**Hướng dẫn sử dụng TTOS trên dòng chip STM8S**

**Giới thiệu:**

Phiên bản này được tạo ngày: 31/05/2018.  
Viết bởi: Thanh Tùng  
E-Mail: [electricthanhtung@gmail.com](mailto:electricthanhtung@gmail.com)  
Số điện thoại: 01646936196

**Các thông tin cơ bản:**

Phiên bản TTOS này được viết trên trình biên dịch IAR cho STM8S. Hệ thống và các hàm chức năng hoạt động đúng ở tần số 16MHz. TTOS sử dụng Timer 4 có nhiệm vụ chuyển đổi ngữ cảnh giữa các tác vụ, sử dụng Timer 2 để làm định thời đếm thời gian cho chức năng Sleep hoạt động chính xác.

**Thiết lập các giá trị khởi tạo cơ bản ban đầu cho chương trình:**



Hình trên là ảnh chụp của file thư viện “TTOS\_STM8.h” trong đó:

**+ RAMSIZE** là khai báo tổng dung lượng RAM của vi điều khiển mà bạn sử dụng.

**+ STACKSIZE** là mức stack tối đa cho một tác vụ (Task). Nên đặt STACKSIZE là các bội số của 8. STACKSIZE càng lớn thì tác vụ sẽ có càng nhiều RAM để sử dụng nhưng cũng đồng thời làm giảm số lượng các tác vụ có thể đồng thời cùng chạy.

**+ TCPU** là thời gian chuyển đổi ngữ cảnh giữa các tác vụ (Cũng là chu kỳ ngắt của Timer 4) được tính theo micro giây (us).

Cài đặt Option cho IAR:

Chuột phải vào project của bạn và chọn mục “Options…” như hình dưới.



Trong mục “General Options” vào thẻ “Stack/Heap” đặt giá trị “Stack size (byte)” bằng với giá trị STACKSIZE đã khai báo. Trong hình là ví dụ cho STACKSIZE = 128.



Trong mục “C/C++ Compiler” vào thẻ “Optimezations” đặt giá trị “Number of virtual byte registers” đặt bằng 12. (xem hình).



Các cài đặt khác để sử dụng ST-Link hay tạo file HEX bạn vui lòng tự thực hiện.

**Giới thiệu các hàm chức năng của TTOS:**

**+** TTOS\_Init() hàm khởi tạo hệ thống của TTOS. Hàm này chỉ được gọi một lần và gọi sau khi đã khởi tạo xong các ngoại vi cần thiết của bạn.

+ TTOS\_StopKernel() hàm này để tạm dừng hoạt động của kernel (cấm ngắt Timer 4) làm cho các tác vụ khác không thể chiếm quyền sử dụng CPU cho đến khi kernel được tiếp tục hoạt động. Khi TTOS\_StopKernel() được gọi thì chỉ có tác vụ đã gọi có mới có thể hoạt động tiếp còn các tác vụ khác sẽ bị khóa (kể cả main).

+ TTOS\_StartKernel() hàm này để khởi động lại kernel cho kernel tiếp tục hoạt động.

+ TTOS\_SwitchTask() hàm này để chuyển quyền sử dụng CPU cho tác vụ khác khi tác vụ hiện tại không có yêu cầu sử lý gì thêm. Sử dụng TTOS\_SwitchTask hợp lý có thể tối ưu được hiệu năng sử lý của toàn hệ thống.

+ TTOS\_CreateTask(void (\*func)(), unsigned char \*arg) Hàm này để khởi chạy một tác vụ mới, nó nhận vào 2 đối số:

- Đối số func nhận vào là một con trỏ hàm, đây là hàm muốn khởi chạy thành một tác vụ mới.

- Đối số arg là một khiểu con trỏ, đây là giá trị muốn truyền vào cho tác vụ mới.

Hàm TTOS\_CreateTask trả về một giá trị kiểu Task. Đây là giá trị thể hiện của tác vụ mới khởi tạo. Có thể xem đây là một đối tượng như trong các ngôn ngữ lập trình hướng đối tượng. Kiểu giá trị Task có 2 thuộc tính là ID (kiểu unsigned char) và Func (kiểu void(\*)()). ID thể hiện mã ID duy nhất của tác vụ. Func là hàm chức năng được khởi chạy. Nếu sau khi gọi TTOS\_CreateTask mà 2 giá trị này trở về 0 có nghĩ là quá trình khởi tạo không thành công (Do không đủ RAM).

+ TTOS\_ClearTask(Task task) hàm này để hủy một tác vụ, nó nhận vào thể hiện của một tác vụ cần hủy. Hàm này trả về 1 nếu quá trình hủy thành công ngược lại trả về 0 nếu quá trình có lỗi (tác vụ không tồn tại). Chức năng này ko áp dụng cho việc hủy hoạt động của main.

+ TTOS\_This() hàm này trả về thể hiện của tác vụ hiện tại (tác vụ đã gọi nó). Kiểu trả về là kiểu Task. Chức năng này không hoạt động đúng với tác vụ main.

+ TTOS\_Sleep(unsigned int time\_ms) để tạm dừng tạm thời tác vụ đã gọi nó. Thời gian tạm dừng tính theo mini giây (ms). Hàm trả về 1 nếu thành công, trả về 0 nếu thất bại (tác vụ không được phép tạm dừng). Chức năng này không áp dụng tạm dừng hoạt động của main.

+ TTOS\_PauseTask(Task task) hàm này sẽ tạm dừng một tác vụ khi được gọi. Hàm trả về 1 nếu thành công, trả về 0 nếu thất bại (tác vụ không tồn tại hoặc tác vụ cần dừng là main).

+ TTOS\_ResumeTask(Task task) hàm này để tiếp tục lại một tác vụ đã bị dừng bới TTOS\_PauseTask hoặc đang trong trạng thái TTOS\_Sleep.

+ TTOS\_IsRunning(Task task) hàm này để kiểm tra một tác vụ có đang hoạt động tích cực hay không. Trả về 1 nếu có và ngược lại sẽ trả về 0 (tác vụ đã bị dừng bởi TTOS\_PauseTask hoặc đang trong trạng thái TTOS\_Sleep.

+ TTOS\_IsExist(Task task) hàm này để kiểm tra một tác vụ có tồn tại hay không. Trả về 1 nếu có tồn tại ngược lại sẽ trả về 0.

**Các lưu ý khi sử dụng TTOS:**

+ Sử dụng và gọi TTOS\_StopKernel()/TTOS\_StartKernel() liên tục quá nhiều lần có thể gây ảnh hưởng lớn tới các tác vụ khác. Nó giống như việc dừng Kernel trong một khoảng thời gian dài khiến các tác vụ khác bị khóa quá lâu làm trậm hoặc bị treo các tiến trình đang thực hiện.

+ Lưu ý với các tài nguyên cùng sử dụng trên nhiều tác vụ có thể gây tranh chấp tài nguyên gây hoạt động sai cho tác vụ. Đối với các tài nguyên cùng sử dụng ở nhiều tác vụ thì trước khi truy cập cần gọi TTOS\_StopKernel. Sau khi truy cập xong cần gọi lại TTOS\_StartKernel để tránh tình trạng treo cho các tác vụ khác.

Xem ví dụ sau:

void func1(){ // Nhấp nháy 1Hz trên PD0  
 TTOS\_StopKernel();  
 PD\_ODR\_ODR0 ^= 1;  
 TTOS\_StartKernel();  
 TTOS\_Sleep(500);  
}  
  
void func2(){ // Nhấp nháy 2Hz trên PD1  
 TTOS\_StopKernel();  
 PD\_ODR\_ODR1 ^= 1;  
 TTOS\_StartKernel();  
 TTOS\_Sleep(250);  
}  
  
void main(){  
 TTOS\_Init();  
 TTOS\_CreateTask(func1, 0);  
 TTOS\_CreateTask(func2, 0);  
 while(1){  
 TTOS\_SwitchTask();  
 }  
}

Trong ví dụ có 2 tác vụ được khởi tạo là func1 và func2 làm nhấp nháy LED trên PD0 và PD1, 2 tác vụ này có sử dụng chung 1 tài nguyên là thanh ghi PD\_ODR.

+ Các hàm TTOS\_ClearTask(), TTOS\_TTOS\_Sleep(), TTOS\_This(), TTOS\_PauseTask(), không hoạt động được với tác vụ main. Tác vụ main không thể hủy và cũng không thể tạm dừng nhưng có thể sử dụng TTOS\_SwitchTask() để tiết kiệm tài nguyên CPU cho hệ thống.

**Trong bộ thư viện còn cung cấp thêm thư Timer mềm chạy trên nền TTOS bạn đọc vui lòng đọc ví dụ để biết cách sử dụng.**

**Tài liệu hướng dẫn này còn nhiều thiếu sót mong bạn đọc thông cảm.**

**Mọi ý kiến đóng góp, thắc mắc hay phản hồi về lỗi phát sinh xin vui lòng liên hệ với các đường dẫn bên trên. Cám ơn bạn đã quan tâm đến dự án.**