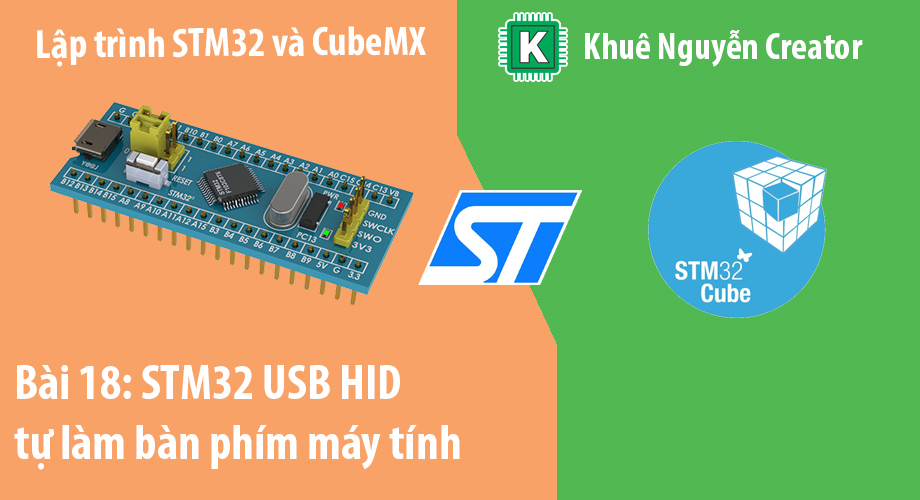
**Bài 18: Lập trình SMT32 USB HID Keyboard bàn phím máy tính**

POSTED ON [11/05/2021](https://khuenguyencreator.com/lap-trinh-smt32-usb-hid-keyboard-ban-phim-may-tinh/) BY [KHUÊ NGUYỄN](https://khuenguyencreator.com/author/nguyenkhue2608/)

[](https://khuenguyencreator.com/lap-trinh-smt32-usb-hid-keyboard-ban-phim-may-tinh/)

**11  
Th5**

Ở bài trước chúng ta đã làm quen với USB HID để tạo 1 con chuột máy tính, ở bài này chúng ta sẽ sử dụng STM32 USB HID Keyboard để chế tạo 1 chiếc bàn phím đơn giản, giao tiếp với máy tính nhé.

Bài 18 trong Serie [Lập trình STM32 từ A tới Z](https://khuenguyencreator.com/lap-trinh-stm32-tu-a-toi-z/)

Mục Lục

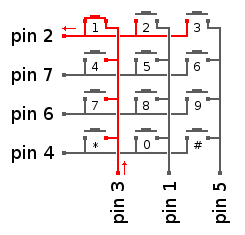
[](https://khuenguyencreator.com/lap-trinh-smt32-usb-hid-keyboard-ban-phim-may-tinh/)

* [Bàn phím máy tính hoạt động như thế nào?](https://khuenguyencreator.com/lap-trinh-smt32-usb-hid-keyboard-ban-phim-may-tinh/#Ban_phim_may_tinh_hoat_dong_nhu_the_nao)
  + [Lý thuyết quét phím ma trận](https://khuenguyencreator.com/lap-trinh-smt32-usb-hid-keyboard-ban-phim-may-tinh/#Ly_thuyet_quet_phim_ma_tran)
  + [Bàn phím thực tế](https://khuenguyencreator.com/lap-trinh-smt32-usb-hid-keyboard-ban-phim-may-tinh/#Ban_phim_thuc_te)
* [Report Descriptor USB HID của bàn phím máy tính](https://khuenguyencreator.com/lap-trinh-smt32-usb-hid-keyboard-ban-phim-may-tinh/#Report_Descriptor_USB_HID_cua_ban_phim_may_tinh)
  + [Phân tích Report Descriptor Keyboard](https://khuenguyencreator.com/lap-trinh-smt32-usb-hid-keyboard-ban-phim-may-tinh/#Phan_tich_Report_Descriptor_Keyboard)
  + [Cấu trúc dữ liệu khi gửi của HID KeyBoard](https://khuenguyencreator.com/lap-trinh-smt32-usb-hid-keyboard-ban-phim-may-tinh/#Cau_truc_du_lieu_khi_gui_cua_HID_KeyBoard)
* [Lập trình STM32 USB HID Keyboard bàn phím máy tính](https://khuenguyencreator.com/lap-trinh-smt32-usb-hid-keyboard-ban-phim-may-tinh/#Lap_trinh_STM32_USB_HID_Keyboard_ban_phim_may_tinh)
  + [Cấu hình STM32 USB HID Keyboard trên CubeMX](https://khuenguyencreator.com/lap-trinh-smt32-usb-hid-keyboard-ban-phim-may-tinh/#Cau_hinh_STM32_USB_HID_Keyboard_tren_CubeMX)
  + [Lập trình STM32 USB HID Keyboard trên Keil C](https://khuenguyencreator.com/lap-trinh-smt32-usb-hid-keyboard-ban-phim-may-tinh/#Lap_trinh_STM32_USB_HID_Keyboard_tren_Keil_C)
  + [Kết nối phần cứng](https://khuenguyencreator.com/lap-trinh-smt32-usb-hid-keyboard-ban-phim-may-tinh/#Ket_noi_phan_cung)
  + [Kết quả](https://khuenguyencreator.com/lap-trinh-smt32-usb-hid-keyboard-ban-phim-may-tinh/#Ket_qua)
* [Kết](https://khuenguyencreator.com/lap-trinh-smt32-usb-hid-keyboard-ban-phim-may-tinh/#Ket)
  + [Related posts:](https://khuenguyencreator.com/lap-trinh-smt32-usb-hid-keyboard-ban-phim-may-tinh/#Related_posts)

**Bàn phím máy tính hoạt động như thế nào?**

**Lý thuyết quét phím ma trận**

Về mặt phần cứng Bàn phím (Keyboard) là tập hợp các nút nhấn được thiết kế theo dạng ma trận phím. Nghĩa là chúng sẽ được nối chung hàng và cột. Vi điều khiển sẽ quét hàng hoặc cột theo một chu kì cố định để tìm kiếm sự thay đổi về trạng thái nút. Sau đó gửi lên máy tính thông qua chuẩn USB HID

quét phím ma trận 4×3

Ví dụ: Trong hình trên chúng ta có 1 ma trận phím 4×3.

Để đọc được giá trị nhấn vi điều khiển sẽ lần lượt ghi mức 1 vào các chân 2 7 6 4 (Quét Hàng). Sau đó sẽ đọc giá trị từ 3 chân 3 1 5.  
Nếu khi quét pin 2 mà pin3 có tín hiệu  => nút 1 được nhấn.

Tương tự với các nút còn lại.

Khi vi điều khiển biết được nút nào được nhấn. Nó sẽ gửi 1 gói tin sang vi điều khiển tương ứng với nút đó trên bàn phím, để máy tính hiểu rằng, phím đã được nhấn.

Để hiểu rõ hơn về bàn phím các bạn đọc bài viết: [Lập trình STM32 giao tiếp ma trận bàn phím 3×4](https://khuenguyencreator.com/lap-trinh-stm32-quet-ma-tran-phim-keypad-3x4/)

**Bàn phím thực tế**

Trong thực tế bàn phím máy tính có rất nhiều nút nhấn. Bao gồm 2 loại chính:

* Các nút kí tự: A-Z, số, các kí tự đặc biệt …
* Các nút lệnh: Shift, Alt, ……

Tất cả các phím được sắp xếp theo tiêu chuẩn nhất định. Mặc dù các loại bàn phím khác nhau sẽ có thêm hoặc bớt các phím, thế nhưng cấu trúc của bàn phím hầu như không đổi



**Report Descriptor USB HID của bàn phím máy tính**

**Phân tích Report Descriptor Keyboard**

Để máy tính hiểu được dữ liệu chúng ta gửi lên thì Keyboard sẽ phải gửi 1 Report Descriptor, sau đó dữ liệu gửi lên phải đúng fomat của RD đó. Bạn có thể google search: ***Report Descriptor Keyboard***chúng ta sẽ có ngay kết quả

Sử dụng công cụ kiểm tra Report Descriptor ở bài trước phân tích như sau:

0x05, 0x01, // Usage Page (Generic Desktop Ctrls)

0x09, 0x06, // Usage (Keyboard)

0xA1, 0x01, // Collection (Application)

0x05, 0x07, // Usage Page (Kbrd/Keypad)

0x19, 0xE0, // Usage Minimum (0xE0)

0x29, 0xE7, // Usage Maximum (0xE7)

0x15, 0x00, // Logical Minimum (0)

0x25, 0x01, // Logical Maximum (1)

0x75, 0x01, // Report Size (1)

0x95, 0x08, // Report Count (8)

0x81, 0x02, // Input (Data,Var,Abs,No Wrap,Linear,Preferred State,No Null Position)

0x95, 0x01, // Report Count (1)

0x75, 0x08, // Report Size (8)

0x81, 0x03, // Input (Const,Var,Abs,No Wrap,Linear,Preferred State,No Null Position)

0x95, 0x05, // Report Count (5)

0x75, 0x01, // Report Size (1)

0x05, 0x08, // Usage Page (LEDs)

0x19, 0x01, // Usage Minimum (Num Lock)

0x29, 0x05, // Usage Maximum (Kana)

0x91, 0x02, // Output (Data,Var,Abs,No Wrap,Linear,Preferred State,No Null Position,Non-volatile)

0x95, 0x01, // Report Count (1)

0x75, 0x03, // Report Size (3)

0x91, 0x03, // Output (Const,Var,Abs,No Wrap,Linear,Preferred State,No Null Position,Non-volatile)

0x95, 0x06, // Report Count (6)

0x75, 0x08, // Report Size (8)

0x15, 0x00, // Logical Minimum (0)

0x25, 0x65, // Logical Maximum (101)

0x05, 0x07, // Usage Page (Kbrd/Keypad)

0x19, 0x00, // Usage Minimum (0x00)

0x29, 0x65, // Usage Maximum (0x65)

0x81, 0x00, // Input (Data,Array,Abs,No Wrap,Linear,Preferred State,No Null Position)

0xC0, // End Collection

// 63 bytes

Như vậy khi 1 phím được nhấn, bàn phím sẽ phải gửi 63 byte ( với chuột máy tính là 74 byte) để báo cho máy tính biết rằng nút nào được nhấn. Các data được gửi qua lại USB HID với Input sẽ là các nút được nhấn và Output sẽ dùng để điều khiển các đèn Led trên bàn phím (Numlock, Caplock …)

**Cấu trúc dữ liệu khi gửi của HID KeyBoard**

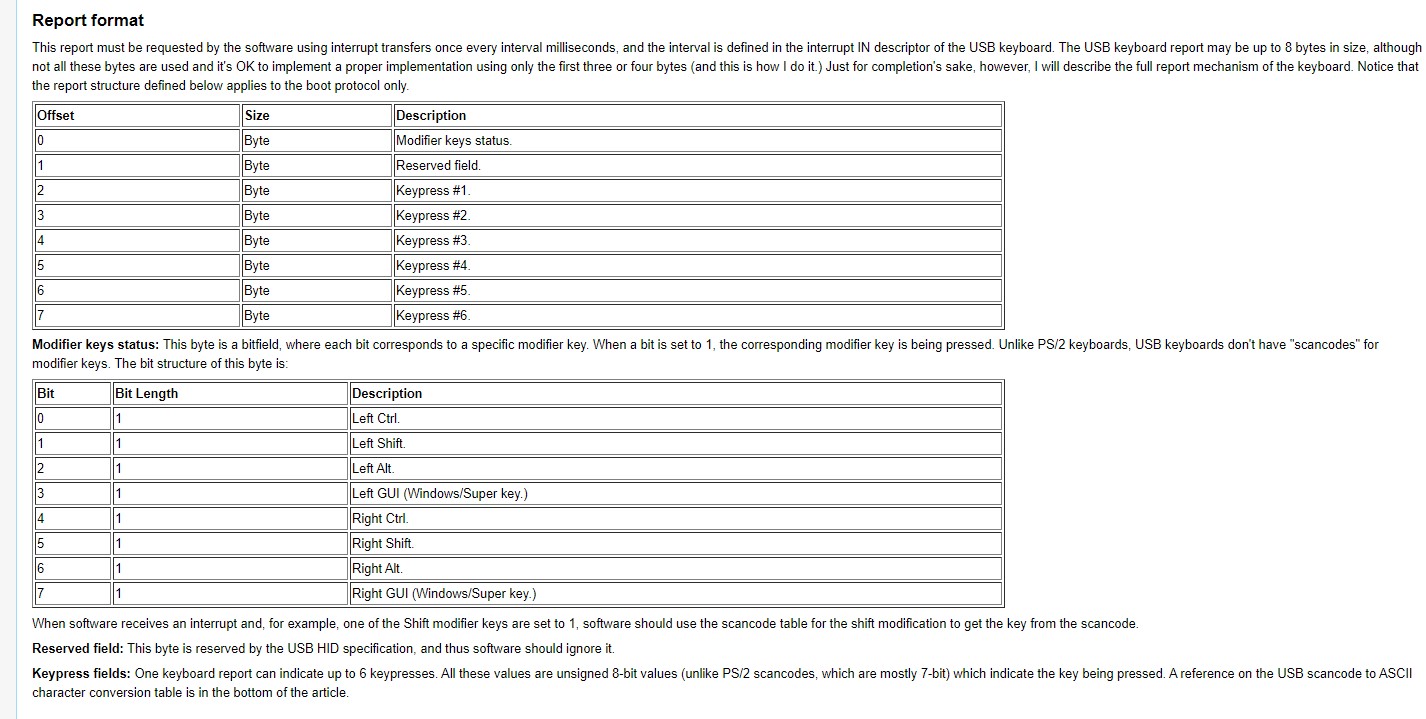
Nhìn vào Report Descriptor của Keyboard chúng ta cũng dễ dàng phân tích được dữ liệu sẽ truyền lên máy tính như thế nào.

Khi Keyboard truyền lên máy tính thì sẽ gồm tất cả Input trong Usage Page (Kbrd/Keypad)

Tống sẽ là 1 Byte + 1 Byte + 6 Byte.

Trong đó:

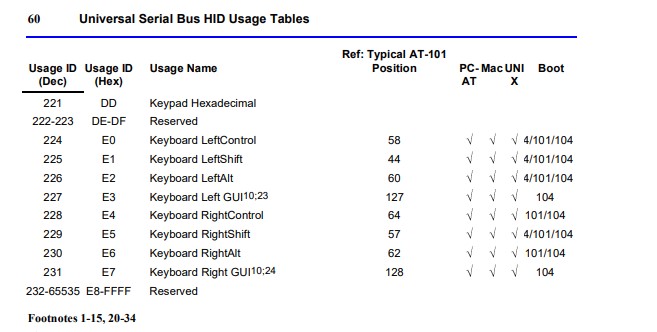
* Byte đầu tiên sẽ là **Modifier keys status**có giá trị từ 0xE0 – 0xE7: Chứa các giá trị của các phím lệnh
* Byte thứ 2 là Byte đặc tả của HID, chúng ta ko cần quan tâm
* 6 Byte còn lại là **Keypress fields**có giá trị từ 0x00 – 0x65 hay từ 0 – 101 theo hệ thập phân: Chứa các giá trị của phím kí tự



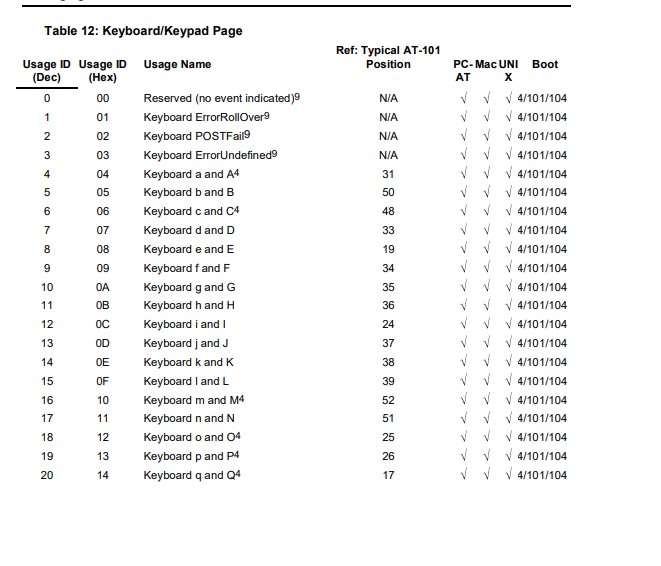
Chi tiết tham khảo tài liệu về USB HID Keyboard:<https://wiki.osdev.org/USB_Human_Interface_Devices>

Vậy làm thế nào chúng ta biết được mỗi phím tương ứng với giá trị nào. Hãy cùng tham khảo bảng sau:

Với trường **Modifier keys status**các giá trị từ E0-E7 tương ứng với nút nào được nhấn:



Với trường **Keypress fields**giá trị từ 0x00 – 0x65 tương ứng với kí tự nào được nhấn



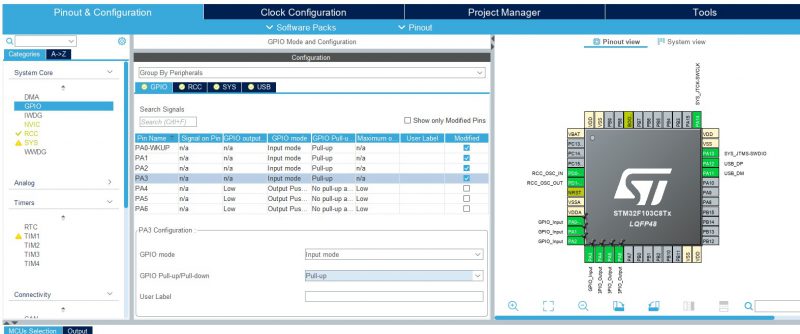
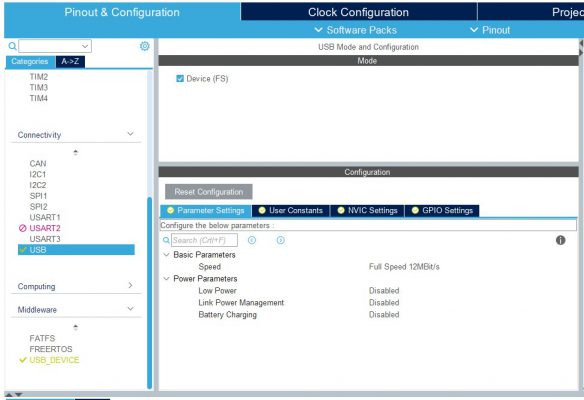
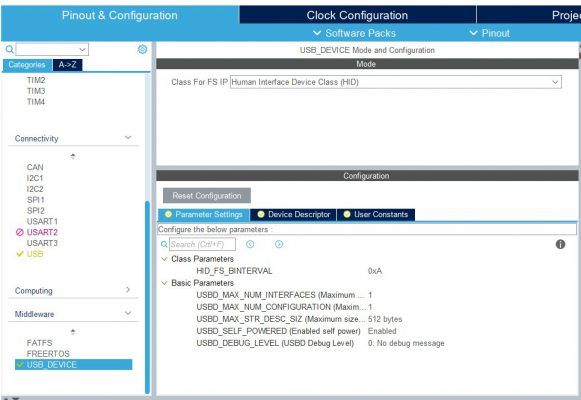
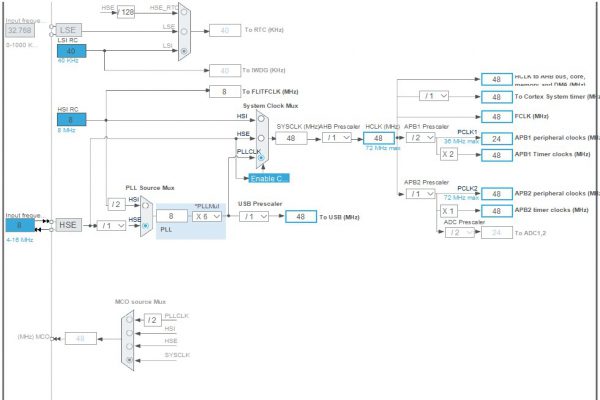
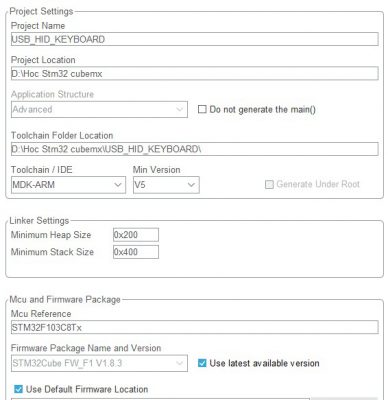
Chi tiết tham khảo tài liệu Tài liệu bảng mã kí tự trên USB HID Keyboard: <https://d1.amobbs.com/bbs_upload782111/files_47/ourdev_692986N5FAHU.pdf>

**Lập trình STM32 USB HID Keyboard bàn phím máy tính**

**Cấu hình STM32 USB HID Keyboard trên CubeMX**

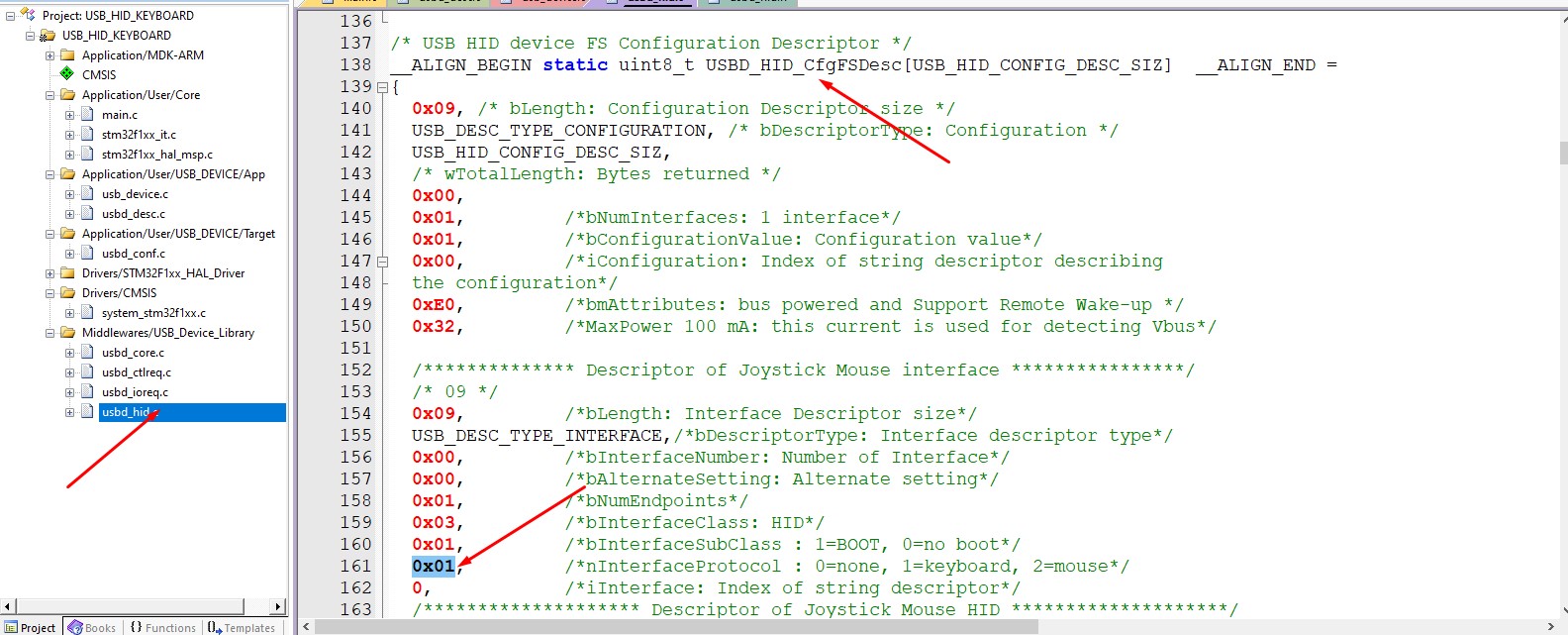
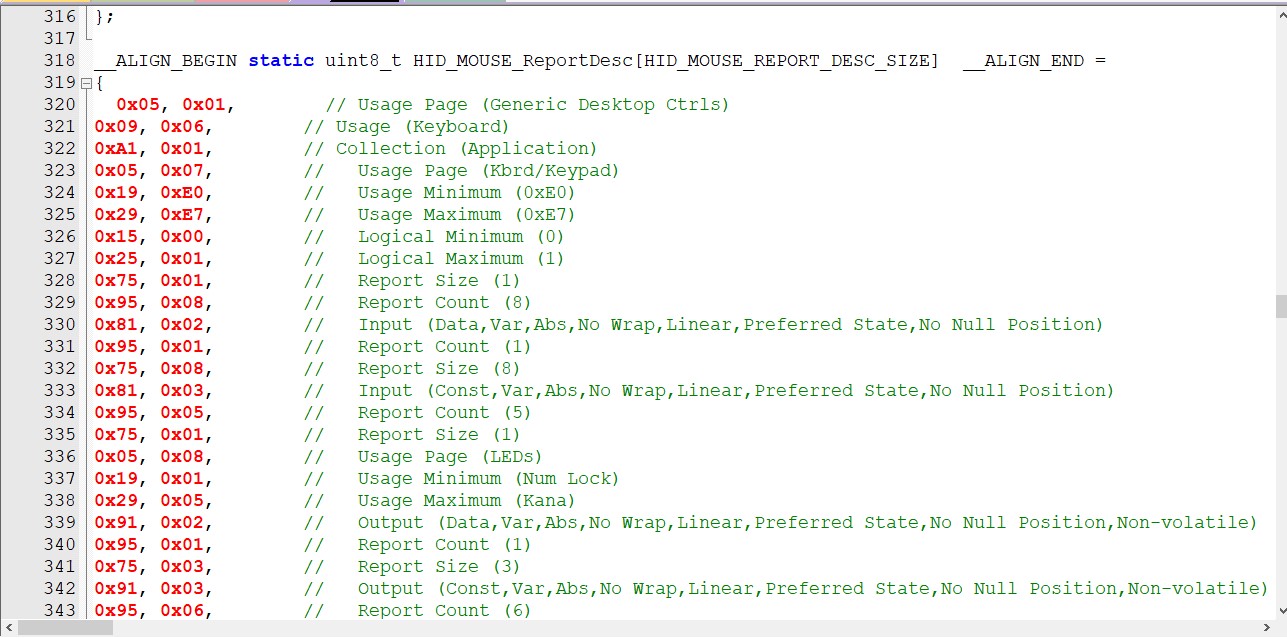
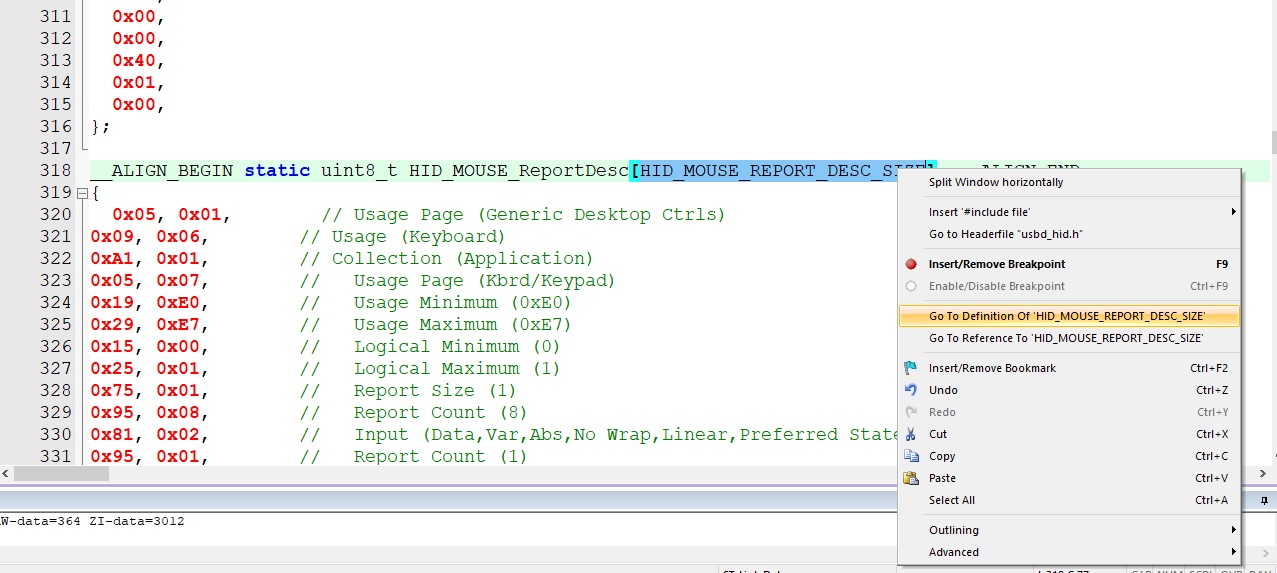
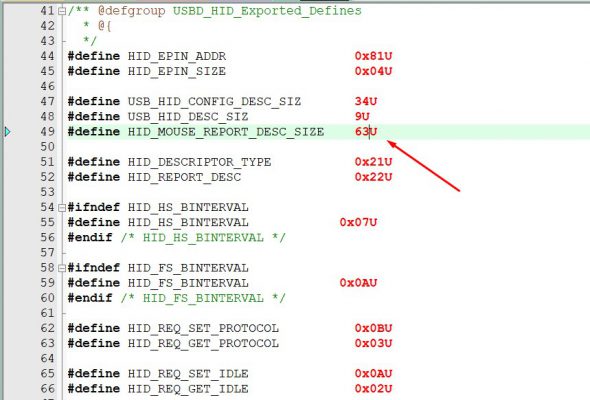
Cấu hình SYS – Debug -Serial Wire. RCC – HSE – Crystal.

Cấu hình GPIO Input – Pull Up. Cấu hình tương tự [Bài STM32 với Keyboard](https://khuenguyencreator.com/lap-trinh-stm32-quet-ma-tran-phim-keypad-3x4/)

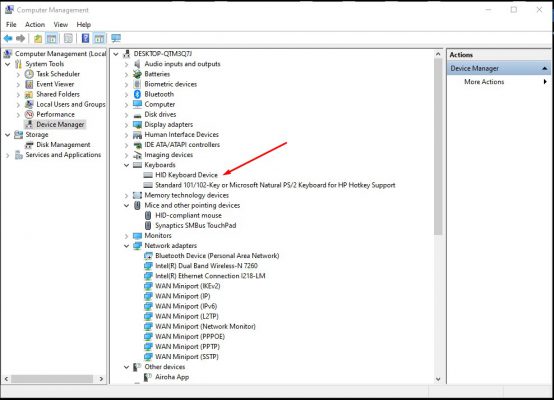
Cấu hình GPIOUSB chế độ DeviceMiddware chọn HID ClassDao động USB chọn 48MhzGen code và Open Project

**Lập trình STM32 USB HID Keyboard trên Keil C**

Sau khi mở project. Chúng ta phải cấu hình lại cho HID, bởi vì mặc đinh CubeMx Gen ra là chuột máy tính. Đầu tiên sửa interfaceprotocol.

Chỉnh Protocol thành 0x01Chỉnh Report DescriptorTìm tới nơi define kích thước ReportChỉnh lại kích thước RD là 63

Nhấn F7 Build và F8 nạp chương trình vào KIT. Sau đó cắm cổng Micro USB vào máy tính. Vào Manager trên Computer để xem thiết bị đã chuyển thanh Keyboard chưa. Các bạn nếu chưa biết làm thì đọc lại bài [STM32 HID Chuột máy tính](https://khuenguyencreator.com/lap-trinh-stm32-usb-hid-chuot-may-tinh/) nhé!

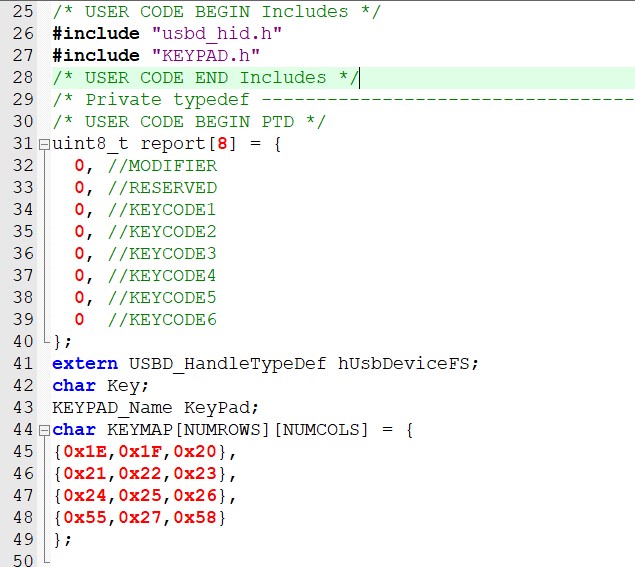


Ok, vậy là đã config xong HID. Chúng ta bắt đầu lập trình cho Keypad có thể gửi được dữ liệu lên máy tính

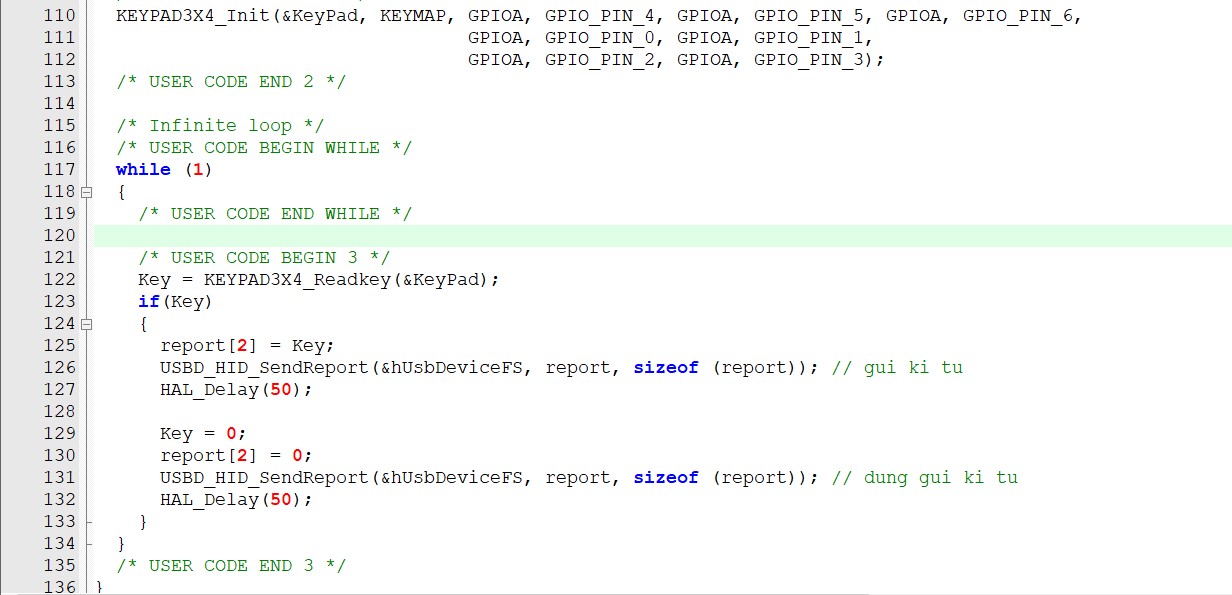
Phần Keypad các bạn down thư viện và add vào theo hướng dẫn:[STM32 Keypad 3×4](https://khuenguyencreator.com/lap-trinh-stm32-quet-ma-tran-phim-keypad-3x4/)

Trong Main, chúng ta Include Keypad và USBD\_HID vào.

Khới tạo các biến cho USB và Keypad

Thêm thư viện và khai báo các biến

Phần KEYMAP thay vì các kí tự từ 1 – 9, chúng ta sẽ đối chiếu với tài liệu bên trên, sau đó thay vào các giá trị Hex của bàn phím tương ứng với Key 1 – 9. Key # mình sẽ thay bằng Enter.



Trước While chúng ta khởi tạo Keypad. Sau đó đọc dữ liêu từ bàn phím.

Nếu phím được nhấn, copy dữ liệu vào report[2] (Keycode 1), sau đó gửi dữ liệu đi

Tiếp đến xóa dữ liệu và dừng gửi dữ liệu bằng cách send report trống.

Lưu ý: Nếu bạn không gửi report trống lên, USB HID sẽ liên tục gửi dữ liệu lên máy tính

Build và nạp vào KIT.

**Kết nối phần cứng**

PA0,1,2,3  theo thứ tự nối với các chân 2,7,6,4

PA4,5,6 theo thứ tự nối với các chân 3,1,5

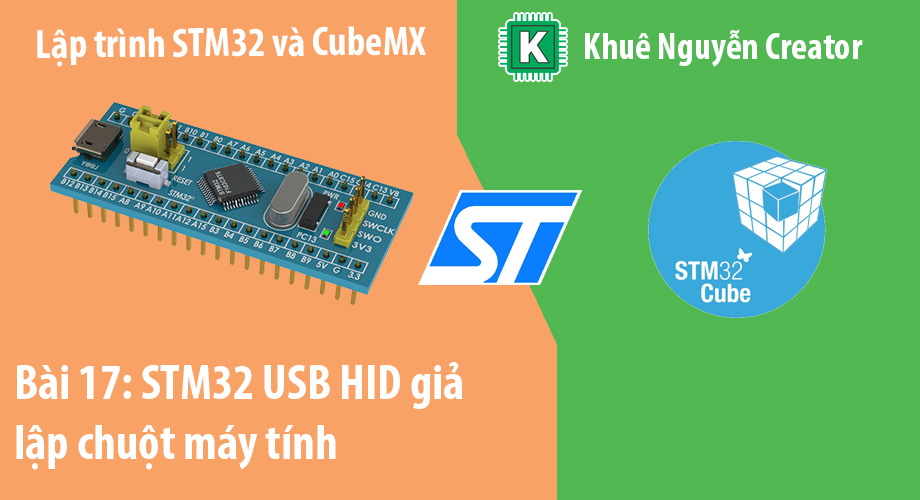
**Kết quả**

**Kết**

STM32 USB HID Keyboard khá giống với HID Mouse hay tất cả các kiểu thiết bị khác sử dụng chuẩn HID, với kiến thức ở bài này, bạn có thể dễ dàng tự làm cho mình 1 chiếc bàn phím máy tính mà không cần mua ở ngoài cửa hàng.  
Nếu thấy bài viết này có ích hãy chia sẻ cho bạn bè và cùng vào Group [Nghiện Lập Trình](https://www.facebook.com/groups/nghienlaptrinh26" \t "_blank) để kết nối với những anh em thích lập trình nhé !!!

# Bài 17: Lập trình STM32 USB HID chuột máy tính

POSTED ON [17/03/2021](https://khuenguyencreator.com/lap-trinh-stm32-usb-hid-chuot-may-tinh/) BY [KHUÊ NGUYỄN](https://khuenguyencreator.com/author/nguyenkhue2608/)

[](https://khuenguyencreator.com/lap-trinh-stm32-usb-hid-chuot-may-tinh/)

**17  
Th3**

Các thiết bị USB HID rất gần gũi với chúng ta ví dụ như chuột máy tính, bàn phím, soud card… Vi điều khiển STM32 hỗ trợ giao thức HID Device giúp chúng ta có thể lập trình tạo ra các sản phẩm giống như những thiết bị đó. Bài hôm nay mình sẽ nói đến USB HID Mouse, làm thế nào để nó có thể truyền nhận dữ liệu với máy tính, hãy cùng tìm hiểu nhé!

Bài 17 trong Serie [Học lập trình STM32 từ A tới Z](https://khuenguyencreator.com/lap-trinh-stm32-tu-a-toi-z/)

Mục Lục

[](https://khuenguyencreator.com/lap-trinh-stm32-usb-hid-chuot-may-tinh/)

* [USB HID là gì?](https://khuenguyencreator.com/lap-trinh-stm32-usb-hid-chuot-may-tinh/#USB_HID_la_gi)
* [Cách giao tiếp với tất cả các thiết bị USB HID](https://khuenguyencreator.com/lap-trinh-stm32-usb-hid-chuot-may-tinh/#Cach_giao_tiep_voi_tat_ca_cac_thiet_bi_USB_HID)
  + [Human Interface Device Class hoạt động như thế nào?](https://khuenguyencreator.com/lap-trinh-stm32-usb-hid-chuot-may-tinh/#Human_Interface_Device_Class_hoat_dong_nhu_the_nao)
  + [Cấu trúc của Report Descriptor](https://khuenguyencreator.com/lap-trinh-stm32-usb-hid-chuot-may-tinh/#Cau_truc_cua_Report_Descriptor)
* [Các tool dùng trong lập trình STM32 USB HID](https://khuenguyencreator.com/lap-trinh-stm32-usb-hid-chuot-may-tinh/#Cac_tool_dung_trong_lap_trinh_STM32_USB_HID)
* [Cách biến Kit Bluepill thành chuột máy tính với USB HID](https://khuenguyencreator.com/lap-trinh-stm32-usb-hid-chuot-may-tinh/#Cach_bien_Kit_Bluepill_thanh_chuot_may_tinh_voi_USB_HID)
* [Lập trình STM32 USB HID như 1 con chuột (Mouse) điều khiển con trỏ máy tính](https://khuenguyencreator.com/lap-trinh-stm32-usb-hid-chuot-may-tinh/#Lap_trinh_STM32_USB_HID_nhu_1_con_chuot_Mouse_dieu_khien_con_tro_may_tinh)
  + [Cấu hình STM32 USB HID trên Cube MX](https://khuenguyencreator.com/lap-trinh-stm32-usb-hid-chuot-may-tinh/#Cau_hinh_STM32_USB_HID_tren_Cube_MX)
  + [Lập trình STM32 USB HID trên Keil C](https://khuenguyencreator.com/lap-trinh-stm32-usb-hid-chuot-may-tinh/#Lap_trinh_STM32_USB_HID_tren_Keil_C)
  + [Kết nối và chạy thử USB HID](https://khuenguyencreator.com/lap-trinh-stm32-usb-hid-chuot-may-tinh/#Ket_noi_va_chay_thu_USB_HID)
  + [Tinh chỉnh code (Calibration)](https://khuenguyencreator.com/lap-trinh-stm32-usb-hid-chuot-may-tinh/#Tinh_chinh_code_Calibration)
* [Kết](https://khuenguyencreator.com/lap-trinh-stm32-usb-hid-chuot-may-tinh/#Ket)
  + [Related posts:](https://khuenguyencreator.com/lap-trinh-stm32-usb-hid-chuot-may-tinh/#Related_posts)

## USB HID là gì?

HID (viết tắt của Human Interface Device) là một tiêu chuẩn cho các thiết bị máy tính được vận hành bởi con người. Tiêu chuẩn này cho phép dễ dàng sử dụng các thiết bị này mà không cần bất kỳ phần mềm hoặc trình điều khiển bổ sung nào.

HID là một tiêu chuẩn được tạo ra nhằm đơn giản hóa quá trình cài đặt các thiết bị đầu vào thông qua từng giao thức cụ thể cho từng thiết bị như chuột, bàn phím,… Một thiết bị tuân thủ HID bao gồm “gói dữ liệu” có thể chứa tất cả các hành động của thiết bị.

Ví dụ: Bàn phím có thể có một phím để điều chỉnh âm lượng. Khi nhấn phím đó, “bộ mô tả HID” sẽ cho máy tính biết mục đích của hành động đó được lưu trữ trong các gói tin ở đâu và lệnh đó sẽ được thực thi.

## Cách giao tiếp với tất cả các thiết bị USB HID

Trước tiên nếu bạn chưa bao giờ điều khiển một thiết bị USB thì hãy quay lại Bài 16 để đọc bài viết và  tài liệu USB in a Nutshell để có cái nhìn tổng quát nhất.

Thiết bị HID và Host (máy chủ) giao tiếp với nhau qua kiểu Control Transfer ( hay Endpoint 0). Sử dụng Ngắt tại chiều IN và tùy chọn ở chiều Out. Đặc tả của lớp HID cho phép chúng có thể truyền dữ liệu ở cả tốc dộ low speed , full speed và high speed.

### Human Interface Device Class hoạt động như thế nào?

Trước khi máy chủ có thể nói chuyện với thiết bị, nó cần biết cách sử dụng hoặc ứng dụng của thiết bị này là gì? Dữ liệu của nó được tổ chức như thế nào? và Dữ liệu thực sự đo lường điều gì?

Lấy ví dụ: Nếu thiết bị của bạn là một con chuột máy tính, các nút bấm và tọa độ sẽ điều khiển Pointer trên màn hình. Sự kiện click hoặc righ click sẽ làm gì, scroll sẽ làm gì. Để tất cả các sự kiện đó được sảy ra, trình điều khiển lớp HID phải biết rõ:

* Thiết bị HID kết nối sẽ điều khiển Pointer của máy tính
* Có 3 nút trên thiết bị và khi nhấn tương ứng với các hàm của Pointer
* Có 2 byte dữ liệu tọa độ sẽ thay đổi tọa độ của Pointer

Tất cả các thông tin này sẽ được mô tả trong phần Report Descriptor. Khi trình điều khiển phân tích cú pháp của Report Descriptor nó sẽ hiểu được khi thiết bị chuột máy tính truyền dữ liệu lên, dữ liệu nào sẽ thuộc ứng dụng nào của máy tính. (Tương tự bạn phân luồng dữ liệu UART vậy).

Khi một thiết bị HID được kết nối, Host sẽ tạo ra 1 Request đó là GET\_DESCRIPTOR, sau khi hoàn tất quá trình. Chuột máy tính và máy tính sẽ giao tiếp với nhau mà ko cần thêm driver gì cả.

### Cấu trúc của Report Descriptor

Bộ Report Descriptor được mô tả bởi chuỗi các mục, các mục này mô tả dữ liệu sẽ truyền đi khi thiết bị USB HID device truyền hoặc nhận. Mỗi mục bắt đầu bằng tiền tố là 1 Byte quy định vai trò của mục và độ dài dữ liệu của nó.

Mỗi mục chia làm 3 loại thẻ chính:

* **Main**: Mô tả thực tế dữ liệu truyền đi và nơi dữ liệu được sử dụng. Các thẻ Global và Local có chức năng bổ nghĩa cho Main
* **Global**: Mô tả thuộc tính của tất cả các thẻ Main phía sau nó, cho đến khi có 1 thẻ Gobal khác xuất hiện
* **Local**: Thẻ này mô thả thuộc tính của thẻ Main phía sau nó.

Mỗi loại thẻ bao gồm một số loại chính như:

Với Main item:

* **Input**: Mô tả dữ liệu truyền từ thiết bị lên host như sự kiện nhấn nút, dữ liệu cảm biến, dữ liệu của nhà phát hành muốn gửi
* **Output**: Mô tả dữ liệu từ Host truyền về thiết bị như điều khiển led, động cơ ….
* **Feature**: Mô tả  dữ liệu được truyền đi được sử đụng để cấu hình cài đặt thiết bị như, tăng giảm tốc độ nháy của led, tốc độ động cơ …
* **Collection và End Collection**: Mỗi thiết bị HID phải có 1 bộ sưu tập ứng dụng ( Application Collection), để trình xử lý có thể biết được dữ liệu đang sử dụng trong ứng dụng nào. Ví dụ: 1 thiết bị có 3 tín năng chuột, bàn phím và nút nguồn thì phải có 3 Application Collection để phân biệt dữ liệu gửi đi và trả về.

Với Global Item:

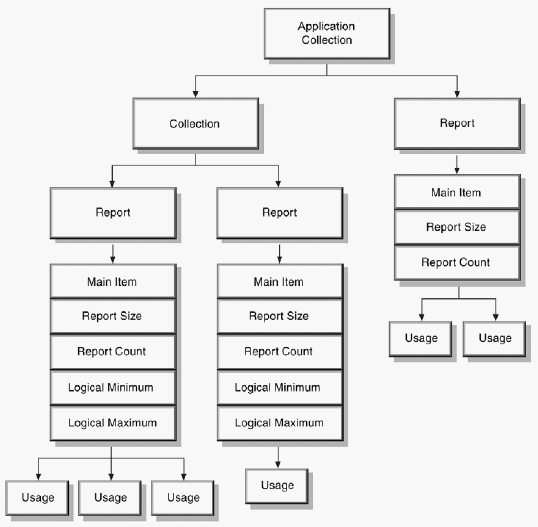
* **Usage Page**: Mô tả danh mục cao nhất của thiết bị như Generic Desktop Controls ( điều khiển thiết bị để bàn), Game control, điện thoại ….
* **Logical Minimum**: Giá trị số nguyên nhỏ nhất của Main Item
* **Logical Maximum**: Giá trị số nguyên lớn nhất của Maih Item
* **Report Size:** Kích thước của Main Item (tính theo Bit)
* **Report Count**: Số lượng Main Item

Với Local Item:

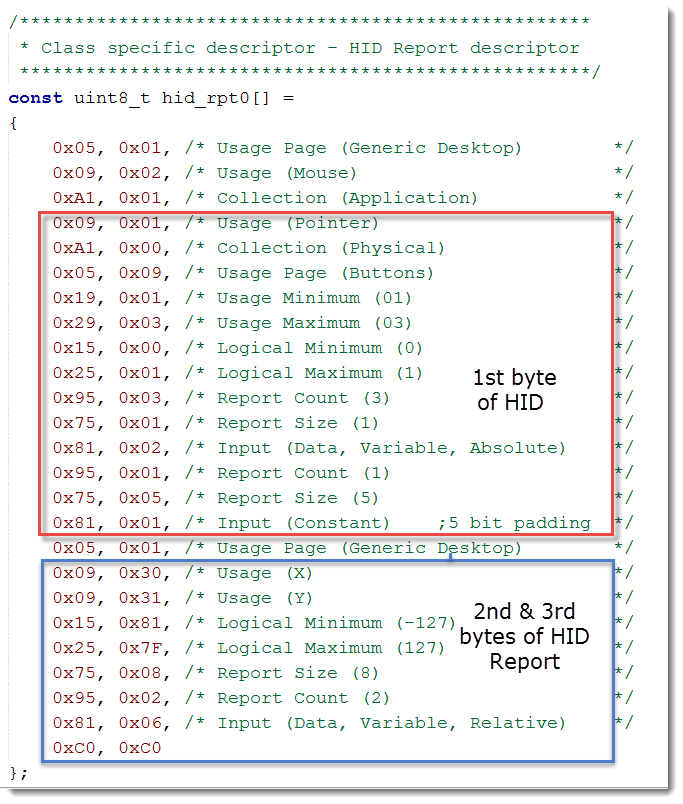
* **Usage**: Mô tả nhỏ hơn về lớp Usage Page: Ví dụ: như Usage Page là Generic Desktop Controls thì Usage có thể là *System Control hoặc Application control.*

Mỗi thẻ mục sẽ được phân loại tương ứng với 1 mã từ 0 – 255 ( 1 Byte).

Cấu trúc của Report Descriptor như sau:



Lấy ví dụ về Report Descriptor cho chuột máy tính như sau:



Phần khoanh đỏ cấu hình 3 nút nhấn của chuột, phần khoanh xanh cấu hình tọa độ của chuột.

Phần Usage Page và Usage xác định kiểu thiết bị đó là Mouse và thuộc máy tính để bàn (Generic Desktop).

Các bạn có thể tham khảo link này để phân tích 1 RD: <https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html>

## Các tool dùng trong lập trình STM32 USB HID

Phần mềm gửi và nhận dữ liệu HID Terminal:

Link download:[HID Terminal](http://vidieukhien.org/hid-terminal-phan-mem-test-giao-tiep-usb-hid.html)

Tools giải mã report descriptor.

<http://eleccelerator.com/usbdescreqparser/>

## Cách biến Kit Bluepill thành chuột máy tính với USB HID

Để sử dụng kit stm32f103c8t6 Bluepill thành chuột máy tính chúng ta cần phân tích dữ liệu truyền lên máy tính của 1 con chuột.

Cấu trúc dữ liệu truyền lên bao gồm:

* 3 phím bấm: Right, Left, Scroll với 2 mức logic 0 và 1
* 2 biến tọa độ X và Y với mức logic từ -127 đến 127

Chúng ta sẽ sử dụng Joystick đọc ADC 2 kênh X Y và nút nhấn trên nó sẽ tương ứng với phím Left Click.



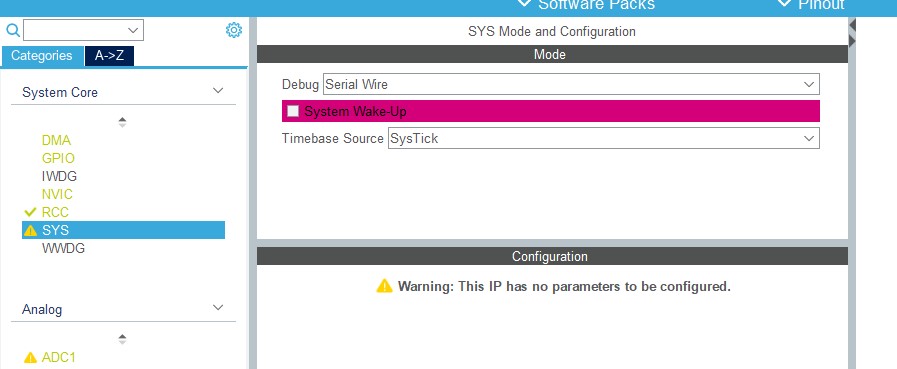
Mỗi khi có sử kiện nhấn nút hoặc di chuyển, chúng ta sẽ gửi dữ liệu theo gói thông qua USB HID.

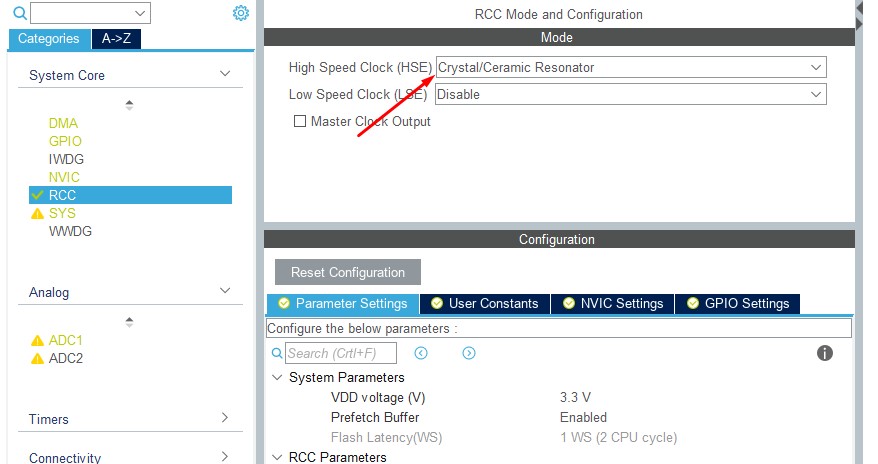
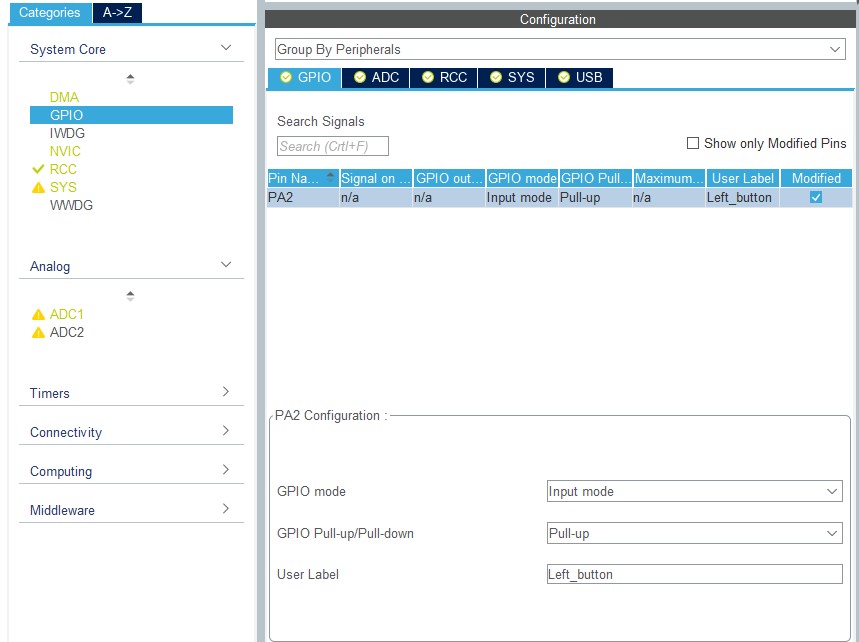
## Lập trình STM32 USB HID như 1 con chuột (Mouse) điều khiển con trỏ máy tính

Trong bài này mình sẽ sử dụng Joystick đọc giá trị ADC biểu thị trục X, Y. Và nút nhấn trên Joystick biểu thị cho Left\_Button trên chuột máy tính.

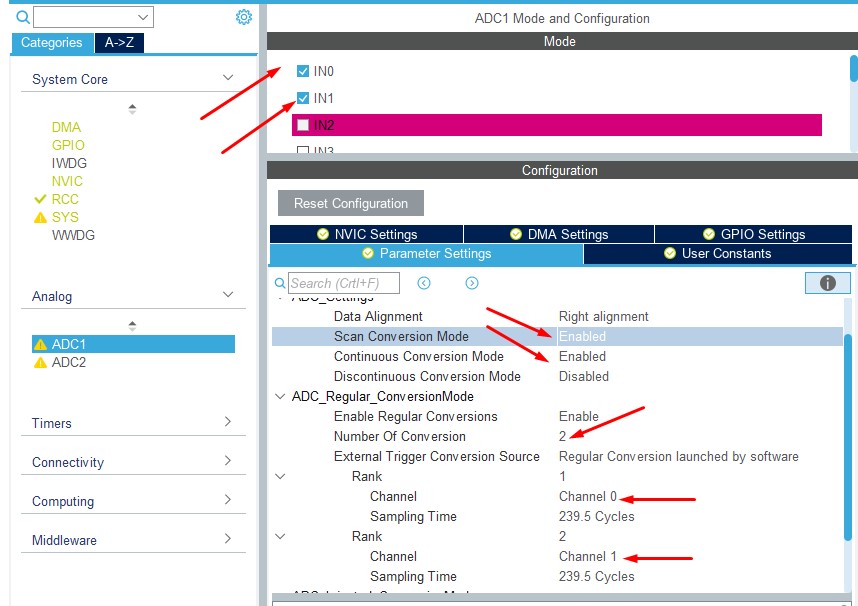
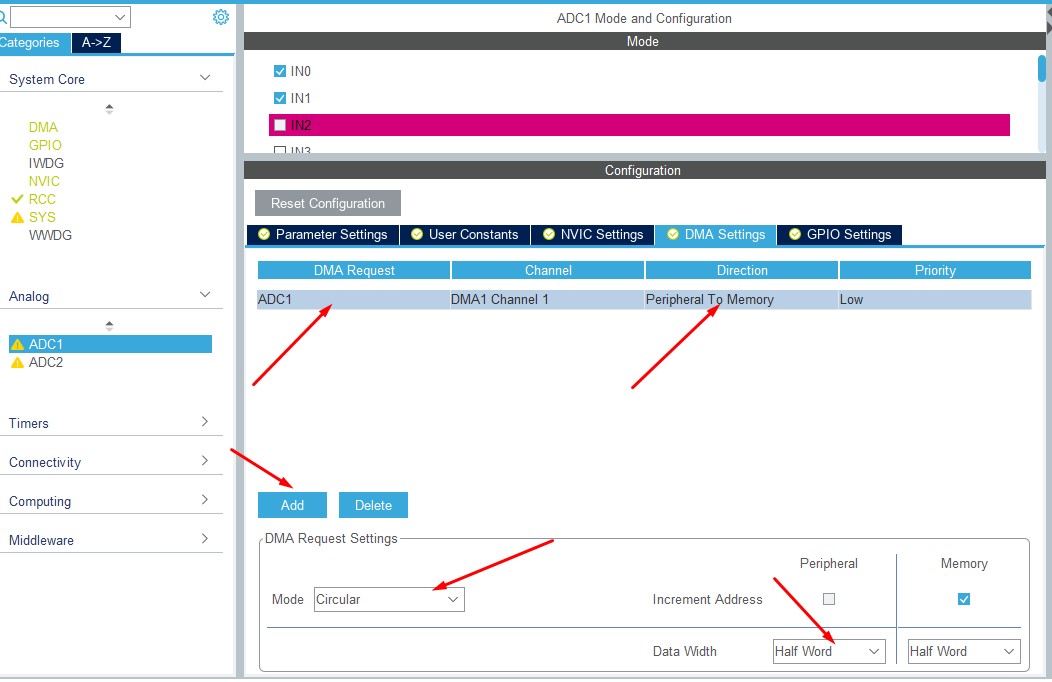
### Cấu hình STM32 USB HID trên Cube MX

Mở CubeMx, chọn chip STM32f103C8T6, trong System Core

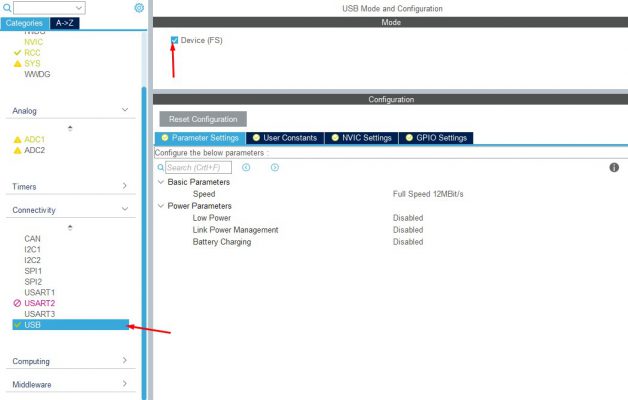
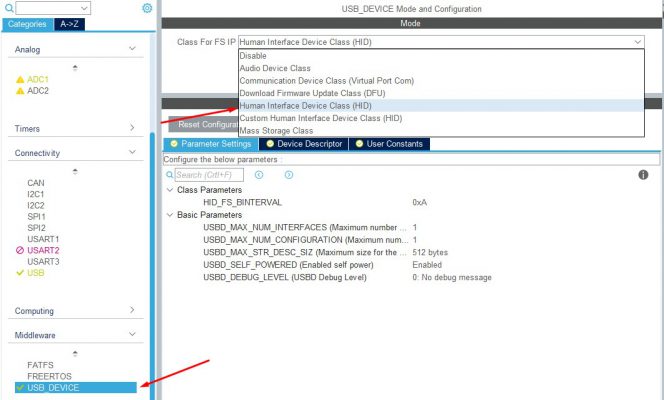
SYS chọn Debug Serial Wire: cho phép debug qua stlink

RCC chọn HSE: Thạch anh ngoạiGPIO chọn PA2 là nút nhấn Pull Up

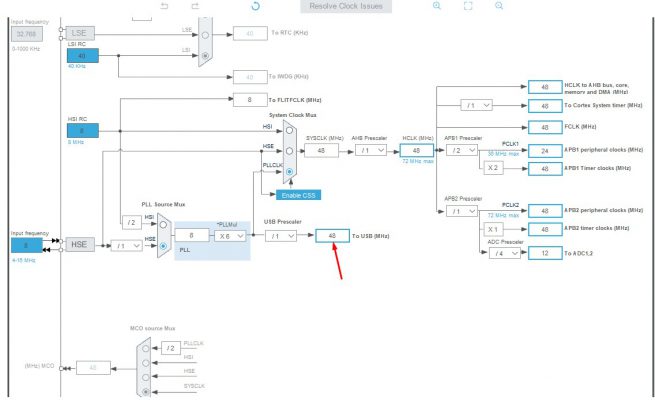
Trong Tab Analog ta sẽ cấu hình ADC1 để đọc giá trị Joystick

Chọn 2 kênh In1 và In2. Bật chế độ scan và cont. Nhớ chọn Num of Con là 2 và chỉnh các chanel về đúng 0 và 1Add thêm DMA từ ADC về Memory, chế độ Circular và Width là Half Word

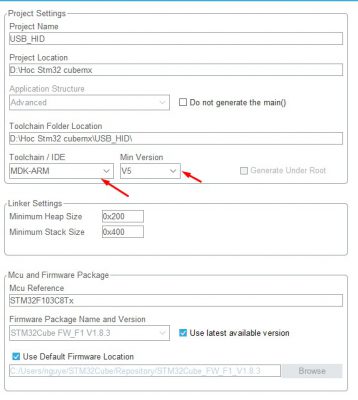
Thiết lập USB

Chọn Usb Device (FS), USB chế độ full speedChọn HID trong Class của Middleware

Thiết lập Clock cho USB là **48Mhz** (Bắt buộc)

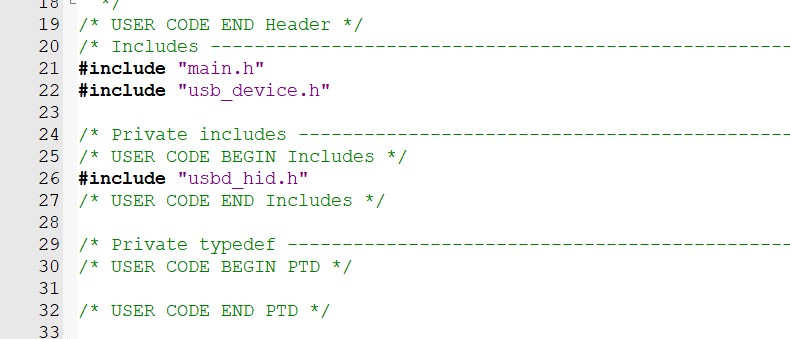


Đặt tên bài học rồi Gen code như tất cả các bài trước

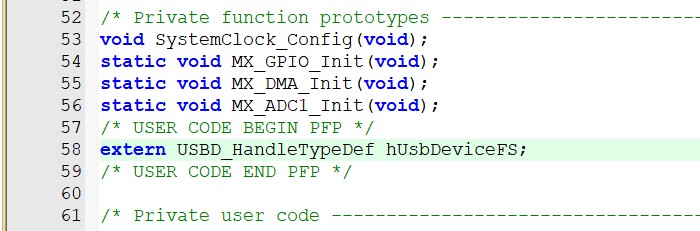


### Lập trình STM32 USB HID trên Keil C

Open bằng Keil C. Đầu tiên chúng ta sẽ thêm thư viện Hid vào main để dễ dàng thao tác bằng lệnh **#include “usbd\_hid.h”**

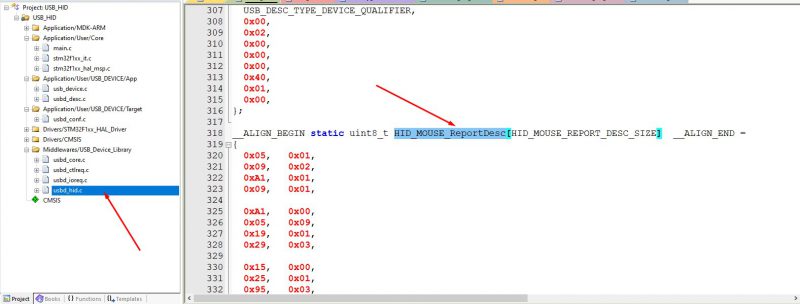


Sau đó extern biến chứa giá trị cài đặt của USB vào main

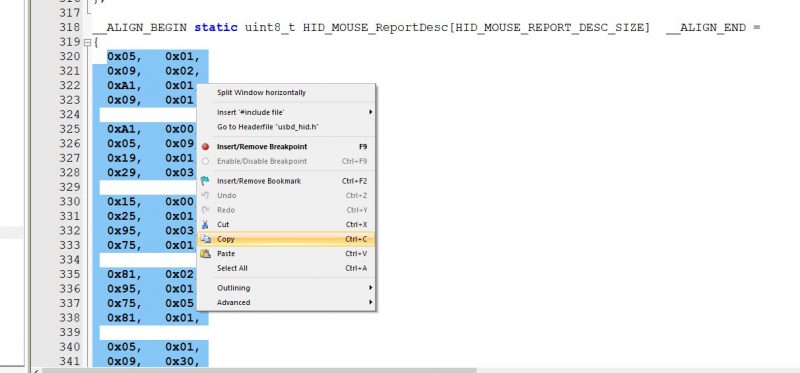


Tìm hiểu HID\_MOUSE\_ReportDesc

Khi chọn chế độ USB HID, CubeMX đã mặc định chọn thiết bị HID đó là chuột máy tính. Chúng ta cùng phân tích Report Desciptor mà CubeMx đã cho sẵn nhé. Các bạn vào USBD\_HID.c và tìm dòng code như sau:



Copy đoạn dữ liệu và paste vào công cụ phân tích mình vừa mới nêu trên: <http://eleccelerator.com/usbdescreqparser/>



Kết quả phân tích sẽ như sau:

0x05, 0x01, // Usage Page (Generic Desktop Ctrls)

0x09, 0x02, // Usage (Mouse)

0xA1, 0x01, // Collection (Application)

0x09, 0x01, // Usage (Pointer)

0xA1, 0x00, // Collection (Physical)

0x05, 0x09, // Usage Page (Button)

0x19, 0x01, // Usage Minimum (0x01)

0x29, 0x03, // Usage Maximum (0x03)

0x15, 0x00, // Logical Minimum (0)

0x25, 0x01, // Logical Maximum (1)

0x95, 0x03, // Report Count (3)

0x75, 0x01, // Report Size (1)

0x81, 0x02, // Input (Data,Var,Abs,No Wrap,Linear,Preferred State,No Null Position)

0x95, 0x01, // Report Count (1)

0x75, 0x05, // Report Size (5)

0x81, 0x01, // Input (Const,Array,Abs,No Wrap,Linear,Preferred State,No Null Position)

0x05, 0x01, // Usage Page (Generic Desktop Ctrls)

0x09, 0x30, // Usage (X)

0x09, 0x31, // Usage (Y)

0x09, 0x38, // Usage (Wheel)

0x15, 0x81, // Logical Minimum (-127)

0x25, 0x7F, // Logical Maximum (127)

0x75, 0x08, // Report Size (8)

0x95, 0x03, // Report Count (3)

0x81, 0x06, // Input (Data,Var,Rel,No Wrap,Linear,Preferred State,No Null Position)

0xC0, // End Collection

0x09, 0x3C, // Usage (Motion Wakeup)

0x05, 0xFF, // Usage Page (Reserved 0xFF)

0x09, 0x01, // Usage (0x01)

0x15, 0x00, // Logical Minimum (0)

0x25, 0x01, // Logical Maximum (1)

0x75, 0x01, // Report Size (1)

0x95, 0x02, // Report Count (2)

0xB1, 0x22, // Feature (Data,Var,Abs,No Wrap,Linear,No Preferred State,No Null Position,Non-volatile)

0x75, 0x06, // Report Size (6)

0x95, 0x01, // Report Count (1)

0xB1, 0x01, // Feature (Const,Array,Abs,No Wrap,Linear,Preferred State,No Null Position,Non-volatile)

0xC0, // End Collection

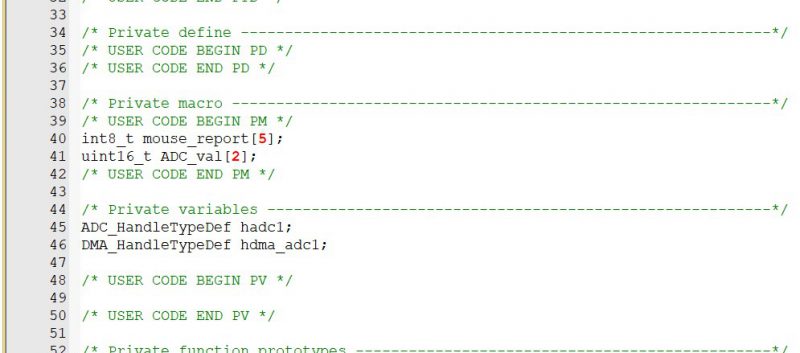
// 74 bytes

Ta thấy rằng trình tự các byte gửi như sau:

1 byte nút nhấn -> 3 byte X, Y, Whell -> 2byte Wakeup -> 1 byte kết thúc

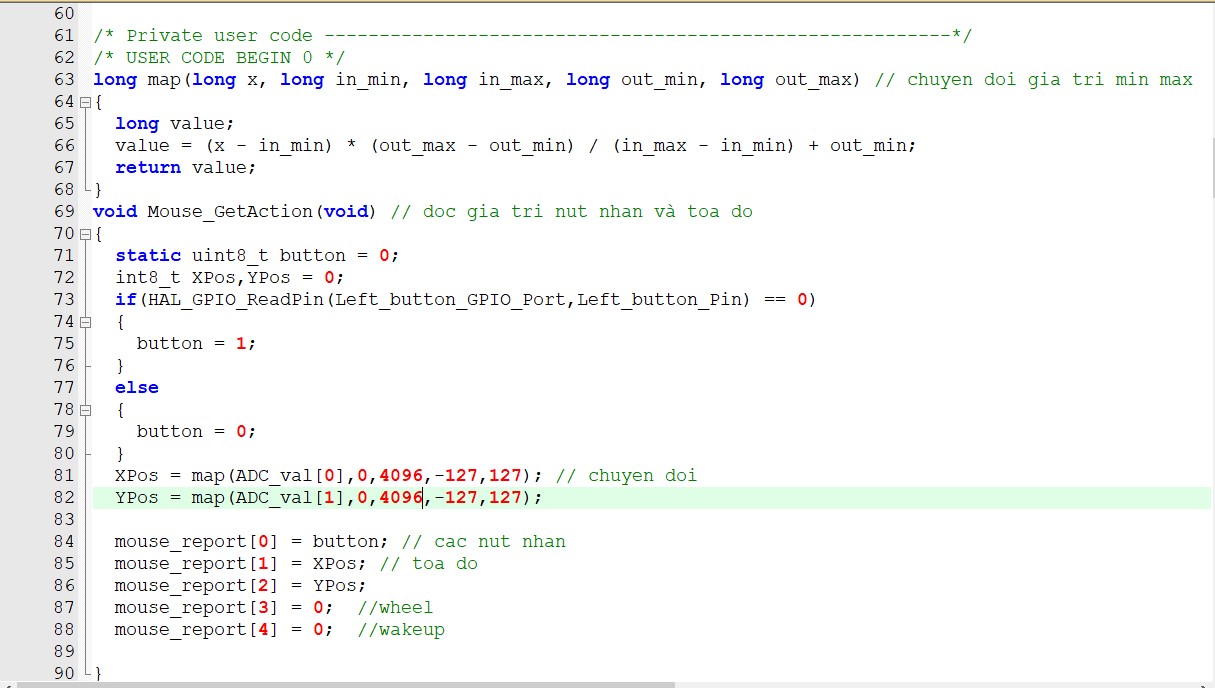
Số byte này bạn chỉ cần đếm trong các đoạn **Report Count** theo thứ tự từ trên xuống dưới. Còn ý nghĩa của các từ mình đã giải thích bên trên rồi nhé.

Vậy nên chúng ta sẽ tạo 1 mảng chứa giá trị của các byte gửi lên với 1byte kết thúc là mặc định nên ko cần thêm vào. Mình tạo mảng mouse\_report[5], và mảng lưu 2 giá trị ADC1 truyền qua DMA



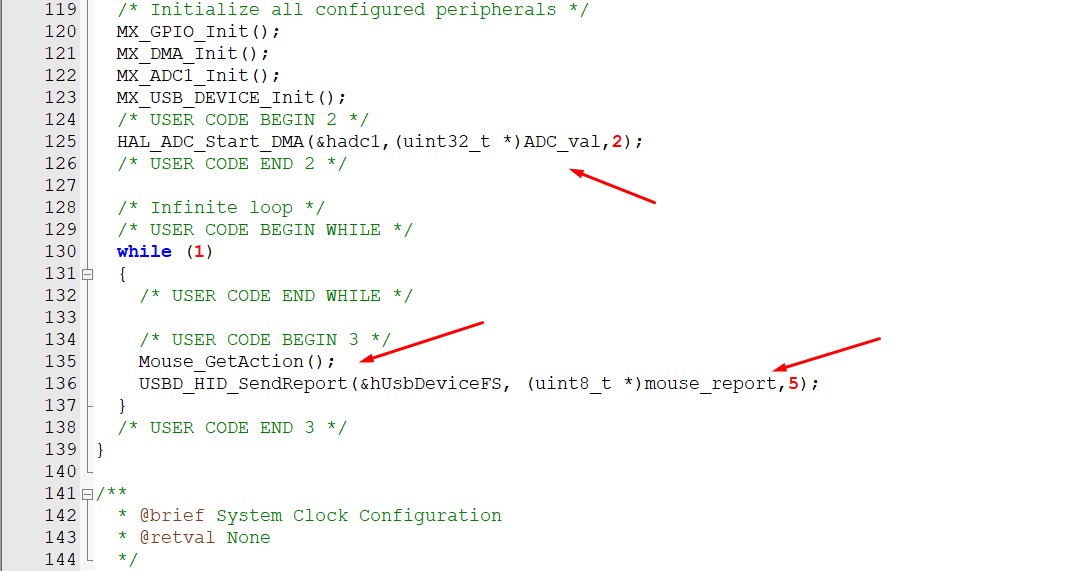
Trước Main ta sẽ lập trình như sau

Tạo một hàm tên là map để chuyển đổi giá trị ADC từ 0 – 4096 thành -127,127



Tạo hàm đọc giá trị nút nhấn, thực hiện chuyển đổi và ghi vào mảng mouse\_report

Trước While cho bắt đầu chuyển đổi ADC DMA, Trong While ta đọc giá trị của mouse và gửi qua cổng USB HID



### Kết nối và chạy thử USB HID

Nhấn F7 để Build và F8 để nạp Code vào Kit.

Kết nối Joystick vào Kit theo hướng dẫn:

**5V —> 3.3V**

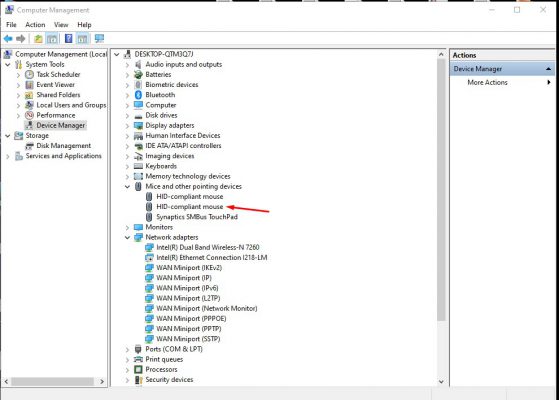
**GND —> GND**

**VRX —-> PA0**

**VRY —-> PA1**

**SW —-> PA2**

Cắm dây Micro USB vào mạch và cắm đầu còn lại vào máy tính. Mở Manager ra xem bạn sẽ thấy thêm 1 thiết bị Mouse



Mở Paint Sử dụng Joystick điều khiển thử.

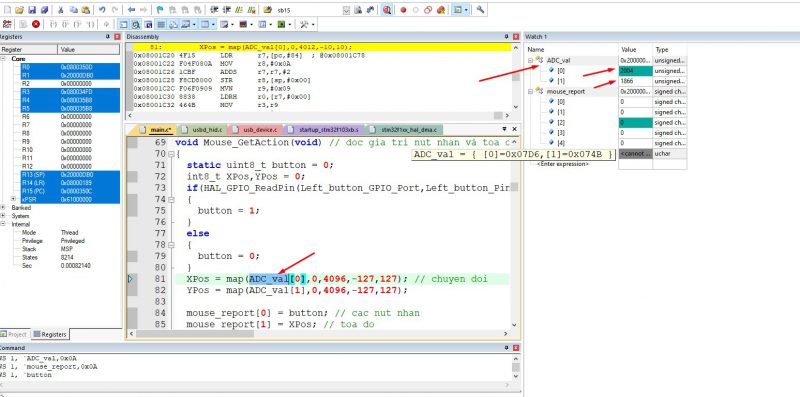
Ta thấy rằng chuột chạy quá nhanh, không thể kiếm xoát và khi không di chuyển chuột cũng vẫn tự chạy. Lý do đó là điểm cân bằng của Joystick không giống lý thuyết đó là giá trị 2048.

Và chuột chạy quá nhanh do mạch sẽ gủi các giá trị từ -127 đến 127 cực nhanh khi giữ Joystick, các tọa độ này khiến chuột di chuyển rất nhanh. Vậy làm ntn để hiệu chỉnh hai thứ đó

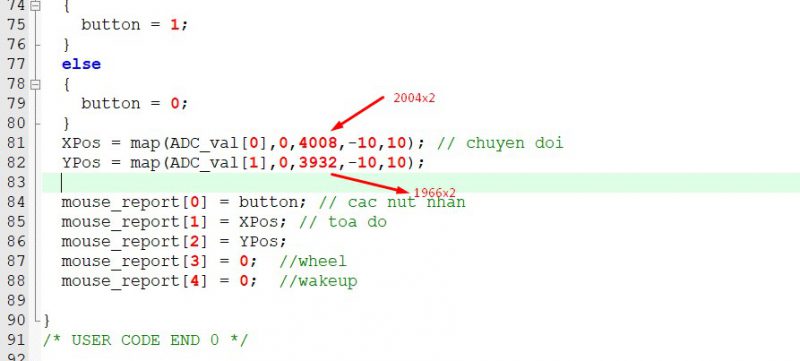
### Tinh chỉnh code (Calibration)

Rút dây cắm HID từ chuột ra, chạy chế độ debug.

Trong debug ta add giá trị ADC\_Val vào Watch 1 và nhấn chạy chương trình. Chúng ta thấy rằng: giá trị cân bằng khác nhau dẫn tới chuột luôn luôn di chuyển.



Sửa lại trong hàm Get Action, như sau:



Sửa -127, 127 thành -10 và 10 sẽ giúp chuột di chuyển chậm hơn.

Nạp và chạy thử kết quả

## Kết

STM32 USB HID được sử dụng rất rộng rãi khi muốn giao tiếp với máy tính, điện thoại, game pad một cách đơn giản nhất. Về cơ bản tất cả các device sử dụng USB HID đều làm việc giống nhau, sự khác nhau của chúng là cấu trúc gói tin truyền và đích đến sẽ được định nghĩa hết trong Report Descriptor.

**2.2. An HID Report Example analysis**

Follow is the example.

According to the format, defined in specification:

**图 2.4. HID Data Format**

|  |
| --- |
| **HID Data Format** |

Now to analysis the corresponding meaning for every group bytes:

|  |  |
| --- | --- |
| [注意] | **Data send by LSB** |
| The data is send by LSB, so when “***0x45, 0xFF***’ is sent, first send is “***0x45***”, second send “***0xFF***”, the LSB is “***0x45***” located in low address, the MSB “***0xFF***” located in high address, so the hex value is ”***0xFF45***“ |

0x06, 0x45, 0xFF,[1](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.note.06_45_FF)

0x0A, 0x00, 0xA0,[2](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.note.0A_00_A0)

0xA1, 0x01,[3](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.note.A1_01)

0x75, 0x08,[4](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.note.75_08_1)

0x96, 0x07, 0x01,[5](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.note.96_07_01)

0x15, 0x00,[6](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.note.15_00)

0x26, 0xFF, 0x00,[7](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.note.26_FF_00)

0x0A, 0x01, 0xA0,[8](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.note.0A_01_A0)

0x91, 0x02,[9](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.note.91_02)

0x75, 0x08,[10](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.note.75_08_2)

0x95, 0x08,[11](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.note.95_08)

0x0A, 0x02, 0xA0,[12](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.note.0A_02_A0)

0x81, 0x02,[13](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.note.81_02)

0xC0[14](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.note.C0)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [1](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.06_45_FF) | | **[Data]** | **[Data]** | **bTag=7:4** | **bType=3:2** | **bSize=1:0** | ***0x06*** | | --- | --- | --- | --- | --- | --- | | ***0xFF*** | ***0x45*** | 0000 | 01 | 10 | 0000 0110 | | 0xFF45 -> 0xFF00 -0xFFFF = Vendor defined | | Usage Page | Global item | 2 bytes |  | |
| [2](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.0A_00_A0) | | **[Data]** | **[Data]** | **bTag=7:4** | **bType=3:2** | **bSize=1:0** | ***0x0A*** | | --- | --- | --- | --- | --- | --- | | ***0xA0*** | ***0x00*** | 0000 | 10 | 10 | 0000 1010 | | Usage=0xA000 | | Usage | Local item | 2 bytes |  |   here Usage=0xA000 is just self defined, just need not confict with self's others, no other special meaning |
| [3](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.A1_01) | |  | **[Data]** | **bTag=7:4** | **bType=3:2** | **bSize=1:0** | ***0xA1*** | | --- | --- | --- | --- | --- | --- | |  | ***0x01*** | 1010 | 00 | 01 | 1010 0001 | |  | Application (mouse, keyboard) | Collection | Main item | 1 bytes |  | |
| [4](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.75_08_1) | |  | **[Data]** | **bTag=7:4** | **bType=3:2** | **bSize=1:0** | ***0x75*** | | --- | --- | --- | --- | --- | --- | |  | ***0x08*** | 0111 | 01 | 01 | 0111 0101 | |  | Report Size = 0x08 bits | Report Size | Global item | 1 bytes |  | |
| [5](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.96_07_01) | | **[Data]** | **[Data]** | **bTag=7:4** | **bType=3:2** | **bSize=1:0** | ***0x96*** | | --- | --- | --- | --- | --- | --- | | ***0x01*** | ***0x07*** | 1001 | 01 | 10 | 1001 0110 | | Report Count = 0x0107=263 | | Report Count | Global item | 2 bytes |  | |
| [6](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.15_00) | |  | **[Data]** | **bTag=7:4** | **bType=3:2** | **bSize=1:0** | ***0x15*** | | --- | --- | --- | --- | --- | --- | |  | ***0x00*** | 0001 | 01 | 01 | 0001 0101 | |  | Logical Minimum = 0x00 | Logical Minimum | Global item | 1 bytes |  | |
| [7](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.26_FF_00) | | **[Data]** | **[Data]** | **bTag=7:4** | **bType=3:2** | **bSize=1:0** | ***0x26*** | | --- | --- | --- | --- | --- | --- | | ***0x00*** | ***0xFF*** | 0010 | 01 | 10 | 0010 0110 | | Logical Maximum =x00FF=255 | | Logical Maximum | Global item | 2 bytes |  | |
| [8](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.0A_01_A0) | | **[Data]** | **[Data]** | **bTag=7:4** | **bType=3:2** | **bSize=1:0** | ***0x0A*** | | --- | --- | --- | --- | --- | --- | | ***0xA0*** | ***0x01*** | 0000 | 10 | 10 | 0000 1010 | | Usage = 0xA001 | | Usage | Local item | 2 bytes |  |   here Usage=0xA001 is just self defined, just need not confict with self's others, no other special meaning |
| [9](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.91_02) | |  | **[Data]** | **bTag=7:4** | **bType=3:2** | **bSize=1:0** | ***0x91*** | | --- | --- | --- | --- | --- | --- | |  | ***0x02*** | 1001 | 00 | 01 | 1001 0001 | |  | 0x02=0000 0010  Bit 1 { Variable (1)}  All other bits are Reserved (0) | Output | Main item | 1 bytes |  | |
| [10](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.75_08_2) | |  | **[Data]** | **bTag=7:4** | **bType=3:2** | **bSize=1:0** | ***0x75*** | | --- | --- | --- | --- | --- | --- | |  | ***0x08*** | 0111 | 01 | 01 | 0111 0101 | |  | Report Size = 0x08 bits | Report Size | Global item | 1 bytes |  | |
| [11](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.95_08) | |  | **[Data]** | **bTag=7:4** | **bType=3:2** | **bSize=1:0** | ***0x95*** | | --- | --- | --- | --- | --- | --- | |  | ***0x08*** | 1001 | 01 | 01 | 1001 0101 | |  | Report Count= 0x08 = 8 | Report Count | Global item | 1 bytes |  | |
| [12](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.0A_02_A0) | | **[Data]** | **[Data]** | **bTag=7:4** | **bType=3:2** | **bSize=1:0** | ***0x0A*** | | --- | --- | --- | --- | --- | --- | | ***0xA0*** | ***0x02*** | 0000 | 10 | 10 | 0000 1010 | | Usage = 0xA002 | | Usage | Local item | 2 bytes |  | |
| [13](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.81_02) | |  | **[Data]** | **bTag=7:4** | **bType=3:2** | **bSize=1:0** | ***0x81*** | | --- | --- | --- | --- | --- | --- | |  | ***0x02*** | 1000 | 00 | 01 | 1000 0001 | |  | 0x02=0000 0010  Bit 1 { Variable (1)}  All other bits are Reserved (0) | Input | Main item | 1 bytes |  | |
| [14](https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_example_analysis.html#co.C0) | |  | **[Data]** | **bTag=7:4** | **bType=3:2** | **bSize=1:0** | ***0xC0*** | | --- | --- | --- | --- | --- | --- | |  |  | 1100 | 00 | 00 | 1100 0000 | | Closes an item collection | | End Collection | Main item | 0 bytes |  | |

After analyze following data, the summary is:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Usage Page (0xFF45) | =Vendor-Specific | | 0xFF45 06 | | Global |
| Usage (0xA000) |  | | 0xA000 0A | | Local |
| Collection (Application) |  | | ***0x01 A1*** | | Main |
|  | Report Size (8) | | 0x08 75 | | Global |
|  | Report Count (263) | | 0x0107 96 | | Global |
|  | Logical Minimum (0) | | 0x00 15 | | Global |
|  | Logical Maximum (255) | | 0x00FF 26 | | Global |
|  | Usage (0xA001) | | 0xA0001 0A | | Local |
|  | Output (Variable) | | 0x02 91 | | Main |
|  | Report Size (8) | | 0x08 75 | | Global |
|  | Report Count (8) | | 0x08 95 | | Global |
|  | Usage (0xA002) | | 0xA002 0A | | Local |
|  | Input (Variable) | | 0x02 81 | | Main |
| End Collection |  | | ***0xC0*** | | Main |
| ***Report (Usage=0xA0000)*** | |  | |
|  | | ***Output Report (Usage=0xA001)*** | |
| 263 bytes | | Byte 0 | |
|  | | Byte 1 | |
|  | | ... | |
|  | | ... | |
|  | | Byte 261 | |
|  | | Byte 262 | |
|  | |  | |
|  | | ***Input Report (Usage=0xA002)*** | |
| 8 bytes | | Byte 0 | |
|  | | Byte 1 | |
|  | | ... | |
|  | | Byte 7 | |
|  | |  | |

All of above means:

1. This is a ***vendor-specific HID device***
2. input or output is relative to Host side:
   * Output = Host ⇒ Device = ***host send 263 bytes to device***
   * Input = Host ⇐ Device = ***device send 8 bytes to host***
3. Every bytes ***value range is 0~255***

### 2.1.2. Keyboard

**图 2.2. HID Report Example - keyboard - 1/2**

|  |
| --- |
| **HID Report Example - keyboard - 1/2** |

**图 2.3. HID Report Example - keyboard - 2/2**

|  |
| --- |
| **HID Report Example - keyboard - 2/2** |

## 1.3. Usage Page Summary

**图 1.9. HID Usage page summary**

|  |
| --- |
| **HID Usage page summary** |

|  |
| --- |
| HID Usage page summary |