Alan Rieger and Ayden Dauenhauer

5/20/2024

ELEN 121L Lab 7 Report

Part 1:

1) What value in the code needs to be changed to alter the amount of LED movement for a given amount of mouse movement? Identify the value, then adjust it to work well with your mouse.

We need to change the constant SPOTFRAC to alter the amount of LED movement to the amount of mouse movement. We noticed that a high value resulted in less LED spots per mouse movement, and small values resulted in more LED spots per mouse movement. We changed it from 100 to 150 to slow the movement down which seemed to work well for our mouse.

```
#define SPOTFRAC 150    // fractional representation of the spot location
```

2) What is the Vendor ID and the Product ID for your mouse? Also go to devicehunt.com and see what it knows about your Vendor ID and the Product ID.

The Vendor ID is 0x046D and the Product ID is 0xC077. Devicehunt just knows from the Vendor ID and Product ID that it is a Logitech Mouse but it doesn't know that it is specifically a B100.

phost->device.DevDesc.idP...	0xC077	ushort
phost->device.DevDesc.idV...	0x046D	ushort
<Enter expression>		

3) How many USB pipes are open to your mouse? How do you know?

There are 16 USB pipes open to the mouse. We know because there is an array of pipes with a length of 16.

```
uint32_t                Pipes[16];
```

4) Does USBH_HID_SOFProcess() ever get called in this program? How do you know?

USBH_HID_SOFProcess gets called when you initialize an HID class. We know this because we CNTR + F the function name and noticed it is only referred to by the HID class struct. Since the mouse is a HID class, the initialization to this function is called.

```
static USBH_StatusTypeDef USBH_HID_SOFProcess(USBH_HandleTypeDef *phost);
static void  USBH_HID_ParseHIDDesc(HID_DescTypeDef *desc, uint8_t *buf);

extern USBH_StatusTypeDef USBH_HID_MouseInit(USBH_HandleTypeDef *phost);
extern USBH_StatusTypeDef USBH_HID_KeybdInit(USBH_HandleTypeDef *phost);

USBH_ClassTypeDef  HID_Class =
{
  "HID",
  USB_HID_CLASS,
  USBH_HID_InterfaceInit,
  USBH_HID_InterfaceDeInit,
  USBH_HID_ClassRequest,
  USBH_HID_Process,
  USBH_HID_SOFProcess,
  NULL,
};
```

5) What does USBH_UsrLog() do?

USBH_UsrLog creates a log of what is being found about the USB device. It runs constantly and is a debug function.

```
{
  USBH_UsrLog("PID: %xh", phost->device.DevDesc.idProduct);
  USBH_UsrLog("VID: %xh", phost->device.DevDesc.idVendor);
```

*prints the Product ID and Vendor ID to keep track of what it is.

```
#define  USBH_UsrLog(...)    do { \
                                 printf(__VA_ARGS__); \
                                 printf("\n"); \
} while (0)
#else
#define USBH_UsrLog(...) do {} while (0)
#endif
```

6) What does fixData() do?

fixData is a mask which sign extends if the number is negative. It is only called when the system gets the mouse info. It changes the 8 bit number and returns a 32 bit number.

```
217       if (devType == HID_MOUSE) {
218         mouseInfo = USBH_HID_GetMouseInfo(&hUsbHostFS);
219         if (mouseInfo != NULL) {
220           spotLocation = spotUpdate(spotLocation, fixData(mouseInfo->x));
221         }
```

*sign extends mouseInfo if negative to use to update the blue light's spot location

```
150 int fixData(uint8_t d) {
151   if ((d & 0x80) != 0)
152     return 0xffffff00 | (int) d;
153   else
154     return (int) d;
```

Part 2:

Left Click: mouse_info.buttons[0]

Right Click : mouse_info.buttons[1]

Middle Click: mouse_info.buttons[2]

```
extern HID_MOUSE_Info_TypeDef    mouse_info;
```

```
190 int currcolor=KBLUE;
191  /* USER CODE END 2 */
192
193  /* Infinite loop */
194  /* USER CODE BEGIN WHILE */
195  while (1)
196  {
197    setDot(colors, NUM_LEDS, spotLocation, currcolor);
198    if(mouse_info.buttons[0]==1)
199    {
200      currcolor=KRED;
201    }
202    if(mouse_info.buttons[1]==1)
203    {
204      currcolor=KPURPLE;
205    }
206    if(mouse_info.buttons[2]==1)
207    {
208      currcolor=KGREEN;
209    }
210    send_array(colors);
211    /* USER CODE END WHILE */
```

Part 3:

```
extern HID_MOUSE_Info_TypeDef    mouse_info;
```

```
162  int main(void)
163  {
164    /* USER CODE BEGIN 1 */
165
166    /* USER CODE END 1 */
167
168    /* MCU Configuration--------------------------------------------------------*/
169
170    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
171    HAL_Init();
172
173    /* USER CODE BEGIN Init */
174
175    /* USER CODE END Init */
176
177    /* Configure the system clock */
178    SystemClock_Config();
179
180    /* USER CODE BEGIN SysInit */
181
182    /* USER CODE END SysInit */
183
184    /* Initialize all configured peripherals */
185    MX_GPIO_Init();
186    MX_USB_HOST_Init();
187    /* USER CODE BEGIN 2 */
188    int currcolor[8]={KRED, KYELLOW, KPURPLE, KGREEN, KBLUE, KINDIGO, KWHITE, KORANGE};
189    int index=4;
190    int flag=0;
191    /* USER CODE END 2 */
192
193    /* Infinite loop */
194    /* USER CODE BEGIN WHILE */
195    while (1)
196    {
197      setDot(colors, NUM_LEDS, spotLocation, currcolor[index]);
198      send_array(colors);
199      /* USER CODE END WHILE */
200      MX_USB_HOST_Process();
201
202      /* USER CODE BEGIN 3 */
203      if (hUsbHostFS.gState == HOST_CLASS) {
```

```
204
205 #ifdef KBMOUSECOMBO
206         hUsbHostFS.device.current_interface = 1;
207 #endif
208       devType = USBH_HID_GetDeviceType(&hUsbHostFS);
209       if (devType == HID_MOUSE) {
210         HAL_GPIO_WritePin(LD_R_GPIO_Port, LD_R_Pin,GPIO_PIN_SET);
211       } else {
212         HAL_GPIO_WritePin(LD_R_GPIO_Port, LD_R_Pin,GPIO_PIN_RESET);
213       }
214       if (devType == HID_KEYBOARD) {
215         HAL_GPIO_WritePin(LD_G_GPIO_Port, LD_G_Pin,GPIO_PIN_SET);
216       } else {
217         HAL_GPIO_WritePin(LD_G_GPIO_Port, LD_G_Pin,GPIO_PIN_RESET);
218       }
219       if (devType == HID_MOUSE) {
220         mouseInfo = USBH_HID_GetMouseInfo(&hUsbHostFS);
221         if (mouseInfo != NULL) {
222           spotLocation = spotUpdate(spotLocation, fixData(mouseInfo->x));
223             if(mouse_info.buttons[0]!=0){
224                 flag=1;
225             }
226             if(mouse_info.buttons[0]!=1&&flag==1){
227                 index++;
228                 if(index>7){
229                   index=0;
230                 }
231                 flag=0;
232             }
233         }
234         }
235       }
236     else if (hUsbHostFS.gState == HOST_IDLE) {
237       HAL_GPIO_WritePin(LD_G_GPIO_Port, LD_G_Pin,GPIO_PIN_RESET);
238       HAL_GPIO_WritePin(LD_R_GPIO_Port, LD_R_Pin,GPIO_PIN_RESET);
239     }
240   }
241   /* USER CODE END 3 */
242 }
```