) - acc_gcalibrationLeftX;
        acc_gLeft.Y = (int)(averageLeft((Int16)(report_bufLeft[15 + 0 * 12] | ((report_bufLeft[16 + 0 * 12] << 8) & 0xff00)),
(Int16)(report_bufLeft[15 + 1 * 12] | ((report_bufLeft[16 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufLeft[15 + 2 * 12] |
((report_bufLeft[16 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f) - acc_gcalibrationLeftY;
        acc_gLeft.Z = (int)(averageLeft((Int16)(report_bufLeft[17 + 0 * 12] | ((report_bufLeft[18 + 0 * 12] << 8) & 0xff00)),
(Int16)(report_bufLeft[17 + 1 * 12] | ((report_bufLeft[18 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufLeft[17 + 2 * 12] |
((report_bufLeft[18 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f) - acc_gcalibrationLeftZ;
        gyr_gLeft.X = (int)(averageLeft((int)((Int16)((report_bufLeft[19 + 0 * 12] | ((report_bufLeft[20 + 0 * 12] << 8) &
0xff00)))), (int)((Int16)((report_bufLeft[19 + 1 * 12] | ((report_bufLeft[20 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufLeft[19 + 2 * 12] | ((report_bufLeft[20 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f) -
gyr_gcalibrationLeftX;
        gyr_gLeft.Y = (int)(averageLeft((int)((Int16)((report_bufLeft[21 + 0 * 12] | ((report_bufLeft[22 + 0 * 12] << 8) &
0xff00)))), (int)((Int16)((report_bufLeft[21 + 1 * 12] | ((report_bufLeft[22 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufLeft[21 + 2 * 12] | ((report_bufLeft[22 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f) -
gyr_gcalibrationLeftY;
        gyr_gLeft.Z = (int)(averageLeft((int)((Int16)((report_bufLeft[23 + 0 * 12] | ((report_bufLeft[24 + 0 * 12] << 8) &
0xff00)))), (int)((Int16)((report_bufLeft[23 + 1 * 12] | ((report_bufLeft[24 + 1 * 12] << 8) & 0xff00)))),
```

```
(int)((Int16)((report_bufLeft[23 + 2 * 12] | ((report_bufLeft[24 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f) -
gyr_gcalibrationLeftZ;
        acc_rLeft.X = ((int)(acc_gLeft.X * roundLeft)) / roundLeft;
        acc_rLeft.Y = ((int)(acc_gLeft.Y * roundLeft)) / roundLeft;
        acc_rLeft.Z = ((int)(acc_gLeft.Z * roundLeft)) / roundLeft;
        gyr_rLeft.X = ((int)(gyr_gLeft.X * roundLeft)) / roundLeft;
        gyr_rLeft.Y = ((int)(gyr_gLeft.Y * roundLeft)) / roundLeft;
        gyr_rLeft.Z = ((int)(gyr_gLeft.Z * roundLeft)) / roundLeft;
        if (!Getstate)
          InitDirectAnglesLeft =acc_rLeft;
        DirectAnglesLeft =acc_rLeft - InitDirectAnglesLeft;
        if (((int)(i_cLeft.X * roundLeft)) / roundLeft == 1f | (((int)(i_cLeft.Y * roundLeft)) / roundLeft == 0f & ((int)(i_cLeft.Z *
roundLeft)) / roundLeft == 0f))
          i_cLeft = new Vector3(1, 0, 0);
        if (((int)(j_cLeft.Y * roundLeft)) / roundLeft == 1f | (((int)(j_cLeft.X * roundLeft)) / roundLeft == 0f & ((int)(j_cLeft.Z *
roundLeft)) / roundLeft == 0f))
          j_cLeft = new Vector3(0, 1, 0);
        if (((int)(k_cLeft.Z * roundLeft)) / roundLeft == 1f | (((int)(k_cLeft.X * roundLeft)) / roundLeft == 0f & ((int)(k_cLeft.Y *
roundLeft)) / roundLeft == 0f))
          k_cLeft = new Vector3(0, 0, 1);
        if (((int)(i_bLeft.X * roundLeft)) / roundLeft == 1f | (((int)(i_bLeft.Y * roundLeft)) / roundLeft == 0f & ((int)(i_bLeft.Z *
roundLeft)) / roundLeft == 0f))
          i_bLeft = new Vector3(1, 0, 0);
        if (((int)(j_bLeft.Y * roundLeft)) / roundLeft == 1f | (((int)(j_bLeft.X * roundLeft)) / roundLeft == 0f & ((int)(j_bLeft.Z *
roundLeft)) / roundLeft == 0f))
          j_bLeft = new Vector3(0, 1, 0);
        if (((int)(k_bLeft.Z * roundLeft)) / roundLeft == 1f | (((int)(k_bLeft.X * roundLeft)) / roundLeft == 0f & ((int)(k_bLeft.Y
* roundLeft)) / roundLeft == 0f))
          k_bLeft = new Vector3(0, 0, 1);
        if (((int)(i_aLeft.X * roundLeft)) / roundLeft == 1f | (((int)(i_aLeft.Y * roundLeft)) / roundLeft == 0f & ((int)(i_aLeft.Z *
roundLeft)) / roundLeft == 0f))
          i_aLeft = new Vector3(1, 0, 0);
        if (((int)(j_aLeft.Y * roundLeft)) / roundLeft == 1f | (((int)(j_aLeft.X * roundLeft)) / roundLeft == 0f & ((int)(j_aLeft.Z *
roundLeft)) / roundLeft == 0f))
          j_aLeft = new Vector3(0, 1, 0);
        if (((int)(k_aLeft.Z * roundLeft)) / roundLeft == 1f | (((int)(k_aLeft.X * roundLeft)) / roundLeft == 0f & ((int)(k_aLeft.Y *
roundLeft)) / roundLeft == 0f))
          k_aLeft = new Vector3(0, 0, 1);
        if (((int)(EulerAnglescLeft.Y * roundLeft)) / roundLeft == 0f | ((int)(EulerAnglesLeft.Y * roundLeft)) / roundLeft == 0f)
        {
          i_cLeft = new Vector3(1, 0, 0);
          j_cLeft = new Vector3(0, 1, 0);
          k_cLeft = new Vector3(0, 0, 1);
          InitEulerAnglescLeft = ToEulerAnglesLeft(GetVectorcLeft());
        }
        if (((int)(EulerAnglesbLeft.X * roundLeft)) / roundLeft == 0f | ((int)(EulerAnglesLeft.X * roundLeft)) / roundLeft == 0f)
        {
          i_bLeft = new Vector3(1, 0, 0);
          j_bLeft = new Vector3(0, 1, 0);
          k_bLeft = new Vector3(0, 0, 1);
          InitEulerAnglesbLeft = ToEulerAnglesLeft(GetVectorbLeft());
        }
        if (((int)(EulerAnglesaLeft.Z * roundLeft)) / roundLeft == 0f | ((int)(EulerAnglesLeft.Z * roundLeft)) / roundLeft == 0f)
        {
          i_aLeft = new Vector3(1, 0, 0);
          j_aLeft = new Vector3(0, 1, 0);
          k_aLeft = new Vector3(0, 0, 1);
          InitEulerAnglesaLeft = ToEulerAnglesLeft(GetVectoraLeft());
        }
        i_cLeft = new Vector3(1, 0, 0);
        j_cLeft.X = 0f;
        k_cLeft.X = 0f;
        k_cLeft.Y = 0f;
```

```
    k_cLeft.Z = 1f;
    k_bLeft = new Vector3(0, 0, 1);
    i_bLeft.Z = 0f;
    j_bLeft.Z = 0f;
    j_aLeft = new Vector3(0, 1, 0);
    i_aLeft.Y = 0f;
    k_aLeft.X = 0f;
    k_aLeft.Y = 0f;
    if (!Getstate)
    {
      i_cLeft = new Vector3(1, 0, 0);
      j_cLeft = new Vector3(0, 1, 0);
      k_cLeft = new Vector3(0, 0, 1);
      InitEulerAnglescLeft = ToEulerAnglesLeft(GetVectorcLeft());
      i_bLeft = new Vector3(1, 0, 0);
      j_bLeft = new Vector3(0, 1, 0);
      k_bLeft = new Vector3(0, 0, 1);
      InitEulerAnglesbLeft = ToEulerAnglesLeft(GetVectorbLeft());
      i_aLeft = new Vector3(1, 0, 0);
      j_aLeft = new Vector3(0, 1, 0);
      k_aLeft = new Vector3(0, 0, 1);
      InitEulerAnglesaLeft = ToEulerAnglesLeft(GetVectoraLeft());
    }
    i_cLeft += Vector3.Cross(Vector3.Negate(gyr_rLeft) * 0.04f, i_cLeft);
    j_cLeft += Vector3.Cross(Vector3.Negate(gyr_rLeft) * 0.04f, j_cLeft);
    k_cLeft += Vector3.Cross(Vector3.Negate(gyr_rLeft) * 0.04f, k_cLeft);
    i_cLeft = Vector3.Normalize(i_cLeft - Vector3.Dot(i_cLeft, j_cLeft) * 0.5f * j_cLeft);
    j_cLeft = Vector3.Normalize(j_cLeft - Vector3.Dot(i_cLeft, j_cLeft) * 0.5f * i_cLeft);
    k_cLeft = Vector3.Cross(i_cLeft, j_cLeft);
    EulerAnglescLeft = ToEulerAnglesLeft(GetVectorcLeft()) - InitEulerAnglescLeft;
    i_bLeft += Vector3.Cross(Vector3.Negate(gyr_rLeft) * 0.04f, i_bLeft);
    j_bLeft += Vector3.Cross(Vector3.Negate(gyr_rLeft) * 0.04f, j_bLeft);
    k_bLeft += Vector3.Cross(Vector3.Negate(gyr_rLeft) * 0.04f, k_bLeft);
    i_bLeft = Vector3.Normalize(i_bLeft - Vector3.Dot(i_bLeft, j_bLeft) * 0.5f * j_bLeft);
    j_bLeft = Vector3.Normalize(j_bLeft - Vector3.Dot(i_bLeft, j_bLeft) * 0.5f * i_bLeft);
    k_bLeft = Vector3.Cross(i_bLeft, j_bLeft);
    EulerAnglesbLeft = ToEulerAnglesLeft(GetVectorbLeft()) - InitEulerAnglesbLeft;
    i_aLeft += Vector3.Cross(Vector3.Negate(gyr_rLeft) * 0.04f, i_aLeft);
    j_aLeft += Vector3.Cross(Vector3.Negate(gyr_rLeft) * 0.04f, j_aLeft);
    k_aLeft += Vector3.Cross(Vector3.Negate(gyr_rLeft) * 0.04f, k_aLeft);
    i_aLeft = Vector3.Normalize(i_aLeft - Vector3.Dot(i_aLeft, j_aLeft) * 0.5f * j_aLeft);
    j_aLeft = Vector3.Normalize(j_aLeft - Vector3.Dot(i_aLeft, j_aLeft) * 0.5f * i_aLeft);
    k_aLeft = Vector3.Cross(i_aLeft, j_aLeft);
    EulerAnglesaLeft = ToEulerAnglesLeft(GetVectoraLeft()) - InitEulerAnglesaLeft;
    EulerAnglesLeft = new Vector3(EulerAnglesbLeft.X, EulerAnglescLeft.Y, EulerAnglesaLeft.Z);
    if (!Getstate)
    {
      WidthS = System.Windows.Forms.Screen.PrimaryScreen.Bounds.Width / 2;
      HeightS = System.Windows.Forms.Screen.PrimaryScreen.Bounds.Height / 2;
      signchangewheelZ = false;
    }
  }
  private double averageLeft(double val1, double val2, double val3)
  {
    arrayLeft = new double[] { val1, val2, val3 };
    return arrayLeft.Average();
  }
  private double[] CenterSticksLeft(UInt16[] vals)
  {
    double[] s = { 0, 0 };
    s[0] = ((int)((vals[0] - stick_calibrationLeft[0]) / 100f)) / 20f;
    s[1] = ((int)((vals[1] - stick_calibrationLeft[1]) / 100f)) / 20f;
    return s;
```

```
    }
    private void SubcommandLeft(byte sc, byte[] buf, uint len)
    {
        Array.Copy(buf, 0, buf_Left, 11, len);
        buf_Left[0] = 0x1;
        buf_Left[1] = 0;
        buf_Left[10] = sc;
        hid_write(handleLeft, buf_Left, (UIntPtr)(len + 11));
        hid_read_timeout(handleLeft, buf_Left, (UIntPtr)49);
    }
    private static double[] stickRight = { 0, 0 };
    private static SafeFileHandle handleRight;
    private static byte[] stick_rawRight = { 0, 0, 0 };
    private static UInt16[] stick_calibrationRight = { 0, 0 };
    private static UInt16[] stick_precalRight = { 0, 0 };
    private static Vector3 acc_gRight = new Vector3();
    private static Vector3 gyr_gRight = new Vector3();
    private static Vector3 acc_rRight = new Vector3();
    private static Vector3 gyr_rRight = new Vector3();
    private const uint report_lenRight = 49;
    public static Vector3 i_cRight = new Vector3(1, 0, 0);
    public static Vector3 j_cRight = new Vector3(0, 1, 0);
    public static Vector3 k_cRight = new Vector3(0, 0, 1);
    public static Vector3 i_bRight = new Vector3(1, 0, 0);
    public static Vector3 j_bRight = new Vector3(0, 1, 0);
    public static Vector3 k_bRight = new Vector3(0, 0, 1);
    public static Vector3 i_aRight = new Vector3(1, 0, 0);
    public static Vector3 j_aRight = new Vector3(0, 1, 0);
    public static Vector3 k_aRight = new Vector3(0, 0, 1);
    private static Vector3 InitDirectAnglesRight, DirectAnglesRight;
    private static Vector3 InitEulerAnglesaRight, EulerAnglesaRight, InitEulerAnglesbRight, EulerAnglesbRight,
InitEulerAnglescRight, EulerAnglescRight, EulerAnglesRight;
    private static bool RightButtonSHOULDER_1, RightButtonSHOULDER_2, RightButtonSR, RightButtonSL,
RightButtonDPAD_DOWN, RightButtonDPAD_RIGHT, RightButtonDPAD_UP, RightButtonDPAD_LEFT, RightButtonPLUS,
RightButtonSTICK, RightButtonHOME;
    private static byte[] report_bufRight, report_bufaRight = new byte[report_lenRight];
    private static double[] arrayRight;
    private static byte[] buf_Right = new byte[report_lenRight];
    public static uint hDevInfoRight;
    public static double indexRight = 0;
    private static float roundRight = 16000f;
    public static float acc_gcalibrationRightX, acc_gcalibrationRightY, acc_gcalibrationRightZ, gyr_gcalibrationRightX,
gyr_gcalibrationRightY, gyr_gcalibrationRightZ;
    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_r))
                {
                    ISRIGHT = true;
```

```
            AttachJoyRight(diDetail.DevicePath);
            AttachJoyRight(diDetail.DevicePath);
            AttachJoyRight(diDetail.DevicePath);
            return true;
          }
        }
        index++;
    }
    return false;
}
public double[] GetStickRight()
{
    return stickRight;
}
public Vector3 GetAccelRight()
{
    if (!Fbool["//swap"])
        return acc_gRight;
    else
        return acc_gLeft;
}
public Quaternion GetVectoraRight()
{
    Vector3 v1 = new Vector3(j_aRight.X, i_aRight.X, k_aRight.X);
    Vector3 v2 = -(new Vector3(j_aRight.Z, i_aRight.Z, k_aRight.Z));
    return QuaternionLookRotationRight(v1, v2);
}
public Quaternion GetVectorbRight()
{
    Vector3 v1 = new Vector3(j_bRight.X, i_bRight.X, k_bRight.X);
    Vector3 v2 = -(new Vector3(j_bRight.Z, i_bRight.Z, k_bRight.Z));
    return QuaternionLookRotationRight(v1, v2);
}
public Quaternion GetVectorcRight()
{
    Vector3 v1 = new Vector3(j_cRight.X, i_cRight.X, k_cRight.X);
    Vector3 v2 = -(new Vector3(j_cRight.Z, i_cRight.Z, k_cRight.Z));
    return QuaternionLookRotationRight(v1, v2);
}
private static Quaternion QuaternionLookRotationRight(Vector3 forward, Vector3 up)
{
    Vector3 vector = Vector3.Normalize(forward);
    Vector3 vector2 = Vector3.Normalize(Vector3.Cross(up, vector));
    Vector3 vector3 = Vector3.Cross(vector, vector2);
    var m00 = vector2.X;
    var m01 = vector2.Y;
    var m02 = vector2.Z;
    var m10 = vector3.X;
    var m11 = vector3.Y;
    var m12 = vector3.Z;
    var m20 = vector.X;
    var m21 = vector.Y;
    var m22 = vector.Z;
    double num8 = (m00 + m11) + m22;
    var quaternion = new Quaternion();
    if (num8 > 0f)
    {
        var num = (double)Math.Sqrt(num8 + 1f);
        quaternion.W = (float)num * 0.5f;
        num = 0.5f / num;
        quaternion.X = (float)(m12 - m21) * (float)num;
        quaternion.Y = (float)(m20 - m02) * (float)num;
        quaternion.Z = (float)(m01 - m10) * (float)num;
```

```
        return quaternion;
    }
    if ((m00 >= m11) && (m00 >= m22))
    {
        var num7 = (double)Math.Sqrt(((1f + m00) - m11) - m22);
        var num4 = 0.5f / num7;
        quaternion.X = 0.5f * (float)num7;
        quaternion.Y = (float)(m01 + m10) * (float)num4;
        quaternion.Z = (float)(m02 + m20) * (float)num4;
        quaternion.W = (float)(m12 - m21) * (float)num4;
        return quaternion;
    }
    if (m11 > m22)
    {
        var num6 = (double)Math.Sqrt(((1f + m11) - m00) - m22);
        var num3 = 0.5f / num6;
        quaternion.X = (float)(m10 + m01) * (float)num3;
        quaternion.Y = 0.5f * (float)num6;
        quaternion.Z = (float)(m21 + m12) * (float)num3;
        quaternion.W = (float)(m20 - m02) * (float)num3;
        return quaternion;
    }
    var num5 = (double)Math.Sqrt(((1f + m22) - m00) - m11);
    var num2 = 0.5f / num5;
    quaternion.X = (float)(m20 + m02) * (float)num2;
    quaternion.Y = (float)(m21 + m12) * (float)num2;
    quaternion.Z = 0.5f * (float)num5;
    quaternion.W = (float)(m01 - m10) * (float)num2;
    return quaternion;
}
public static Vector3 ToEulerAnglesRight(Quaternion q)
{
    Vector3 pitchYawRoll = new Vector3();
    double sqw = q.W * q.W;
    double sqx = q.X * q.X;
    double sqy = q.Y * q.Y;
    double sqz = q.Z * q.Z;
    double unit = sqx + sqy + sqz + sqw;
    double test = q.X * q.Y + q.Z * q.W;
    if (test > 0.4999f * unit)                // 0.4999f OR 0.5f - EPSILON
    {
        pitchYawRoll.Y = 2f * (float)Math.Atan2(q.X, q.W);  // Yaw
        pitchYawRoll.X = (float)Math.PI * 0.5f;             // Pitch
        pitchYawRoll.Z = 0f;                    // Roll
        return pitchYawRoll;
    }
    else if (test < -0.4999f * unit)            // -0.4999f OR -0.5f + EPSILON
    {
        pitchYawRoll.Y = -2f * (float)Math.Atan2(q.X, q.W); // Yaw
        pitchYawRoll.X = -(float)Math.PI * 0.5f;            // Pitch
        pitchYawRoll.Z = 0f;                    // Roll
        return pitchYawRoll;
    }
    else
    {
        pitchYawRoll.Y = (float)Math.Atan2(2f * q.Y * q.W - 2f * q.X * q.Z, sqx - sqy - sqz + sqw);      // Yaw
        pitchYawRoll.X = (float)Math.Asin(2f * test / unit);                                             // Pitch
        pitchYawRoll.Z = (float)Math.Atan2(2f * q.X * q.W - 2f * q.Y * q.Z, -sqx + sqy - sqz + sqw);     // Roll
    }
    return pitchYawRoll;
}
public void AttachJoyRight(string path)
{
```

```
        do
        {
          IntPtr handle = CreateFile(path, System.IO.FileAccess.ReadWrite, System.IO.FileShare.ReadWrite, new
System.IntPtr(), System.IO.FileMode.Open, EFileAttributes.Normal, new System.IntPtr());
          handleRight = hid_open_path(handle);
          SubcommandRight(0x3, new byte[] { 0x30 }, 1);
          SubcommandRight(0x40, new byte[] { 0x1 }, 1);
        }
        while (handleRight.IsInvalid);
      }
      private void ReceiveRawRight()
      {
        Rhid_read_timeout(handleRight, report_bufaRight, (UIntPtr)49);
      }
      private void ProcessButtonsAndStickRight()
      {
        RightButtonSHOULDER_1 = (report_bufRight[3 + (!ISRIGHT ? 2 : 0)] & 0x40) != 0;
        if (!Fbool["//swap"])
          RightButtonSHOULDER_2 = (report_bufRight[3 + (!ISRIGHT ? 2 : 0)] & 0x80) != 0;
        else
          LeftButtonSHOULDER_2 = (report_bufRight[3 + (!ISRIGHT ? 2 : 0)] & 0x80) != 0;
        RightButtonSR = (report_bufRight[3 + (!ISRIGHT ? 2 : 0)] & 0x10) != 0;
        RightButtonSL = (report_bufRight[3 + (!ISRIGHT ? 2 : 0)] & 0x20) != 0;
        RightButtonDPAD_DOWN = (report_bufRight[3 + (!ISRIGHT ? 2 : 0)] & (!ISRIGHT ? 0x01 : 0x04)) != 0;
        RightButtonDPAD_RIGHT = (report_bufRight[3 + (!ISRIGHT ? 2 : 0)] & (!ISRIGHT ? 0x04 : 0x08)) != 0;
        RightButtonDPAD_UP = (report_bufRight[3 + (!ISRIGHT ? 2 : 0)] & (!ISRIGHT ? 0x02 : 0x02)) != 0;
        RightButtonDPAD_LEFT = (report_bufRight[3 + (!ISRIGHT ? 2 : 0)] & (!ISRIGHT ? 0x08 : 0x01)) != 0;
        RightButtonPLUS = ((report_bufRight[4] & 0x02) != 0);
        RightButtonHOME = ((report_bufRight[4] & 0x10) != 0);
        RightButtonSTICK = ((report_bufRight[4] & (!ISRIGHT ? 0x08 : 0x04)) != 0);
        stick_rawRight[0] = report_bufRight[6 + (!ISRIGHT ? 0 : 3)];
        stick_rawRight[1] = report_bufRight[7 + (!ISRIGHT ? 0 : 3)];
        stick_rawRight[2] = report_bufRight[8 + (!ISRIGHT ? 0 : 3)];
        stick_precalRight[0] = (UInt16)(stick_rawRight[0] | ((stick_rawRight[1] & 0xf) << 8));
        stick_precalRight[1] = (UInt16)((stick_rawRight[1] >> 4) | (stick_rawRight[2] << 4));
        stickRight = CenterSticksRight(stick_precalRight);
      }
      private void ExtractIMUValuesRight()
      {
        acc_gRight.X = (int)(averageRight((Int16)(report_bufRight[13 + 0 * 12] | ((report_bufRight[14 + 0 * 12] << 8) &
0xff00)), (Int16)(report_bufRight[13 + 1 * 12] | ((report_bufRight[14 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufRight[13 +
2 * 12] | ((report_bufRight[14 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f) - acc_gcalibrationRightX;
        acc_gRight.Y = -(int)(averageRight((Int16)(report_bufRight[15 + 0 * 12] | ((report_bufRight[16 + 0 * 12] << 8) &
0xff00)), (Int16)(report_bufRight[15 + 1 * 12] | ((report_bufRight[16 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufRight[15 +
2 * 12] | ((report_bufRight[16 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f) - acc_gcalibrationRightY;
        acc_gRight.Z = -(int)(averageRight((Int16)(report_bufRight[17 + 0 * 12] | ((report_bufRight[18 + 0 * 12] << 8) &
0xff00)), (Int16)(report_bufRight[17 + 1 * 12] | ((report_bufRight[18 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufRight[17 +
2 * 12] | ((report_bufRight[18 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f) - acc_gcalibrationRightZ;
        gyr_gRight.X = (int)(averageRight((int)((Int16)((report_bufRight[19 + 0 * 12] | ((report_bufRight[20 + 0 * 12] << 8) &
0xff00)))), (int)((Int16)((report_bufRight[19 + 1 * 12] | ((report_bufRight[20 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufRight[19 + 2 * 12] | ((report_bufRight[20 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f) -
gyr_gcalibrationRightX;
        gyr_gRight.Y = -(int)(averageRight((int)((Int16)((report_bufRight[21 + 0 * 12] | ((report_bufRight[22 + 0 * 12] << 8) &
0xff00)))), (int)((Int16)((report_bufRight[21 + 1 * 12] | ((report_bufRight[22 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufRight[21 + 2 * 12] | ((report_bufRight[22 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f) -
gyr_gcalibrationRightY;
        gyr_gRight.Z = -(int)(averageRight((int)((Int16)((report_bufRight[23 + 0 * 12] | ((report_bufRight[24 + 0 * 12] << 8) &
0xff00)))), (int)((Int16)((report_bufRight[23 + 1 * 12] | ((report_bufRight[24 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufRight[23 + 2 * 12] | ((report_bufRight[24 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f) -
gyr_gcalibrationRightZ;
        acc_rRight.X = ((int)(acc_gRight.X * roundRight)) / roundRight;
        acc_rRight.Y = ((int)(acc_gRight.Y * roundRight)) / roundRight;
        acc_rRight.Z = ((int)(acc_gRight.Z * roundRight)) / roundRight;
```

```
        gyr_rRight.X = ((int)(gyr_gRight.X * roundRight)) / roundRight;
        gyr_rRight.Y = ((int)(gyr_gRight.Y * roundRight)) / roundRight;
        gyr_rRight.Z = ((int)(gyr_gRight.Z * roundRight)) / roundRight;
        if (!Getstate)
          InitDirectAnglesRight = acc_rRight;
        DirectAnglesRight = acc_rRight - InitDirectAnglesRight;
        if (((int)(i_cRight.X * roundRight)) / roundRight == 1f | (((int)(i_cRight.Y * roundRight)) / roundRight == 0f &
((int)(i_cRight.Z * roundRight)) / roundRight == 0f))
          i_cRight = new Vector3(1, 0, 0);
        if (((int)(j_cRight.Y * roundRight)) / roundRight == 1f | (((int)(j_cRight.X * roundRight)) / roundRight == 0f &
((int)(j_cRight.Z * roundRight)) / roundRight == 0f))
          j_cRight = new Vector3(0, 1, 0);
        if (((int)(k_cRight.Z * roundRight)) / roundRight == 1f | (((int)(k_cRight.X * roundRight)) / roundRight == 0f &
((int)(k_cRight.Y * roundRight)) / roundRight == 0f))
          k_cRight = new Vector3(0, 0, 1);
        if (((int)(i_bRight.X * roundRight)) / roundRight == 1f | (((int)(i_bRight.Y * roundRight)) / roundRight == 0f &
((int)(i_bRight.Z * roundRight)) / roundRight == 0f))
          i_bRight = new Vector3(1, 0, 0);
        if (((int)(j_bRight.Y * roundRight)) / roundRight == 1f | (((int)(j_bRight.X * roundRight)) / roundRight == 0f &
((int)(j_bRight.Z * roundRight)) / roundRight == 0f))
          j_bRight = new Vector3(0, 1, 0);
        if (((int)(k_bRight.Z * roundRight)) / roundRight == 1f | (((int)(k_bRight.X * roundRight)) / roundRight == 0f &
((int)(k_bRight.Y * roundRight)) / roundRight == 0f))
          k_bRight = new Vector3(0, 0, 1);
        if (((int)(i_aRight.X * roundRight)) / roundRight == 1f | (((int)(i_aRight.Y * roundRight)) / roundRight == 0f &
((int)(i_aRight.Z * roundRight)) / roundRight == 0f))
          i_aRight = new Vector3(1, 0, 0);
        if (((int)(j_aRight.Y * roundRight)) / roundRight == 1f | (((int)(j_aRight.X * roundRight)) / roundRight == 0f &
((int)(j_aRight.Z * roundRight)) / roundRight == 0f))
          j_aRight = new Vector3(0, 1, 0);
        if (((int)(k_aRight.Z * roundRight)) / roundRight == 1f | (((int)(k_aRight.X * roundRight)) / roundRight == 0f &
((int)(k_aRight.Y * roundRight)) / roundRight == 0f))
          k_aRight = new Vector3(0, 0, 1);
        if (((int)(EulerAnglescRight.Y * roundRight)) / roundRight == 0f | ((int)(EulerAnglesRight.Y * roundRight)) / roundRight
== 0f)
        {
          i_cRight = new Vector3(1, 0, 0);
          j_cRight = new Vector3(0, 1, 0);
          k_cRight = new Vector3(0, 0, 1);
          InitEulerAnglescRight = ToEulerAnglesRight(GetVectorcRight());
        }
        if (((int)(EulerAnglesbRight.X * roundRight)) / roundRight == 0f | ((int)(EulerAnglesRight.X * roundRight)) /
roundRight == 0f)
        {
          i_bRight = new Vector3(1, 0, 0);
          j_bRight = new Vector3(0, 1, 0);
          k_bRight = new Vector3(0, 0, 1);
          InitEulerAnglesbRight = ToEulerAnglesRight(GetVectorbRight());
        }
        if (((int)(EulerAnglesaRight.Z * roundRight)) / roundRight == 0f | ((int)(EulerAnglesRight.Z * roundRight)) / roundRight
== 0f)
        {
          i_aRight = new Vector3(1, 0, 0);
          j_aRight = new Vector3(0, 1, 0);
          k_aRight = new Vector3(0, 0, 1);
          InitEulerAnglesaRight = ToEulerAnglesRight(GetVectoraRight());
        }
        i_cRight = new Vector3(1, 0, 0);
        j_cRight.X = 0f;
        k_cRight.X = 0f;
        k_cRight.Y = 0f;
        k_cRight.Z = 1f;
        k_bRight = new Vector3(0, 0, 1);
```

```
    i_bRight.Z = 0f;
    j_bRight.Z = 0f;
    j_aRight = new Vector3(0, 1, 0);
    i_aRight.Y = 0f;
    k_aRight.X = 0f;
    k_aRight.Y = 0f;
    if (!Getstate)
    {
      i_cRight = new Vector3(1, 0, 0);
      j_cRight = new Vector3(0, 1, 0);
      k_cRight = new Vector3(0, 0, 1);
      InitEulerAnglescRight = ToEulerAnglesRight(GetVectorcRight());
      i_bRight = new Vector3(1, 0, 0);
      j_bRight = new Vector3(0, 1, 0);
      k_bRight = new Vector3(0, 0, 1);
      InitEulerAnglesbRight = ToEulerAnglesRight(GetVectorbRight());
      i_aRight = new Vector3(1, 0, 0);
      j_aRight = new Vector3(0, 1, 0);
      k_aRight = new Vector3(0, 0, 1);
      InitEulerAnglesaRight = ToEulerAnglesRight(GetVectoraRight());
    }
    i_cRight += Vector3.Cross(Vector3.Negate(gyr_rRight) * 0.04f, i_cRight);
    j_cRight += Vector3.Cross(Vector3.Negate(gyr_rRight) * 0.04f, j_cRight);
    k_cRight += Vector3.Cross(Vector3.Negate(gyr_rRight) * 0.04f, k_cRight);
    i_cRight = Vector3.Normalize(i_cRight - Vector3.Dot(i_cRight, j_cRight) * 0.5f * j_cRight);
    j_cRight = Vector3.Normalize(j_cRight - Vector3.Dot(i_cRight, j_cRight) * 0.5f * i_cRight);
    k_cRight = Vector3.Cross(i_cRight, j_cRight);
    EulerAnglescRight = ToEulerAnglesRight(GetVectorcRight()) - InitEulerAnglescRight;
    i_bRight += Vector3.Cross(Vector3.Negate(gyr_rRight) * 0.04f, i_bRight);
    j_bRight += Vector3.Cross(Vector3.Negate(gyr_rRight) * 0.04f, j_bRight);
    k_bRight += Vector3.Cross(Vector3.Negate(gyr_rRight) * 0.04f, k_bRight);
    i_bRight = Vector3.Normalize(i_bRight - Vector3.Dot(i_bRight, j_bRight) * 0.5f * j_bRight);
    j_bRight = Vector3.Normalize(j_bRight - Vector3.Dot(i_bRight, j_bRight) * 0.5f * i_bRight);
    k_bRight = Vector3.Cross(i_bRight, j_bRight);
    EulerAnglesbRight = ToEulerAnglesRight(GetVectorbRight()) - InitEulerAnglesbRight;
    i_aRight += Vector3.Cross(Vector3.Negate(gyr_rRight) * 0.04f, i_aRight);
    j_aRight += Vector3.Cross(Vector3.Negate(gyr_rRight) * 0.04f, j_aRight);
    k_aRight += Vector3.Cross(Vector3.Negate(gyr_rRight) * 0.04f, k_aRight);
    i_aRight = Vector3.Normalize(i_aRight - Vector3.Dot(i_aRight, j_aRight) * 0.5f * j_aRight);
    j_aRight = Vector3.Normalize(j_aRight - Vector3.Dot(i_aRight, j_aRight) * 0.5f * i_aRight);
    k_aRight = Vector3.Cross(i_aRight, j_aRight);
    EulerAnglesaRight = ToEulerAnglesRight(GetVectoraRight()) - InitEulerAnglesaRight;
    EulerAnglesRight = new Vector3(EulerAnglesbRight.X, EulerAnglescRight.Y, EulerAnglesaRight.Z);
    if (!Getstate)
    {
      WidthS = System.Windows.Forms.Screen.PrimaryScreen.Bounds.Width / 2;
      HeightS = System.Windows.Forms.Screen.PrimaryScreen.Bounds.Height / 2;
      signchangewheelZ = false;
    }
  }
  private double averageRight(double val1, double val2, double val3)
  {
    arrayRight = new double[] { val1, val2, val3 };
    return arrayRight.Average();
  }
  private double[] CenterSticksRight(UInt16[] vals)
  {
    double[] s = { 0, 0 };
    s[0] = ((int)((vals[0] - stick_calibrationRight[0]) / 100f)) / 20f;
    s[1] = ((int)((vals[1] - stick_calibrationRight[1]) / 100f)) / 20f;
    return s;
  }
  private void SubcommandRight(byte sc, byte[] buf, uint len)
```

```
    {
      Array.Copy(buf, 0, buf_Right, 11, len);
      buf_Right[0] = 0x1;
      buf_Right[1] = 0;
      buf_Right[10] = sc;
      hid_write(handleRight, buf_Right, (UIntPtr)(len + 11));
      Rhid_read_timeout(handleRight, buf_Right, (UIntPtr)49);
    }
    private void timer1_Tick(object sender, EventArgs e)
    {
      try
      {
        report_bufLeft = report_bufaLeft;
        ProcessButtonsAndStickLeft();
        ExtractIMUValuesLeft();
      }
      catch { }
      try
      {
        report_bufRight = report_bufaRight;
        ProcessButtonsAndStickRight();
        ExtractIMUValuesRight();
      }
      catch { }
    }
  }
}
```

5. 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.