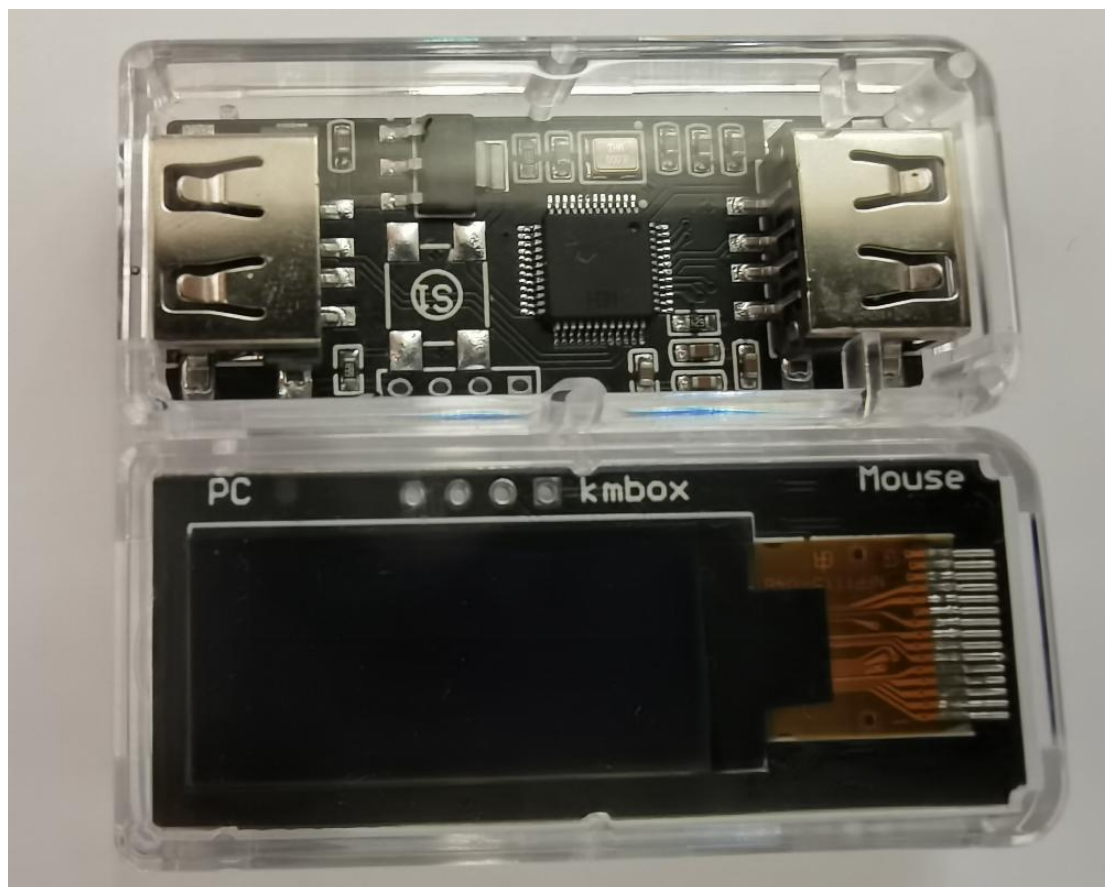


Kmbox 简易版本简介

简易版本 kmbox 是针对 B 版本而言的。虽然简易，但不代表他功能不强大。他是目前常见的易键鼠升级版本。不仅支持现有的易键鼠所有功能。还拥有独特的脱机运行功能。LCD 显示功能。既可以和上位机配套使用（软硬接和外挂），又可以独立脱机使用。并且傻瓜式编程。你只用点几下鼠标便可制作一个纯硬件的物理压枪脚本。自由度虽不及 B 版本，但一般的 FPS 压枪，一键 N 技能这种低端操作一样不含糊。

kmbox 简易版本实物图



正反面



kmbox 简易版本硬件特性

Kmbox 简易版本包含一个 Cortex-M3 核国产 ARM，2 个 USB 硬件端口。一个用户按键（不一定焊接）。左端的 USB 必须连接电脑。右端的 USB 可以连接鼠标或者不连。连不了完全取决于你的应用模式。脱机模式必须连接鼠标。上位机模式可连可不连。

左侧 USB 端口特性

左侧端口必须连接到 PC 端，板卡连接到 PC 后会自动枚举成标准的键盘和鼠标（PC 自动识别不需要任何驱动（键盘鼠标属于标准 HID 设备，系统自带驱动。）。键鼠回报率为 1000Hz. 此 USB 口还支持于上位机直接通信。上位机可以调用 API 直接控制板卡的键鼠数

据。LCD 屏幕显示内容等。详见软件章节。

右侧 USB 端口特性

右侧 USB 口可接鼠标。鼠标数据经过板子中转修改后通过左侧 USB 口上传到电脑。电脑接收的数据均为真实键鼠数据。当运行脱机脚本时。右侧 USB 口可以改变脚本运行状态。

如果你用上位机，那么右端接不接设备没啥关系。如果你想完全脱机运行，右端必须接鼠标控制脱机脚本的运行。

对比易键鼠优势

	Kmbox 简易版本	易键鼠
CPU 主频	72MHz	8MHz
位宽	32bit	8bit
上位机调用	支持	支持
脱机使用	支持	不支持
屏幕显示	支持	不支持
自定义脚本编辑	支持	不支持
外接鼠标	支持	不支持
板载脚本	支持	不支持
安全等级	安全	不安全
...

Kmbox 简易版 CPU 主频 72Mhz@32bit。性能直接甩易键鼠 36 条街。用过易键鼠的肯定知道，易键鼠须有个上位机。不开上位机没有任何功能。Kmbox 简易版可以有上位机，也可以完全没有上位机。有上位机的情况下可能实现变态的图色自瞄等功能。如使用上位机务必做好防检测工作。不要被反外挂机制逮到。在不用上位机，完全脱机模式下。盒子是绝对的安全。直接无视任何检测。

Kmbox 简化板支持板载五组脚本。相当于五档。再加一个空档。每组脚本均可以自定义编程，自定义触发方式。可以无需专业知识。只要点几下鼠标就能制作脚本。详见脚本编辑章节。

连接电脑

将 USB 延长线一端接板子的 USB1 端口。另一端连接电脑到电脑上。你会看到这些内容：



USB 设备 ID:

是 USB 设备的 VID 和 PID。支持修改。修改此 ID 可以防止硬件封号。上位机通过此 ID 与盒子通信。

Factory:

将显示厂商名称，此内容是给定制老板使用。可打上你的厂商名字

Ver:

固件版本号

加密校验信息:

每个板子有唯一的识别码，凡正规渠道购买盒子均加密校验 OK. 如果是盗版盒子（被人抄板的）烧官方固件。此盒子的功能将收到限制。并提示授权失败。授权失败的盒子不支持脱机脚本功能。

如果你不做二次开发可以跳过软件篇，直接去脱机脚本制作章节。

软件篇

Kmbox 的上位机软件可以用来设置盒子，也可以以此为参考，写自己的私有的上位机程序。例如自瞄等需要软件配合的二次开发。Kmbox 上位机完全开源。你可以随意修改。提供 lib 库，dll 供二次开发。源码请去群共享中下载。建议使用 lib 库直接编译到你的工程

中, 不要用 dll, 容易被特征检测。



双击上位机软件后出现如上界面。首先需要连接盒子才能进行后面的操作。上位机通信的 ID (66882021) 可以在盒子的 LCD 上找到。连接成功后所有功能解锁。



注意: 此上位机每个按钮下有对应的源码。你可以参考源码写你自己的上位机。

键盘控制

键盘函数:	
press	0
down	0
up	0

键盘控制一共三个，press,down,up。

press:表示单击键盘的某个按键，效果等同于按下+松开。

```

/*****
键盘指定按键单击函数
输入：    HID键值表
是否测试：是
测试时间：20211001 by hw
注意：由于点击上位机时，电脑焦点在上位机程序里。故键盘的输入
无法直接看到。
*****/
int KM_press(unsigned char vk_key);
  
```

Down:表示键盘某按键一直按下。一般配合 Up 使用。

```

/*****
键盘函数
键盘指定按键一直保持按下
输入：    HID键值表
是否测试：是
测试时间：20211001 by hw
*****/
int KM_down(unsigned char vk_key);
  
```

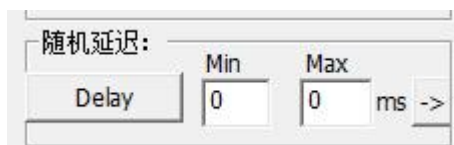
UP: 表示键盘某按键松开。一般配合 Down 使用。

```

/*****
键盘函数
键盘指定按键抬起（配合down函数使用）
输入：    HID键值表
是否测试：是
测试时间：20211001 by hw
*****/
int KM_up(unsigned char vk_key);
  
```

按键后的编辑框中填入你要按键的键值（参考附录 HID 键值表）。然后单击对应的按键就能执行对应的功能。你可以在源码中找到每个函数的用法。二次开发可以此为参考。

随机延迟



随机延迟: Delay Min 0 Max 0 ms ->

随机延迟是给脱机脚本使用的。单击此按钮没有任何效果。

鼠标控制



鼠标函数:	X	Y	
move	0	0	->
left	0		->
right	0		->
middle	0		->
side1	0		->
side2	0		->
wheel	0		->

Move:

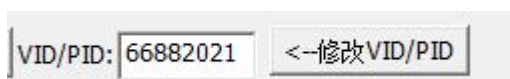
鼠标移动函数, X,Y 为移动的偏移坐标, X 为水平坐标, Y 是垂直坐标, 右为正左为负, 下位正上为负。上位机模式取值范围【-32767, +32717】.脱机脚本模式取值范围【-4095, +4095】.当填入对应的 XY 后, 点击 move 按钮即可移动对应的 XY。

Left,right,middle,side1,side2 这几个函数用于控制鼠标对应的左键, 右键, 中间, 侧键 1, 侧键 2 的状态, 0 为弹起, 1 为按下。

Wheel:

鼠标滚轮控制。取值范围【-127, +127】, 下滚为正上滚负。
以上函数二次开发请参考源码。

修改板卡 ID



VID/PID: 66882021 <-修改VID/PID

将需要修改的 ID 填入左侧的编辑框。点击“修改 VID/PID”按钮后板卡将自动变更硬件 ID。并重启板卡使用新参数。修改 ID 后是永久保存的。

LCDstr 行列显示

LCD指令:

LCDstr	mode:	-1
kmbox		

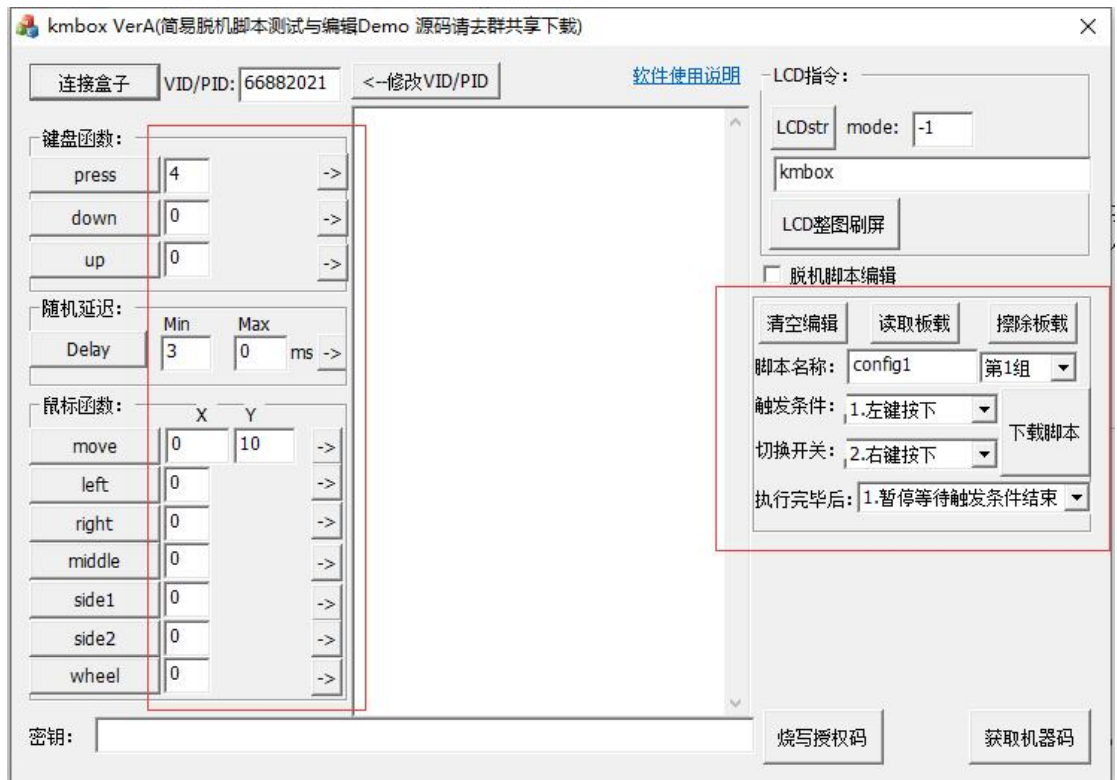
```
/*
LCD显示字符。
mode 是显示模式
    0 : 全屏清理（关闭显示），不会理str
    -1: 滚屏显示，即str显示到最后一行，原来内容顺序上滚
    1: 此模式为指定str显示在显示屏的x,y为起点的位置。
x,y:
    当mode=1时有效，其他情况无效
    x取值0-8
    y取值0-128
*/
int KM_LCDstr(int mode, char *str, int x, int y);
```

单击 LCDstr 按钮会实际调用 KM_LCDstr 函数。详细用法请参考源码。Str 对应的内容在下面的编辑框中。仅支持英文大小写，数字和标点符号。不支持中文。还支持整屏刷新 LCD 刷新数据请参考源码里的 BMP 数组。该数组可用取模软件获得。你可以刷任何你要的图片到 LCD 上显示。图片取模工具群共享下载。

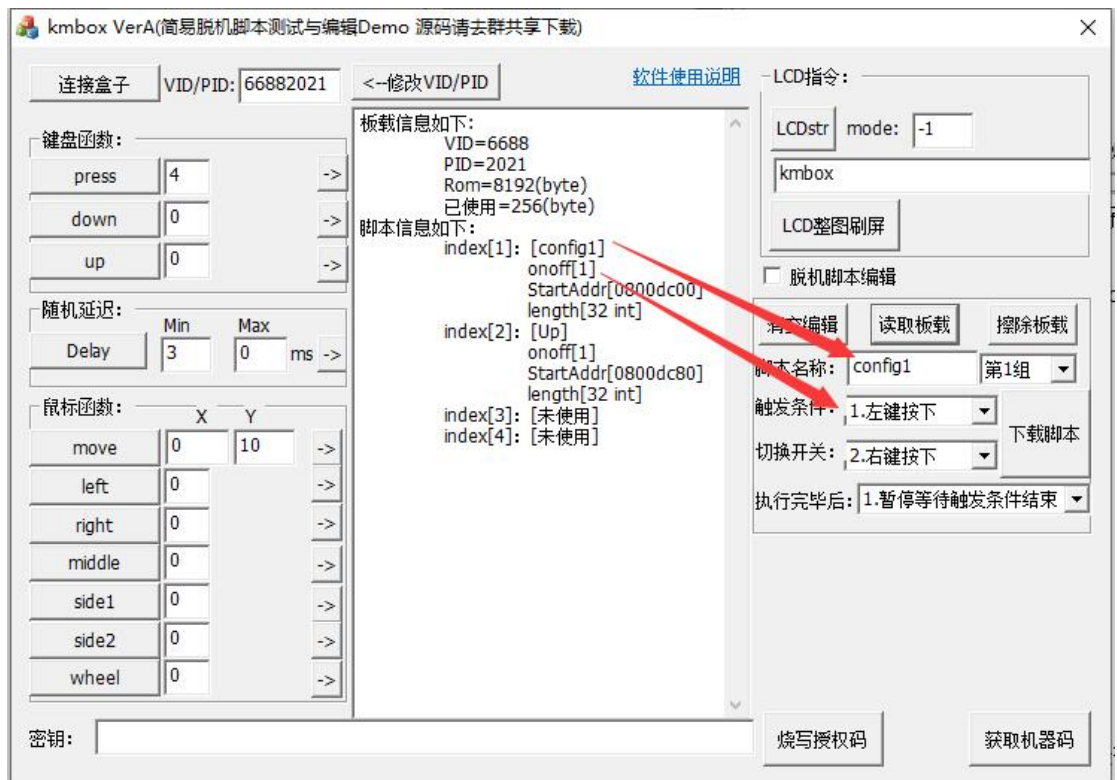
LCD整图刷屏

脱机脚本编辑

盒子支持脱机脚本编辑。一共可以保存五组脱机脚本。每组脱机脚本可单独设置触发方式。脱机脚本编辑主要用到以下区域;



在连接盒子成功后。你可以点击读取板子查看盒子是否存在板子脚本。



板载信息显示目前板卡上一共有两个脚本。1 号脚本名字叫 config1, 2 号脚本叫 Up。两个脚本一共使用了 256 字节。总共可用 8192 字节。脚本触发条件是 1, 表示当鼠标左键按下时触发对应的脚本内容。

举个简单的例子。制作一个压枪脚本。鼠标左键按下时，自动压枪。

首先擦除一下板载脚本。



擦除后可以看到 4 组板载脚本均没有使用。此时可以新建脚本。把此脚本保存在第一组中。



按照上图步骤新建一个压枪脚本。

1: 脚本名称和组。

脚本名称会显示在 LCD 上, 当你切换到对应的组上时会在 LCD 上显示当前脚本名称。

例如上图中的 **AK47**. 当你切换到第一组 (档) 时。LCD 上会显示 **AK47**. 这样你就知道当前运行那一组脚本。

2: 触发条件设置。

触发条件是用来设置满足什么条件时才会自动促发本组脚本的运行。上图中的触发条件是鼠标左键按下。也就是当你切换到 **AK47** 档时, 如果鼠标左键按下。就会自动执行 **delay(5,10)** 的随机延迟, 然后鼠标自动下移 10 个像素。如果促发条件不满足。则上面的脚本将不会执行。


3: 多档位切换

板子一共支持 4 组脚本。也就是四个档位。外加一个空挡。当你有 **AK47** 时你用第一组参数。但是此参数不适合其他枪械。此时你可以新建第二组参数。保存在第二组。假设第二组枪械是冲锋枪汤姆逊。如何从第一组参数切换到第二组参数呢? 这就需要用到多档位切换。上图中, 设置的多档位切换是鼠标中键。也就是说。如果按下鼠标中键。将会自动从当前档位切换到下一个档位。

4: 脚本执行完毕后的动作

上面的脚本就两个动作。随机延迟 5-10ms。鼠标下移 10 个单位。这样组脚本就执行完了。执行完之后可以选择该组脚本是循环执行还是执行完了就执行完了。等触发条件不满足退出。因为是压枪, 所以选择第一种。执行完了再从头执行。这样只要左键按着不松鼠标就会一直下移压枪。

5. 向脚本区域添加脚本指令

所有脚本指令都是通过指令后面的  按钮来添加。因为指令是需要编译成板子能识别的指令。手写容易出错。会导致第八步编译下载不成功。

6: 添加一个延迟指令

Delay(5,10) 表示随机延迟 5 到 10ms 时间。如果第二个参数为 0 表示精确延迟第一个参数的时间。

7: 添加一个鼠标下移指令

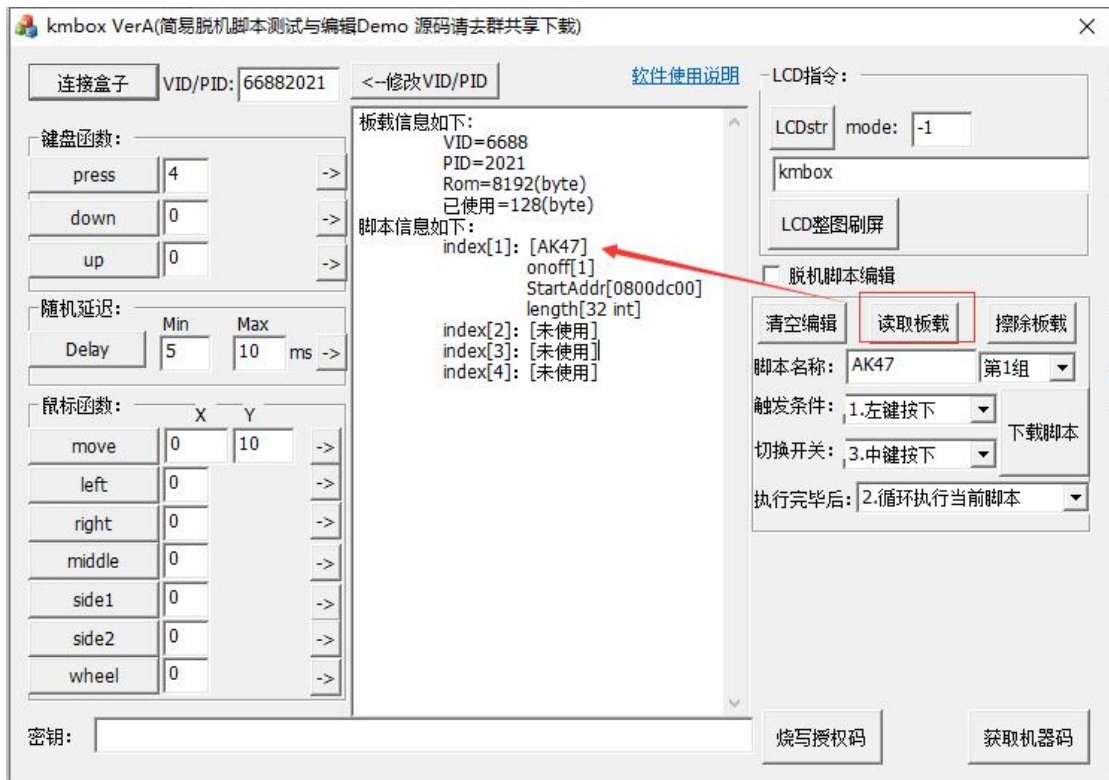
Move(0,10) 表示鼠标 x 轴方向移动 0 个单位, Y 方向下移动 10 个单位。

8: 下载到板卡

将上面编辑好的脚本下载到板卡内部。然后你就可以开始测试这个脚本是否满足你的要求了。

为了简便。我只写了 2 条脚本语句。你可以根据自己的需求任意增加你要的脚本内容。

目前脚本一共有 8KB 的存储空间。大约能存 2K 条脚本指令。

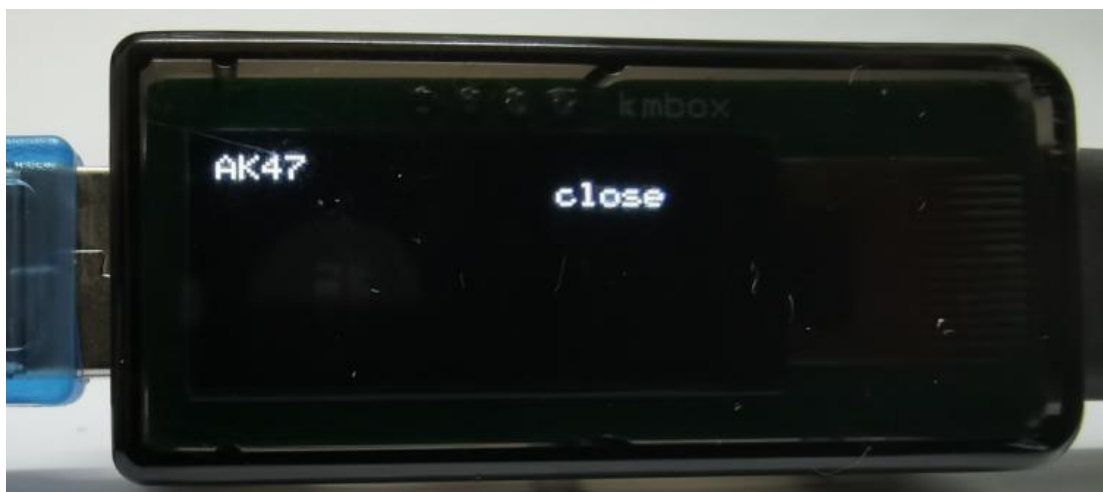


你可以点击读取板载来查看当前四组脚本的存储分配情况。

注意。脱机脚本是需要右端接鼠标的。只有满足触发条件才会触发刚刚下载脚本。下载到盒子后单击鼠标中键，可以看到他切换到 AK47 组了。



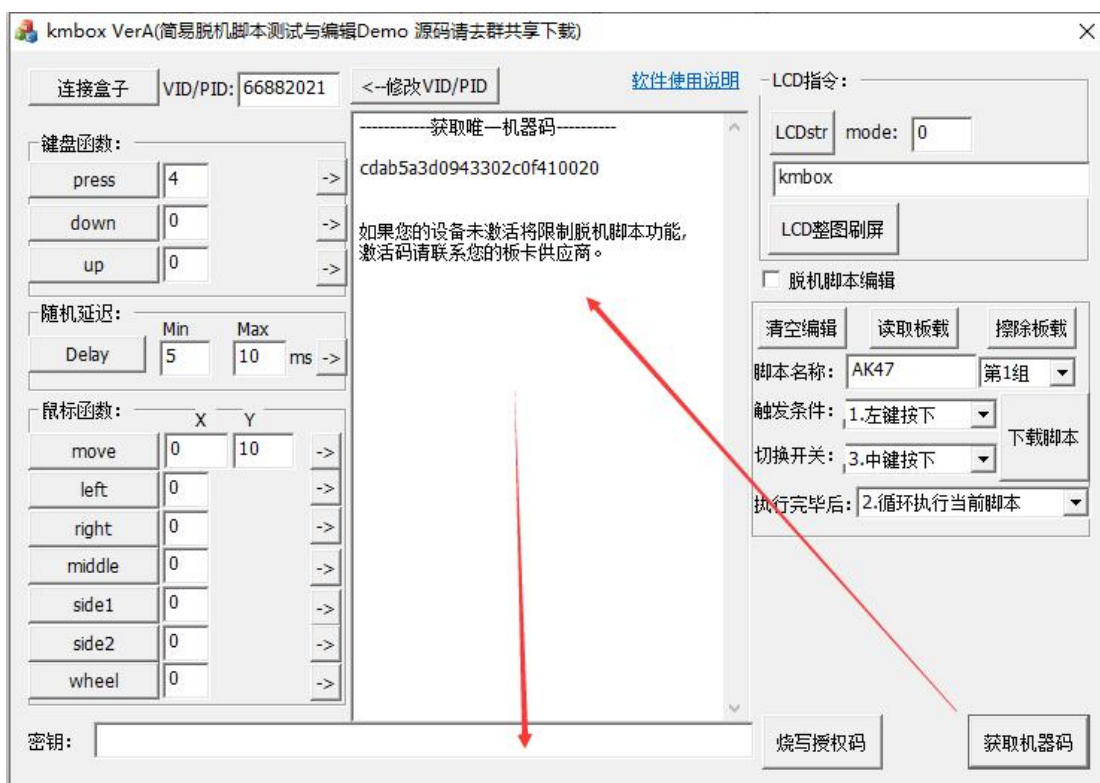
此时如果按住鼠标左键，那么他会每隔 5-10ms 时间鼠标下压 10 个单位。松开鼠标左键。下压停止。如果想关闭该组的脚本。就再按一次鼠标中键。组间切换。此时脚本会切换到空挡。



如果想再次切换，再切中键到对应的标签即可。

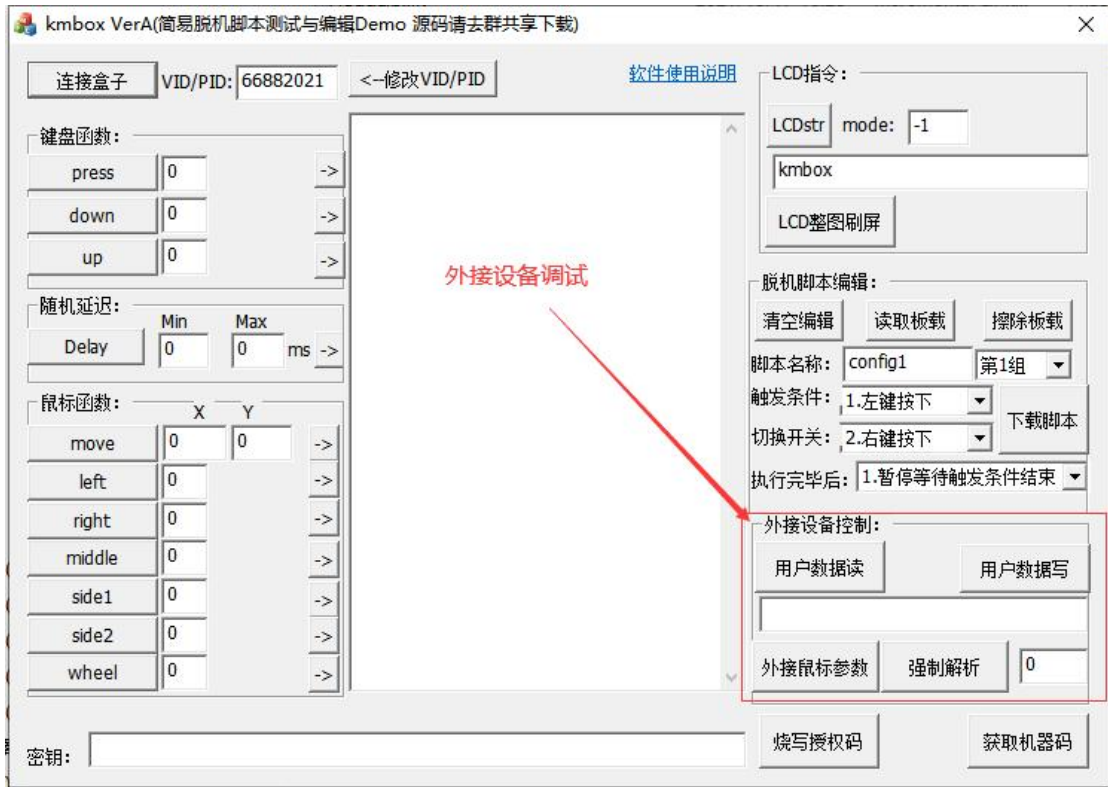
安全与授权码

正规渠道的板子无需授权，自动激活。如果是抄板烧官方固件。就会出现不激活的情况。板卡部分功能将会受到限制。只有激活后才能解除。没有授权码请不要试图暴力破解。板卡授权码连续错误 10 次会直接格式化所有数据。Flash 写次数是大约为 10 万次。暴力破解只会加快芯片损耗。目前最牛逼的计算机暴力破解也要几亿年。有破解的功夫还不如自己写一套代码。授权码请从你的板卡供应商获取。



板子不提供任何读取 flash 数据功能。保证板载脚本的安全。如果你进行二次开发。板卡里运行的脚本数据只有作者自己知道。无法第三方获得。保证作者的劳动成果不被非法侵占。

外接设备匹配与专用加密



盒子提供一个用户数据读写区域。可用于上位机二次加密校验。例如你可以向此区域写一个 RSA 加密的数据。当上位机连接成功后，读取用户数据。在上位机上校验。如果满足上位机的校验机制则是自己的板卡。不满足则上位机拒绝运行。此区域一共 64 字节。

盒子作为主机，会按照通用标准的 HID 设备来枚举连接到盒子上的鼠标。但是现在的鼠标真的千奇百怪。如果鼠标无法正常工作。或者滚轮失效。或者侧键失效。你可以尝试读取外接鼠标参数，然后强制解析来匹配。



如图所示。当点击“外接鼠标参数”按钮后。盒子会返回当前连接到盒子的鼠标参数“04D92083-01200000-00”。前面八个是 VID 和 PID,后面 8 个是 HID 和 DID..最后两个 00 是鼠标的解析模式。如果你的鼠标插上盒子后工作异常。

例如罗技的 M100r 鼠标插盒子上，虽然移动点击正常，但是滚轮没有反应。首先将鼠标插盒子上。点击“外接鼠标参数”读取一下。



可以看到当前解析模式是 FE。此模式无法解析鼠标的滚轮。因此可以强制解析成其他模式。在强制解析模式右边填上 0。点击强制解析。点击完毕后。盒子会自动重启。并将当前鼠标强制按照模式 0 解析。此时可以再测试鼠标是否正常。

强制解析模式目前取值范围是 0-16。你的鼠标适合那种我也不知道。一个一个的尝试吧。总有一个适合。如果都尝试了一遍还不行。那么换鼠标吧。

PS:

模式不匹配会导致鼠标异常。如果不小心将模式匹配错误。可以将鼠标从盒子上拔下。读取一次当前外接鼠标参数。返回的全是“00000000-00000000-00” 此时再点击强制解析。那么原来保存的解析模式就会清除。

强制匹配是永久保存的。只要不换鼠标就一直可以用。如果换了鼠标。不正常了按照前面的步骤再匹配一次即可。

附录

附录 1. 键盘全键值对应表

此表是键盘 HID 数据与按键对应值，也就是 table 函数的内容。 左边是按键名称，右边是 HEX 值。如果使用不在 table 函数中的按键，请记得将 HEX 转换为 10 进制使用。

#define KEY_NONE	0x00
#define KEY_ERRORROLLOVER	0x01
#define KEY_POSTFAIL	0x02
#define KEY_ERRORUNDEFINED	0x03
#define KEY_A	0x04
#define KEY_B	0x05
#define KEY_C	0x06
#define KEY_D	0x07
#define KEY_E	0x08

#define KEY_F	0x09
#define KEY_G	0x0A
#define KEY_H	0x0B
#define KEY_I	0x0C
#define KEY_J	0x0D
#define KEY_K	0x0E
#define KEY_L	0x0F
#define KEY_M	0x10
#define KEY_N	0x11
#define KEY_O	0x12
#define KEY_P	0x13
#define KEY_Q	0x14
#define KEY_R	0x15
#define KEY_S	0x16
#define KEY_T	0x17
#define KEY_U	0x18
#define KEY_V	0x19
#define KEY_W	0x1A
#define KEY_X	0x1B
#define KEY_Y	0x1C
#define KEY_Z	0x1D
#define KEY_1_EXCLAMATION_MARK	0x1E
#define KEY_2_AT	0x1F
#define KEY_3_NUMBER_SIGN	0x20
#define KEY_4_DOLLAR	0x21
#define KEY_5_PERCENT	0x22
#define KEY_6_CARET	0x23
#define KEY_7_AMPERSAND	0x24
#define KEY_8_ASTERISK	0x25
#define KEY_9_OPARENTHESIS	0x26
#define KEY_0_CPARENTHESIS	0x27
#define KEY_ENTER	0x28
#define KEY_ESCAPE	0x29
#define KEY_BACKSPACE	0x2A
#define KEY_TAB	0x2B
#define KEY_SPACEBAR	0x2C
#define KEY_MINUS_UNDERSCORE	0x2D
#define KEY_EQUAL_PLUS	0x2E
#define KEY_OBRACKET_AND_OBRACE	0x2F
#define KEY_CBRACKET_AND_CBACE	0x30
#define KEY_BACKSLASH_VERTICAL_BAR	0x31
#define KEY_NONUS_NUMBER_SIGN_TILDE	0x32
#define KEY_SEMICOLON_COLON	0x33
#define KEY_SINGLE_AND_DOUBLE_QUOTE	0x34

#define KEY_GRAVE ACCENT AND TILDE	0x35
#define KEY_COMMA AND LESS	0x36
#define KEY_DOT GREATER	0x37
#define KEY_SLASH QUESTION	0x38
#define KEY_CAPS LOCK	0x39
#define KEY_F1	0x3A
#define KEY_F2	0x3B
#define KEY_F3	0x3C
#define KEY_F4	0x3D
#define KEY_F5	0x3E
#define KEY_F6	0x3F
#define KEY_F7	0x40
#define KEY_F8	0x41
#define KEY_F9	0x42
#define KEY_F10	0x43
#define KEY_F11	0x44
#define KEY_F12	0x45
#define KEY_PRINTSCREEN	0x46
#define KEY_SCROLL LOCK	0x47
#define KEY_PAUSE	0x48
#define KEY_INSERT	0x49
#define KEY_HOME	0x4A
#define KEY_PAGEUP	0x4B
#define KEY_DELETE	0x4C
#define KEY_END1	0x4D
#define KEY_PAGEDOWN	0x4E
#define KEY_RIGHTARROW	0x4F
#define KEY_LEFTARROW	0x50
#define KEY_DOWNARROW	0x51
#define KEY_UPARROW	0x52
#define KEY_KEYPAD_NUM_LOCK AND CLEAR	0x53
#define KEY_KEYPAD_SLASH	0x54
#define KEY_KEYPAD_asteriks	0x55
#define KEY_KEYPAD_MINUS	0x56
#define KEY_KEYPAD_PLUS	0x57
#define KEY_KEYPAD_ENTER	0x58
#define KEY_KEYPAD_1_END	0x59
#define KEY_KEYPAD_2_DOWN_ARROW	0x5A
#define KEY_KEYPAD_3_PAGEDN	0x5B
#define KEY_KEYPAD_4_LEFT_ARROW	0x5C
#define KEY_KEYPAD_5	0x5D
#define KEY_KEYPAD_6_RIGHT_ARROW	0x5E
#define KEY_KEYPAD_7_HOME	0x5F
#define KEY_KEYPAD_8_UP_ARROW	0x60

#define KEY_KEYPAD_9_PAGEUP	0x61
#define KEY_KEYPAD_0_INSERT	0x62
#define KEY_KEYPAD_DECIMAL_SEPARATOR_DELETE	0x63
#define KEY_NONUS_BACK_SLASH_VERTICAL_BAR	0x64
#define KEY_APPLICATION	0x65
#define KEY_POWER	0x66
#define KEY_KEYPAD_EQUAL	0x67
#define KEY_F13	0x68
#define KEY_F14	0x69
#define KEY_F15	0x6A
#define KEY_F16	0x6B
#define KEY_F17	0x6C
#define KEY_F18	0x6D
#define KEY_F19	0x6E
#define KEY_F20	0x6F
#define KEY_F21	0x70
#define KEY_F22	0x71
#define KEY_F23	0x72
#define KEY_F24	0x73
#define KEY_EXECUTE	0x74
#define KEY_HELP	0x75
#define KEY_MENU	0x76
#define KEY_SELECT	0x77
#define KEY_STOP	0x78
#define KEY_AGAIN	0x79
#define KEY_UNDO	0x7A
#define KEY_CUT	0x7B
#define KEY_COPY	0x7C
#define KEY_PASTE	0x7D
#define KEY_FIND	0x7E
#define KEY_MUTE	0x7F
#define KEY_VOLUME_UP	0x80
#define KEY_VOLUME_DOWN	0x81
#define KEY_LOCKING_CAPS_LOCK	0x82
#define KEY_LOCKING_NUM_LOCK	0x83
#define KEY_LOCKING_SCROLL_LOCK	0x84
#define KEY_KEYPAD_COMMA	0x85
#define KEY_KEYPAD_EQUAL_SIGN	0x86
#define KEY_INTERNATIONAL1	0x87
#define KEY_INTERNATIONAL2	0x88
#define KEY_INTERNATIONAL3	0x89
#define KEY_INTERNATIONAL4	0x8A
#define KEY_INTERNATIONAL5	0x8B
#define KEY_INTERNATIONAL6	0x8C

#define KEY_INTERNATIONAL7	0x8D
#define KEY_INTERNATIONAL8	0x8E
#define KEY_INTERNATIONAL9	0x8F
#define KEY_LANG1	0x90
#define KEY_LANG2	0x91
#define KEY_LANG3	0x92
#define KEY_LANG4	0x93
#define KEY_LANG5	0x94
#define KEY_LANG6	0x95
#define KEY_LANG7	0x96
#define KEY_LANG8	0x97
#define KEY_LANG9	0x98
#define KEY_ALTERNATE_ERASE	0x99
#define KEY_SYSREQ	0x9A
#define KEY_CANCEL	0x9B
#define KEY_CLEAR	0x9C
#define KEY_PRIOR	0x9D
#define KEY_RETURN	0x9E
#define KEY_SEPARATOR	0x9F
#define KEY_OUT	0xA0
#define KEY_OPER	0xA1
#define KEY_CLEAR_AGAIN	0xA2
#define KEY_CRSEL	0xA3
#define KEY_EXSEL	0xA4
#define KEY_KEYPAD_00	0xB0
#define KEY_KEYPAD_000	0xB1
#define KEY_THOUSANDS_SEPARATOR	0xB2
#define KEY_DECIMAL_SEPARATOR	0xB3
#define KEY_CURRENCY_UNIT	0xB4
#define KEY_CURRENCY_SUB_UNIT	0xB5
#define KEY_KEYPAD_OPARENTHESIS	0xB6
#define KEY_KEYPAD_CPARENTHESIS	0xB7
#define KEY_KEYPAD_OBRACE	0xB8
#define KEY_KEYPAD_CBACE	0xB9
#define KEY_KEYPAD_TAB	0xBA
#define KEY_KEYPAD_BACKSPACE	0xBB
#define KEY_KEYPAD_A	0xBC
#define KEY_KEYPAD_B	0xBD
#define KEY_KEYPAD_C	0xBE
#define KEY_KEYPAD_D	0xBF
#define KEY_KEYPAD_E	0xC0
#define KEY_KEYPAD_F	0xC1
#define KEY_KEYPAD_XOR	0xC2
#define KEY_KEYPAD_CARET	0xC3


```
#define KEY_KEYPAD_PERCENT 0xC4
#define KEY_KEYPAD_LESS 0xC5
#define KEY_KEYPAD_GREATER 0xC6
#define KEY_KEYPAD_AMPERSAND 0xC7
#define KEY_KEYPAD_LOGICAL_AND 0xC8
#define KEY_KEYPAD_VERTICAL_BAR 0xC9
#define KEY_KEYPAD_LOGIACL_OR 0xCA
#define KEY_KEYPAD_COLON 0xCB
#define KEY_KEYPAD_NUMBER_SIGN 0xCC
#define KEY_KEYPAD_SPACE 0xCD
#define KEY_KEYPAD_AT 0xCE
#define KEY_KEYPAD_EXCLAMATION_MARK 0xCF
#define KEY_KEYPAD_MEMORY_STORE 0xD0
#define KEY_KEYPAD_MEMORY_RECALL 0xD1
#define KEY_KEYPAD_MEMORY_CLEAR 0xD2
#define KEY_KEYPAD_MEMORY_ADD 0xD3
#define KEY_KEYPAD_MEMORY_SUBTRACT 0xD4
#define KEY_KEYPAD_MEMORY_MULTIPLY 0xD5
#define KEY_KEYPAD_MEMORY_DIVIDE 0xD6
#define KEY_KEYPAD_PLUSMINUS 0xD7
#define KEY_KEYPAD_CLEAR 0xD8
#define KEY_KEYPAD_CLEAR_ENTRY 0xD9
#define KEY_KEYPAD_BINARY 0xDA
#define KEY_KEYPAD_OCTAL 0xDB
#define KEY_KEYPAD_DECIMAL 0xDC
#define KEY_KEYPAD_HEXADECIMAL 0xDD
#define KEY_LEFTCONTROL 0xE0
#define KEY_LEFTSHIFT 0xE1
#define KEY_LEFTALT 0xE2
#define KEY_LEFT_GUI 0xE3
#define KEY_RIGHTCONTROL 0xE4
#define KEY_RIGHTSHIFT 0xE5
#define KEY_RIGHTALT 0xE6
#define KEY_RIGHT_GUI 0xE7
```