用 C 和汇编混合编程实现协程多任务方法
(无标号一步跳转)
用于 ARM Cortex-M

C-example （适用于：KEIL / IAR / CooCox+GCC）
CR_ASM_KIC.c

```
/*---------------------------------------------------------------------------*/
/* Nuvoton M051(ARM Cortex-M0)                                               */
/* by fy_zhu , 2013-01                                                       */
/*---------------------------------------------------------------------------*/
#include "M051Series.h"
#include "CR_asm.h"

//---------------------------------------------------------------------------
// Function PROTOTYPES
//---------------------------------------------------------------------------
uint32_t TASK_1 (struct cr *cr);
uint32_t TASK_2 (struct cr *cr);
struct cr cr1, cr2;
…

/*---------------------------------------------------------------------------*/
/* main function                                                             */
/*---------------------------------------------------------------------------*/
int main (void)
{
    CR_INIT(&cr1);
    CR_INIT(&cr2);
    …                   //SYS_CLK , IO_PORT Init
    while(1)
    {
     TASK_1(&cr1);
     TASK_2(&cr2);
    }
}

uint32_t TASK_1 (struct cr *cr)
{
    CR_START(cr->lc);
while (1) {
    CR_WAIT_UNTIL(cr, cond1);
    dosomething11();
    CR_YIELD(cr);
    dosomething12();
    }
    CR_END(cr);         //warning: statement is unreachable
}

uint32_t TASK_2 (struct cr *cr)
{
    CR_START(cr->lc);
while (1) {
    CR_WAIT_UNTIL(cr, cond2);
    Dosomething21();
    CR_YIELD(cr);
    Dosomething22();
    }
    CR_END(cr);         //warning: statement is unreachable
}
```

H 文件 （适用于：KEIL / IAR / CooCox+GCC）

CR_asm.h

```
/*-------------------------------------------------------------------------*/
/* CR_asm.h                                                                 */
/* by fy_zhu , 2013-01                                                     */
/*-------------------------------------------------------------------------*/
typedef uint32_t lc_t;
struct cr {
                lc_t lc;
            };
#define CR_WAITING 0
#define CR_YIELDED 1
#define CR_EXITED   2
#define CR_ENDED   3
#define CR_LEAVESETTING     127

void *quit_addr;
uint32_t retcode_temp;

extern void yield(lc_t *lc);
extern void way_out(void);
extern void resume_lc(lc_t lc);
```

```
#define CR_INIT(cr)     (cr)->lc=0

#define CR_START(lc)         \
    do { \
          { retcode_temp=CR_LEAVESETTING; way_out(); } \
          if (retcode_temp != CR_LEAVESETTING) { return retcode_temp; } \
          if (lc != 0) { resume_lc(lc); } \
        }while(0)

#define CR_WAIT_UNTIL(cr, condition)    \
        while(!(condition)) { retcode_temp=CR_WAITING; yield(&(cr->lc)); }

#define CR_YIELD(cr)       { retcode_temp=CR_YIELDED; yield(&(cr->lc)); }

#define CR_END(cr)       { (cr)->lc = 0; return CR_ENDED; }
```

汇编语言文件 （适用于：KEIL）

CR_func_K.s

```
;/*------------------------------------------------------------------------*/
;/* CR_func_K.s                                                            */
;/* for KEIL                                                               */
;/* by fy_zhu , 2013-01                                                    */
;/*------------------------------------------------------------------------*/

    AREA TEST, CODE, READONLY

    EXPORT     way_out
    EXPORT     resume_lc
    EXPORT     yield

    IMPORT     quit_addr

;void resume_lc(lc_t lc);
resume_lc PROC
    ;r0=cr->lc
    bx     r0;
    ENDP

;void yield(lc_t *lc);
yield PROC
    ;r0=&(cr->lc)
    mov   r1, lr ;
    str    r1, [r0];
    ldr    r0, =quit_addr;
    ldr   r0, [r0]         ;!!!
    bx    r0;
    ENDP

;void way_out(void);
way_out PROC
    push {r0, r1, lr};
    mov  r0, lr;
    ldr   r1, =quit_addr;
    str   r0, [r1];
    pop   {r0, r1, pc};  ;//    bx     lr ;
    ENDP

    nop                        ;Add 2 bytes of padding
    END
```

汇编语言文件 （适用于：IAR）

CR_func_I.s

```
;/*---------------------------------------------------------------------------*/
;/* CR_func_I.s                                                               */
;/* for IAR                                                                   */
;/* by fy_zhu , 2013-01                                                       */
;/*---------------------------------------------------------------------------*/

    EXPORT      way_out
    EXPORT      resume_lc
    EXPORT      yield

    IMPORT     quit_addr

    NAME CR_FUNC_I

;void resume_lc(lc_t lc);
        SECTION CR_CODE : CODE
    CODE16
resume_lc:
    ;r0=cr->lc
    bx     r0;
;    end of resume_lc
```

```
;void yield(lc_t *lc);
        SECTION CR_CODE : CODE
    CODE16
yield:
    ;r0=&(cr->lc)
    mov   r1, lr ;
    str   r1, [r0, #0];
    ldr   r0, =quit_addr;
    ldr   r0, [r0, #0]          ;!!!
    bx     r0;
;   end of yield

;void way_out(void);
        SECTION CR_CODE : CODE
    CODE16
way_out:
    push {r0, r1, lr};
    mov   r0, lr;
    ldr   r1, =quit_addr;
    str   r0, [r1, #0];
    pop   {r0, r1, pc};        ;//    bx     lr ;
;   end of way_out

    nop             ;Add 2 bytes of padding
    END
```

汇编语言文件 （适用于：CooCox+GCC）
CR_func_C.s

```
#/*--------------------------------------------------------------------------*/
#/* CR_func_C.s                                                              */
#/* for CooCox-GCC                                                           */
#/* by fy_zhu , 2013-01                                                      */
#/*--------------------------------------------------------------------------*/

    .code 16

    .section CR_FUNC_C
    .text

@@ <function resume_lc>
    .align 2
    .global   resume_lc
    .code 16
```

```
        .thumb_func
@@ Declaration : void resume_lc(lc_t lc);
@@      r0=[cr->lc]
resume_lc:
        bx    r0;
@@ <end of function resume_lc>

@@ <function yield>
        .align 2
        .global      yield
        .extern    quit_addr
        .code 16
        .thumb_func
@@ Declaration : void yield(lc_t *lc);
@@      r0=&(cr->lc)
yield:
        mov    r1, lr ;
        str    r1, [r0];
        ldr    r0, =quit_addr;
        ldr      r0, [r0];         @;!!!
        bx      r0;
@@ <end of function yield>

@@ <function way_out>
        .align 2
        .global    way_out
        .extern    quit_addr
        .code 16
        .thumb_func
@@ Declaration : void way_out(void);
way_out:
        push {r0, r1, lr};
        mov  r0, lr;
        ldr    r1, =quit_addr;
        str    r0, [r1] ;
        pop    {r0, r1, pc};  @;// bx      lr ;
@@ <end of function way_out>

        nop                          @;Add 2 bytes of padding
        .end
```

本程序清单（ARM Cortex-M）是
《实现协程多任务的无标号单步跳转方法》
[-- 微控制器中基于协程的实时协作多任务方法 (3)]
的附件。见 http://blog.chinaaet.com/detail/31858.html