

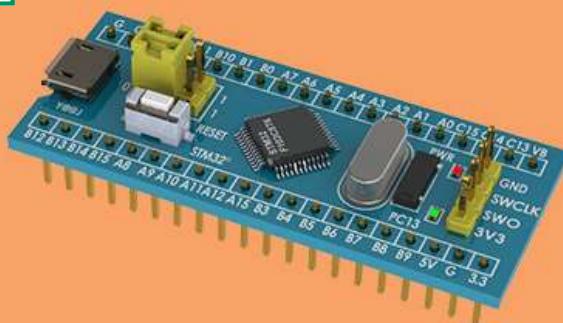
LẬP TRÌNH STM32, NGÔN NGỮ LẬP TRÌNH, THỦ THUẬT LẬP TRÌNH

Các kĩ thuật Debug chương trình nhúng với Keil C

POSTED ON 11/05/2021 BY KHUÊ NGUYỄN

11
Th5

Lập trình STM32 và CubeMX



Khuê Nguyễn Creator



Các Kĩ Thuật Debug chương trình nhúng với Keil C

Trên con đường thành công thì rất nhiều sỏi đá, còn trên con đường trở thành lập trình viên thì rất nhiều Bug. Debug là một thước đo võ công của một ông lập trình viên, càng Pro thì debug càng giỏi. Trong bài này chúng ta sẽ tìm hiểu Bug và cách Debug nhé

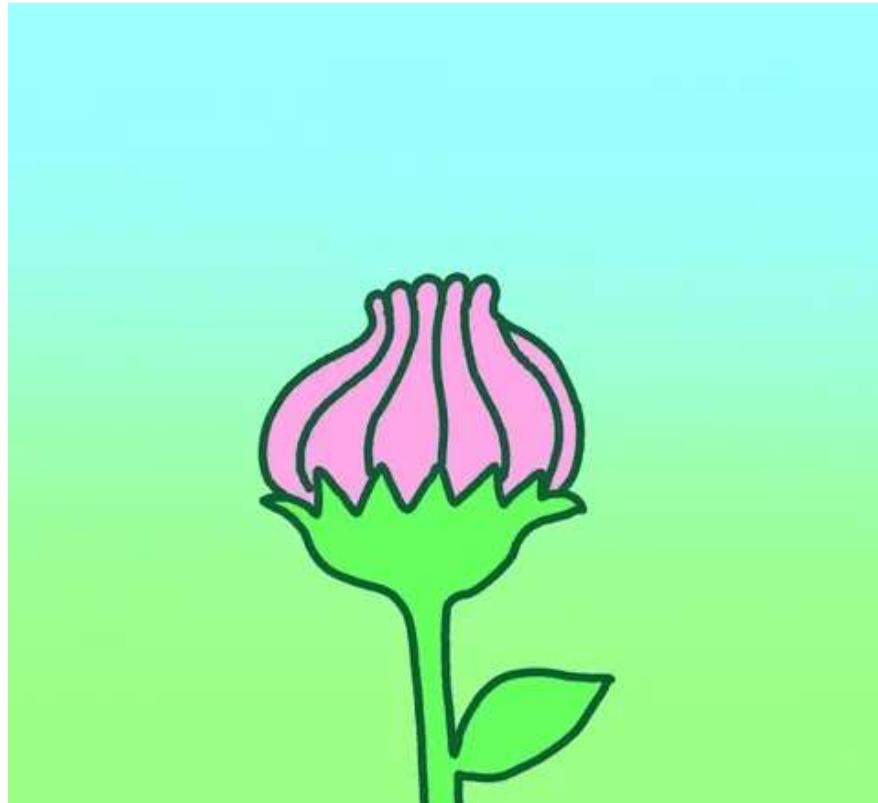
Kĩ năng nâng cao trong Serie Học lập trình từ A tới Z

Mục Lục

1. Bug là gì?
2. Debug là gì?
3. Các kĩ thuật Debug chương trình nhúng
 - 3.1. Các kĩ thuật dùng Debug Tools
 - 3.1.1. Kĩ thuật 1: Break Point và Run Step by Step
 - 3.1.2. Kĩ thuật 2: Watch Window
 - 3.2. Các kĩ thuật Debug khác
 - 3.2.1. Kĩ thuật 1: Tạo lưu đồ giải thuật
 - 3.2.2. Kĩ thuật 2: In log hay Logging (nhật ký chương trình)
 - 3.2.3. Kĩ thuật 3: Find to Define
 - 3.2.4. Kĩ thuật 4: Viết Clean Code
 - 3.2.5. Kĩ thuật 5: Vác code đi hỏi cao nhân
4. Kết
 - 4.1. Related posts:

Bug là gì?

Bug là những lỗi logic hoặc hệ thống khi chương trình chạy tạo ra, khiến nó đi sai lệch so với hướng lập trình ban đầu. Hoặc cũng có thể làm treo, chậm hệ thống ... Bug cũng giống như tên của nó, là một con Bọ săn sàng gặm nhấm, phá hủy hệ thống của bạn nếu không fix.



via GIPHY

Bugs luôn tiềm ẩn ở mọi nơi, và ta không thể lường trước được mọi tình huống có thể xảy ra mà chỉ có thể cố gắng làm giảm nó đến mức thấp nhất có thể tùy vào khả năng của ta tại thời điểm phát triển và bảo trì ứng dụng.

Đôi khi một chương trình có quá nhiều Bug mà ko thể fix kịp, lập trình viên hay nói:

| *Đó không phải là Bug, đó là tính năng*

Trong thực tế, cũng có rất nhiều sản phẩm nhà sản xuất cố tính giữ lại các Bug đó, và coi đó là 1 tính năng thú vị. Thường thấy nhất trong Game. Nhưng đa số Bug thì phải tìm và diệt và quá trình đó gọi là Debug.

Debug là gì?

Nói đơn giản Debug là quá trình tìm kiếm và tiêu diệt Bug. Công việc Debug chiếm phần lớn thời gian khi tạo ra 1 sản phẩm, thực tế code ra 1 sản phẩm không mất quá nhiều thời gian. Thế nhưng tìm diệt hết các Bug để chương trình chạy ổn định thì lại không hề đơn giản chút nào.



Bug len lỗi trong các câu lệnh của bạn, đôi khi chỉ là sai 1 dấu toán tử, sai 1 chữ trong tên biến ... nhưng hậu quả để lại thì khôn lường. Việc tìm kiếm chúng trong hàn ngàn, hoặc chục ngàn dòng lệnh không hề đơn giản chút nào.

Vậy nên mới cần các kĩ thuật Debug. Trong phạm vi bài này, chúng ta sẽ nói tới việc Debug chương trình nhúng sử dụng IDE Keil C

Các kĩ thuật Debug chương trình nhúng

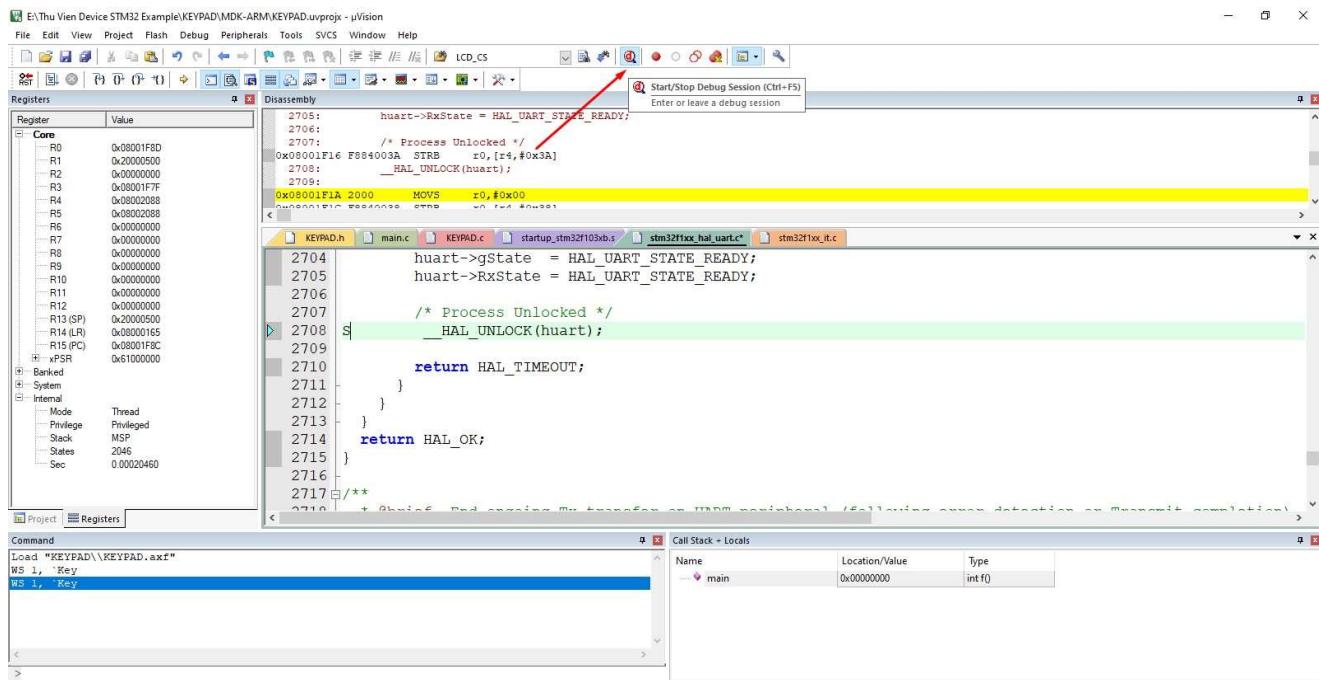
Trước hết để hiểu rõ hơn về việc debug thì các bạn nên hiểu 1 chương trình nhúng sẽ hoạt động như thế nào. Trình biên dịch hoạt động như thế nào. Bạn có thể tham khảo bài viết: Quá trình biên dịch của 1 chương trình C/C++

Trong chương trình C các câu lệnh sẽ được biên dịch theo kiểu Top Down (từ trên xuống) Các câu lệnh bên trên sẽ được thực hiện trước sau đó đến các câu lệnh bên dưới.

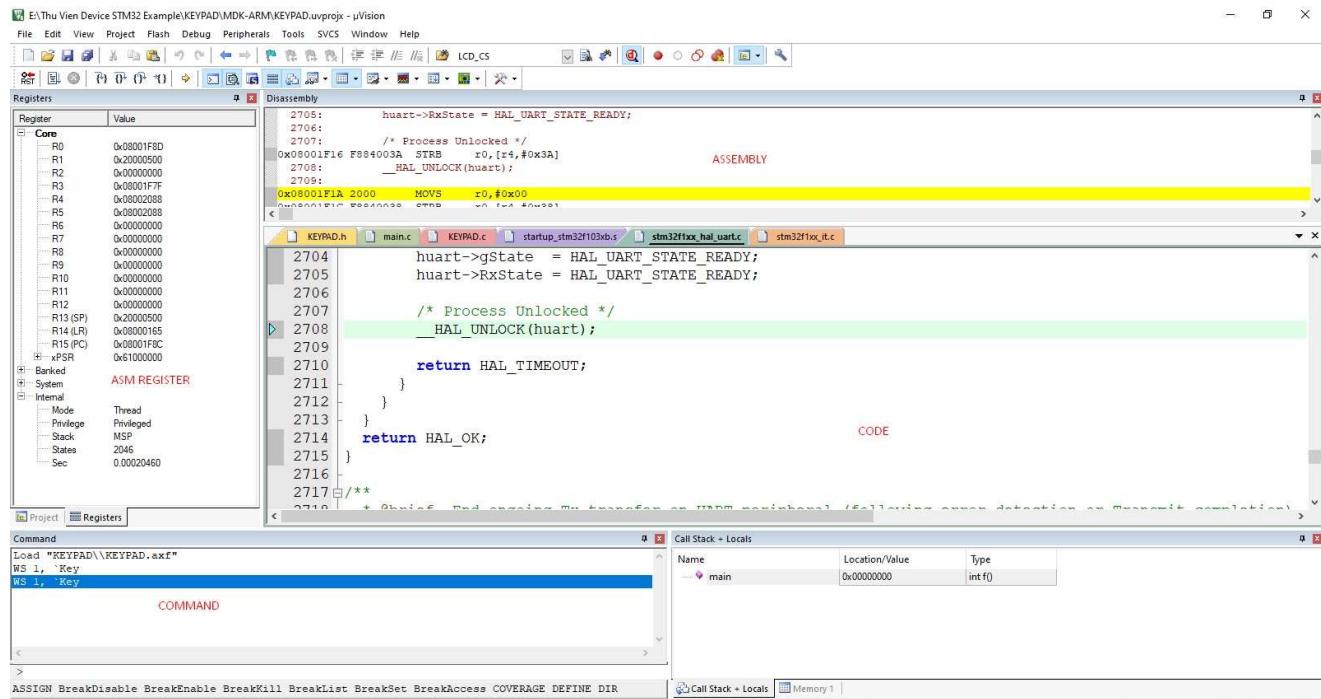
Hãy cùng tham khảo các kĩ thuật Debug này nhé

Các kĩ thuật dùng Debug Tools

Khi nhấn nút Debug trên phần mềm chúng ta sẽ được chuyển tới giao diện deBug



Giao diện bao gồm những thành phần chính sau



Kỹ thuật 1: Break Point và Run Step by Step

Chương trình luôn chạy liên tục với tốc độ cực nhanh, khi muốn dừng ta phải chọn 1 điểm dừng cho nó. Khi chương trình dừng lại, giá trị của tất cả các thanh ghi, biến, bộ nhớ đều được lưu lại tại thời điểm đó.

Cách sử dụng: Click vào điểm bên ngoài thứ tự code, hiện một hình tròn màu đỏ là được.

The screenshot shows the Keil C IDE interface. A break point is set at line 121 of the main.c file. The code at line 121 is highlighted in green: `HAL_UART_Transmit(&huart1, (uint8_t*)&Key, 1, 100);`. A red arrow points to the break point marker on the left margin of line 121. Another red arrow points to the red stop button in the toolbar.

```

111  /* Infinite loop */
112  /* USER CODE BEGIN WHILE */
113  while (1)
114  {
115      /* USER CODE END WHILE */
116
117      /* USER CODE BEGIN 3 */
118      Key = KEYPAD3X4_Readkey(&KeyPad);
119      if(Key)
120      {
121          HAL_UART_Transmit(&huart1, (uint8_t*)&Key, 1, 100);
122      }
123      HAL_Delay(50);
124  }
125  /* USER CODE END 3 */
126 }
127
128 /**
129 * @brief System Clock Configuration
130 * @retval None
131 */
132 void SystemClock_Config(void)
133 {
134     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
135     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
136 }
```

Sau khi đặt Break Point chúng ta nhấn nút Debug để vào chế độ Debug. Sử dụng công cụ Run (F5), khi chương trình chạy đến Break Point sẽ tự động dừng lại.

The screenshot shows the Keil µVision interface during debug mode. The Registers window shows the core registers (R0-R13) with their current values. The Disassembly window shows the assembly code being executed, with the current instruction highlighted in yellow. The code includes calls to `HAL_UNLOCK` and `MOVS`. The bottom window shows the source code in main.c with lines 2692-2695 visible.

Register	Value
R0	0x08001F8D
R1	0x20000500
R2	0x00000000
R3	0x08001F7F
R4	0x08002088
R5	0x08002088
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000500

Ngoài ra các bạn có thể sử dụng các chế độ Step, Step over, Step out để chạy từng bước cho chương trình. Giúp bắt lỗi tại những khoảng code nghi ngờ.

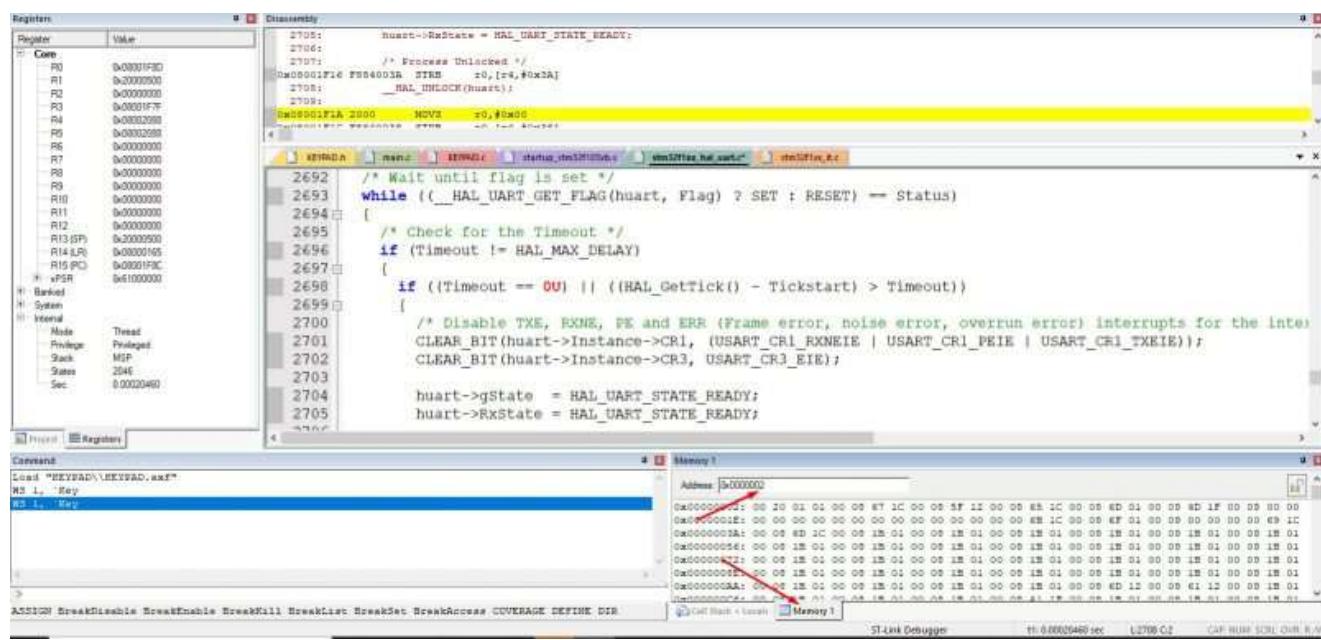
Kĩ thuật này phối hợp rất tốt với các kĩ thuật Watch sau đây.

Kĩ thuật 2: Watch Window

Chương trình hoạt động dẫn tới sự thay đổi về biến, bộ nhớ, thanh ghi ... Để có thể nhìn thấy sự thay đổi này, xem xem chúng có thay đổi đúng theo Logic không, ta sử dụng Watch.

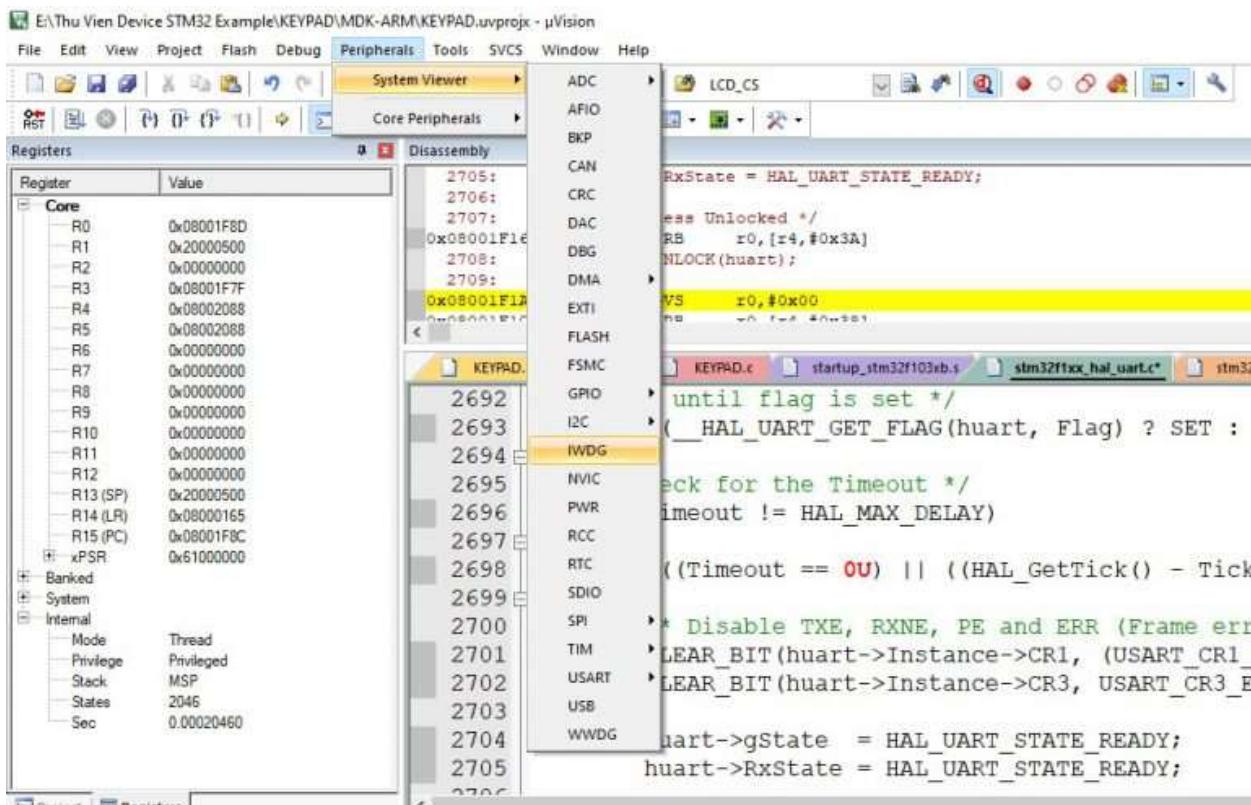
Watch Memory

Chuyển qua tab Memory, sau đó gõ địa chỉ cần view. Với công cụ này chúng ta có thể xem được trực tiếp giá trị của bộ nhớ bên trong mcu

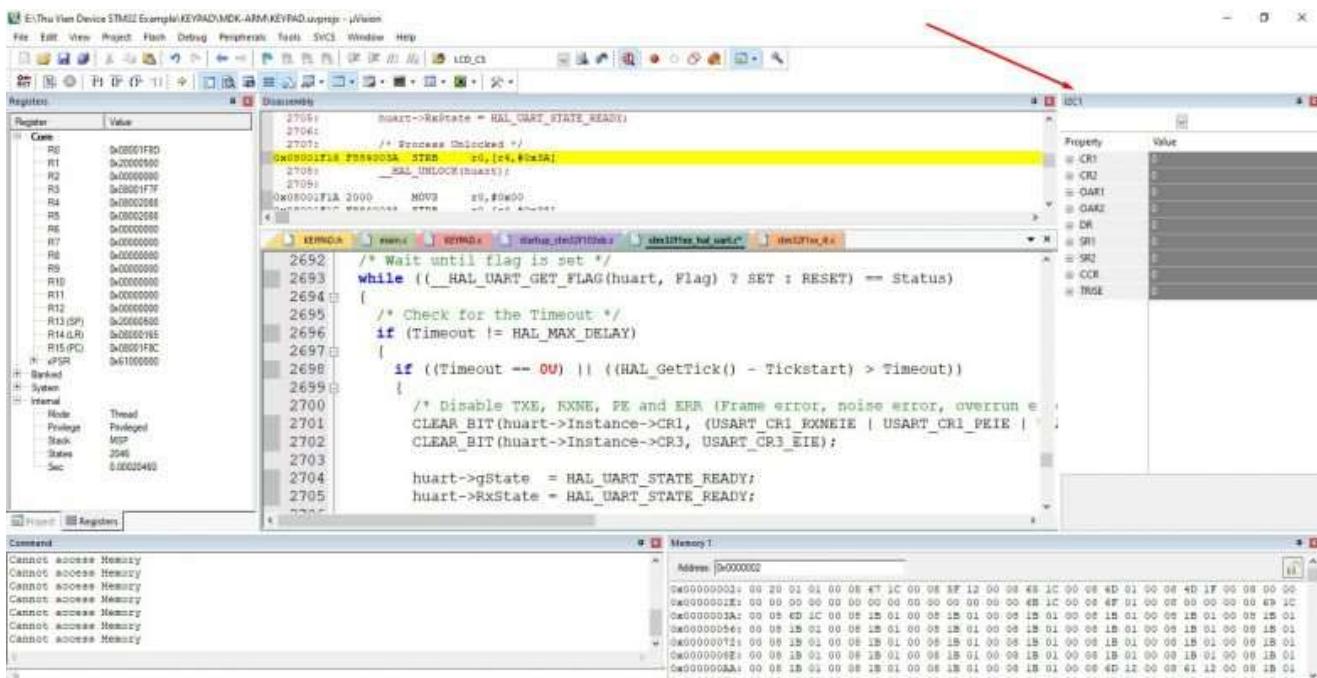


Watch Register

Xem register cũng rất quan trọng trong quá trình debug, các bạn phải nắm rõ cách ngoại vi hoạt động sau khi đổi chiều với sự thay đổi của thanh ghi mới biết rõ chúng có hoạt động đúng hay không



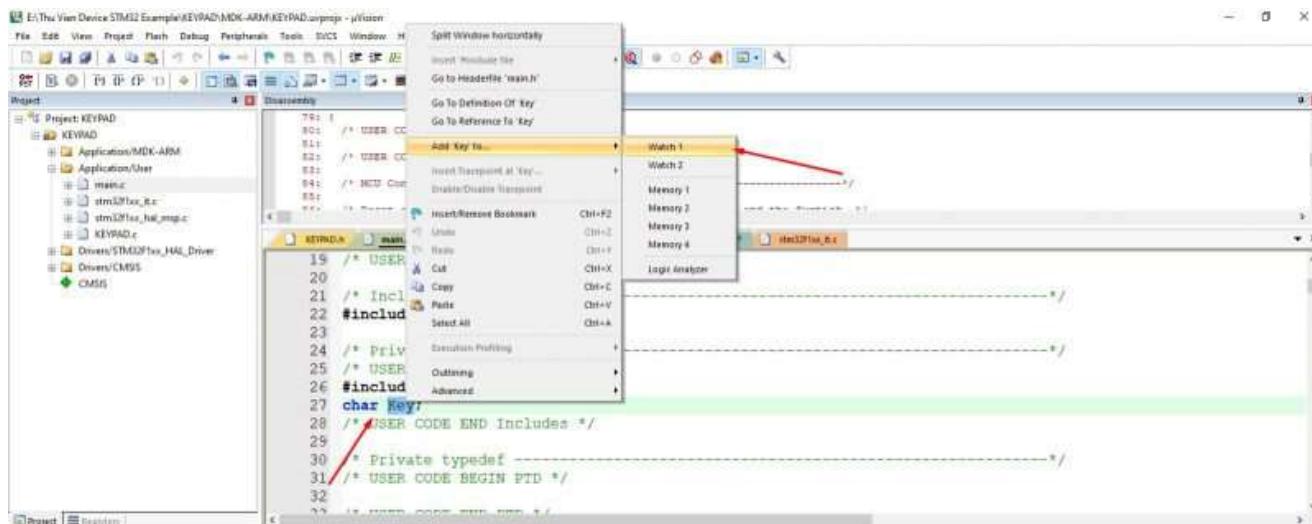
Chon Thanh Ghi



Cửa sổ xem thanh ghi

Watch Variable

Với các biến ta sử dụng cửa sổ Watch, sau đó Add biến cần xem vào.



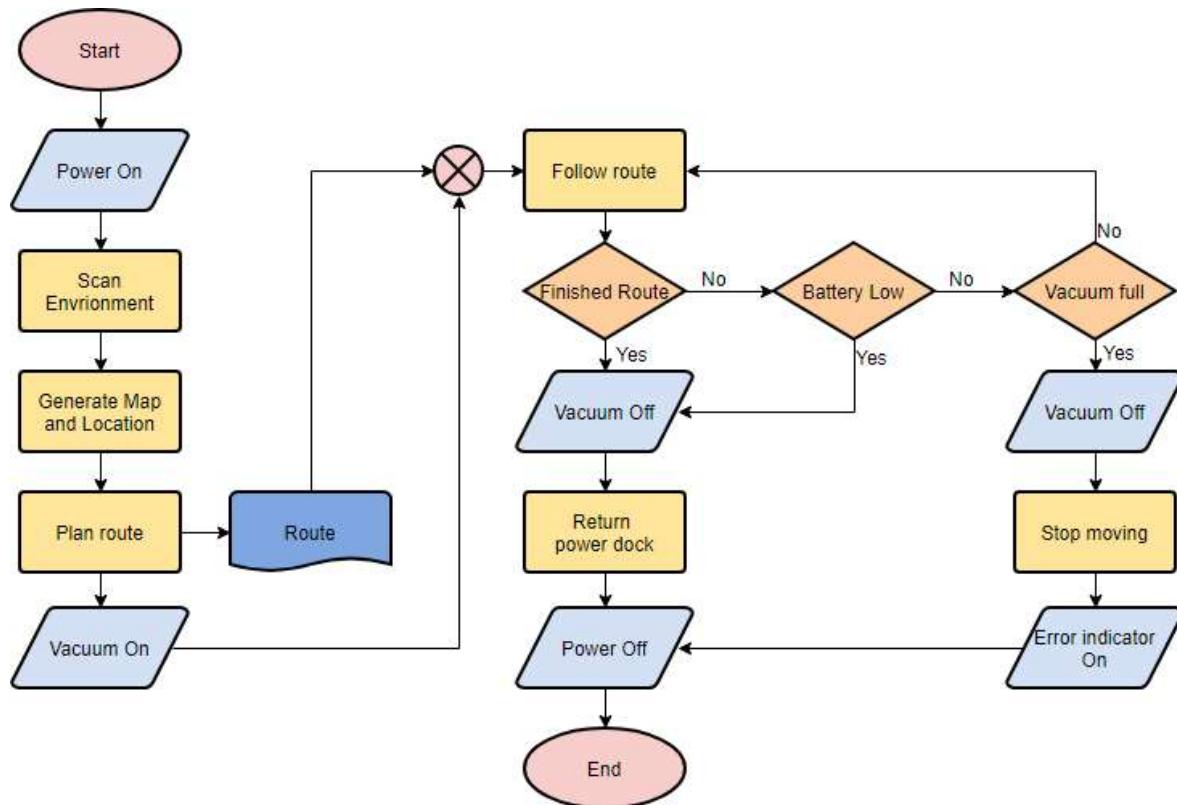
Lưu ý: Chúng ta chỉ xem được khai báo toàn cục (lưu ở phần data) chứ ko thể xem được biến cục bộ. Nếu muốn xem biến cục bộ, bạn có thể dùng phương pháp Log nhé

Các kĩ thuật Debug khác

Kĩ thuật 1: Tạo lưu đồ giải thuật

Muốn chương trình ít Bug thì Đừng tạo ra nhiều Bug ngay từ đầu

Hãy rõ ràng ngay từ khâu phát triển chương trình bằng các lưu đồ giải thuật. Và các thiết kế hệ thống như: States Machine Việc không có các tài liệu về thiết kế hệ thống sẽ tạo ra Bug ngay từ khâu thiết kế. Và các Bug này thì thường siêu to khổng lồ, có thể bạn phải đập cả hệ thống đi và xây lại đó



Kĩ thuật 2: In log hay Logging (nhật ký chương trình)

Bạn có thường thấy, khi cài đặt chương trình trên Window thường có các ô hiện lên: Coping file ..., install success, install fail ... Các dòng text này được in ra liên tục giúp bạn biết được chương trình đã cài đặt tới đâu rồi.

Trong lập trình đó gọi là Log hay kĩ thuật Logging.

Trong hoặc sau khi thực hiện một tác vụ bất kì ta có thể in ra 1 đoạn văn bản thông báo, nếu chương trình bị treo thì bạn chỉ cần tìm tới nơi in ra đoạn văn bản cuối cùng của log đó và tìm kiếm xung quanh nó.

Bạn có thể sử dụng toán tử tiền xử lý để có thể bỏ đi các đoạn log khi không cần thiết nữa.

Ví dụ:

```

#define DEBUG_MOD 1
#ifndef DEBUG_MOD
    println(" In Log tại đây");
#endif
  
```

Trong ví dụ này khi bạn bỏ dòng `#define DEBUG_MOD 1` tất cả các dòng lệnh trong `#ifdef` tới `#endif` đều bị xóa trong khi thực thi tiền xử lý (không làm nặng chương trình). cấu trúc ifdef và endif có thể dùng nhiều chỗ khác nhau trên chương trình mà chỉ cần define 1 lần.

Kĩ thuật 3: Find to Define

Tìm kiếm nơi định nghĩa: Khi bạn sử dụng 1 hàm hoặc một biến bất kì, đôi khi bạn quên nó dùng như thế nào hoặc biến đó có giá trị ra sao. Có thể sử dụng Find to Define để nhảy trực tiếp tới nơi định nghĩa nó.

Việc này giảm tối đa thời gian bạn đi tìm trong cả ngàn dòng code và vài chục file khác nhau trong 1 chương trình.

Đây là một công cụ mình rất hay sử dụng, cũng là công cụ Arduino IDE ko có (Thế nên mình không thích sử dụng Arduino IDE).

The screenshot shows a code editor window with a context menu open over a line of code. The menu is titled with the function name 'HAL_UART_Transmit'. Other options visible in the menu include 'Split Window horizontally', 'Insert #include file', 'Go to Headerfile 'main.h'', 'Insert/Remove Breakpoint' (selected), 'Enable/Disable Breakpoint', 'Go To Definition Of 'HAL_UART_Transmit'', 'Go To Reference To 'HAL_UART_Transmit'', 'Insert/Remove Bookmark', 'Undo', 'Redo', 'Cut', 'Copy', 'Paste', 'Select All', 'Outlining', and 'Advanced'.

```

99
100 /* Initialize all configured peripherals */
101 MX_GPIO_Init();
102 MX_I2C1_Init();
103 MX_USART1_UART_Init();
104 /* USER CODE BEGIN */
105 KEYPAD3X4_Init(&Keypad);
106
107 /* USER CODE END */
108
109 /* USER CODE END 2 */
110
111 /* Infinite loop */
112 /* USER CODE BEGIN */
113 while (1)
114 {
115     /* USER CODE END */
116
117     /* USER CODE BEGIN */
118     Key = KEYPAD3X4_
119     if(Key)
120     {
121         HAL_UART_Transmit(&uartHandle, (uint8_t*)&key, 1, 100);
122     }
123     HAL_Delay(50);
124 }

```

Kĩ thuật 4: Viết Clean Code

Nói cho cùng, bạn không thể nào Debug 1 mình được, vậy nên khi code Clean người khác mới có thể hiểu được bạn đang viết gì, theo đó có thể giúp bạn được. Thủ tướng tượng bạn đặt các biến toàn là a,b,c,d sau đó ngầm hiểu nó dùng làm gì, thì người sau đọc code bạn có hiểu được không?

Vậy nên hãy giữ cho mình sự sạch sẽ, để không có bọ trên người nhé

Kĩ thuật 5: Vá code đi hỏi cao nhân

Đôi khi mò mẫm trong bóng tối quá lâu cũng sẽ làm bạn nản lòng, những thứ bạn thấy thật khó khăn lại là những thứ cực kì đơn giản, mà chỉ có những người giàu kinh nghiệm mới nhìn ra được.

Chả thế mà các cao nhân đều là các bậc tiền bối, có tuổi đời và tuổi nghề lâu năm. Hãy cố gắng đi tìm người Mentor cho mình, đảm bảo việc học và làm việc sẽ rất suôn sẻ đó.

Nhưng lưu ý: Kĩ thuật này chỉ nên sử dụng khi bạn đã làm hết các kĩ thuật ở trên mà không ra được. Chả ai muốn họ bị làm phiền quá nhiều bởi những câu hỏi hiển nhiên cả. Hãy nhớ Google trước khi đi hỏi nhé!

Kết

Debug là một quá trình gian khổ, mệt mỏi. Thế nhưng kĩ năng nào cũng cần rèn luyện, hãy chăm chỉ tập luyện, rồi một ngày nào đó bạn cũng sẽ trở thành 1 Dũng Sĩ Diệt Bug. Cám ơn bạn đã đón đọc, cùng vào hội **Anh Em Nghiện Lập Trình** để cùng trao đổi nhé

4.7/5 - (3 bình chọn)

Related Posts:

1. [Bài 20: Lập trình STM32 Flash đọc, ghi và xóa dữ liệu](#)
2. [Lập trình STM32 điều khiển LCD1602 chế độ 8bit và 4bit](#)

3. [Lập trình STM32 từ A tới Z](#)
4. [Bài 7: STM32 Timer chế độ PWM](#)
5. [Bài 6: STM32 Timer chế độ Input Capture và Output Compare](#)
6. [Hướng dẫn cài đặt STM32 CubeMX và Keil C lập trình STM32](#)

**KHUÊ NGUYỄN**

Chỉ là người đam mê điện tử và lập trình. Làm được gì thì viết cho anh em xem thôi. :D

4 THOUGHTS ON “CÁC KĨ THUẬT DEBUG CHƯƠNG TRÌNH NHÚNG VỚI KEIL C”

**Hieu** says:

Trong khi chương trình đang running ở chế độ debug thì có thể stop lại ở một breakpoint mà mình click vào không? Mình sử dụng IAR thấy việc debug rất ngon lành, đang debug mà muốn stop lại dòng mong muốn thì chỉ cần click set breakpoint ở dòng đó, nhưng khi dùng Keil C thì không được.

13/05/2021 AT 5:37 CHIỀU

TRẢ LỜI

**Khuê Nguyễn** says:

Vẫn được mà bạn, nếu để chắc chắn thì nhấn stop và reset sau đó chạy lại là dc

13/05/2021 AT 8:54 CHIỀU

TRẢ LỜI



Hieu says:

Okay, thank you very much !

14/05/2021 AT 9:47 SÁNG

TRẢ LỜI



Sơn says:

Bạn có thể giới thiệu một vài chương trình dùng vẽ lưu đồ được không?

20/01/2022 AT 11:16 CHIỀU

TRẢ LỜI

Trả lời

Email của bạn sẽ không được hiển thị công khai. Các trường bắt buộc được đánh dấu *

Bình luận *

Tên *

Email *

Trang web

PHẢN HỒI

Fanpage

The screenshot shows a Facebook post from a page named 'Khuê Nguyễn Creator - Học Lập Trình Vi Điều Khiển'. The post has 2,754 likes. It includes two buttons: 'Đã thích' (Liked) and 'Chia sẻ' (Share). The post content discusses a product called vTag, which is a GPS tracking device. It mentions that the device uses WiFi, GPS, and LBS, along with NB-IOT technology. The price is listed as 990,000đ. A photo of a person wearing a pink backpack with the device attached is shown.

Khuê Nguyễn Creator - Học Lập Trình Vi Điều Khiển
khoảng một tháng trước

Lý do thời gian gần đây mình không viết bài và làm thêm gì cả là đây 😊
Chính thức ra mắt sản phẩm định vị thông minh vTag.

Đây là một sản phẩm định vị đa năng với 3 công nghệ định vị WIFI, GPS, LBS kết hợp với sóng NB-IOT dành riêng cho các sản phẩm IOT.

Chỉ với 990.000đ chúng ta đã có thể có sản phẩm để:

- Định vị trẻ em, con cái... [Xem thêm](#)

Bài viết khác

The image is a promotional graphic for a course titled 'Lập trình 8051 - AT89S52'. It features a black integrated circuit (IC) chip labeled 'MICROCHIP AT89S51' and the text 'Bài 1: Tổng quan về 8051'. To the right, there is a logo for 'Khuê Nguyễn Creator' featuring a green microchip icon with a white letter 'K' and the text 'Khuê Nguyễn Creator'. Below it is the word 'PROTEUS' in large blue letters, accompanied by its logo.

Lập trình 8051 - AT89S52

Bài 1: Tổng quan về 8051

và chip AT89S51 - 52

Tổng quan về 8051

8051 là một dòng chip nhập môn cho lập trình viên nhúng, chúng được sử...

[ĐỌC THÊM](#)

Lập trình STM32 và CubeMX



Khuê Nguyễn Creator



Lập trình STM32 HID Host giao tiếp với chuột và bàn phím

Lập trình STM32 USB HID Host giao tiếp với chuột và bàn phím máy tính

Trong bài này chúng ta sẽ cùng học STM32 HID Host, biến STM32 giống như...

[ĐỌC THÊM](#)





Lộ trình học lập trình nhúng từ A tới Z

Lộ trình học lập trình nhúng từ A tới Z

Lập trình nhúng là một ngành có cơ hội nhưng cũng đòi hỏi nhiều kiến...

3 COMMENTS

[ĐỌC THÊM](#)

Lập trình STM32 và CubeMX



Khuê Nguyễn Creator



Lập trình STM32F407 SDIO đọc dữ liệu thẻ nhớ

Lập trình STM32 SDIO đọc ghi dữ liệu vào thẻ nhớ SD card

Trong bài này chúng ta cùng học cách lập trình STM32 SDIO, một chuẩn giao...

[ĐỌC THÊM](#)

Lập trình STM32 và CubeMX



Khuê Nguyễn Creator



Lập trình STM32F407 DAC chuyển đổi số sang tương tự

Lập trình STM32 DAC tạo sóng hình Sin trên KIT STM32F407 Discovery

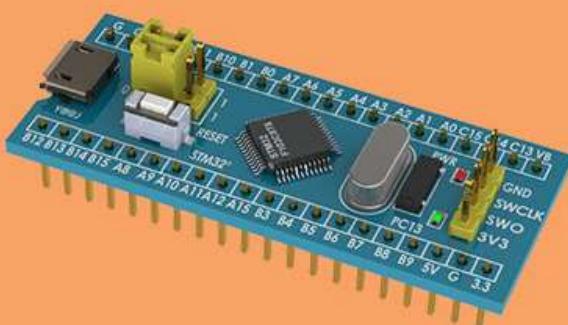
Trong bài này chúng ta sẽ cùng nhau tìm hiểu STM32 DAC với KIT STM32F407VE...

[ĐỌC THÊM](#)

Lập trình STM32 và CubeMX



Khuê Nguyễn Creator



Sử dụng hàm printf để in Log khi Debug trên STM32

Hướng dẫn sử dụng printf với STM32 Uart để in Log trên Keil C

Trong bài này chúng ta sẽ học cách retarget hàm printf của thư viện stdio...

3 COMMENTS

[ĐỌC THÊM](#)

ESP32 và Platform IO



Khuê Nguyễn Creator



Bài 9 WIFI: Lập trình ESP32 OTA nạp firmware trên Internet

Lập trình ESP32 FOTA nạp firmware qua mạng Internet với OTA Drive

Trong bài này chúng ta sẽ học cách sử dụng ESP32 FOTA (Firmware Over The...

4 COMMENTS

[ĐỌC THÊM](#)

Lập trình Nuvoton



Khuê Nguyễn Creator





Cài đặt SDC Complier và Code:Blocks IDE

Hướng dẫn cài đặt SDCC và Code::Blocks lập trình Nuvoton

Ở bài này chúng ta sẽ cài đặt các công cụ cần thiết cho việc...

ĐỌC THÊM



Blog này làm ra để lưu trữ tất cả những kiến thức, những câu chuyện của mình. Đôi khi là những ý tưởng nhất thời, đôi khi là các dự án tự mình làm. Chia sẻ cho người khác cũng là niềm vui của mình, kiến thức mỗi người là khác nhau, không hẳn quá cao siêu nhưng sẽ có lúc hữu dụng.

Liên Kết

Nhóm: Nghiên Lập Trình

Fanpage: Khuê Nguyên Creator

My Shop

Thông Tin

Tác Giả

Chính Sách Bảo Mật



Copyright 2022 © Khuê Nguyễn