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 8051

C-example （适用于：KEIL / IAR / Tasking / Raisonance）

CR_ASM_Test.c

```
/*---------------------------------------------------------------------------*/
/* SiLabs C8051F                                                             */
/* by fy_zhu , 2013-01                                                       */
/*---------------------------------------------------------------------------*/
#include <c8051f300.h>                  // SFR declarations
#include "CR_asm.h"

//---------------------------------------------------------------------------
// Function PROTOTYPES
//---------------------------------------------------------------------------
unsigned char TASK_1 (struct cr *cr);
unsigned char 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);
    }
}

unsigned char TASK_1 (struct cr *cr)
{
    CR_START(cr);
```

```
while (1) {
      CR_WAIT_UNTIL(cond1);
      dosomething11();
      CR_YIELD();
      dosomething12();
      }
      CR_END(cr);
}

unsigned char TASK_2 (struct cr *cr)
{
      CR_START(cr);
while (1) {
      CR_WAIT_UNTIL(cond2);
      Dosomething21();
      CR_YIELD();
      Dosomething22();
      }
      CR_END(cr);
}
```

H 文件 （适用于：KEIL / IAR / Tasking / Raisonance）

CR_asm.h

```
/*---------------------------------------------------------------------------*/
/* CR_asm.h    for ALL 8051 C-Compiler                                       */
/* by fy_zhu , 2013-01                                                       */
/*---------------------------------------------------------------------------*/
typedef unsigned short 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_GETWAYOUT     127

extern lc_t lc_addr;
extern unsigned char retcode_temp;

extern void yield(void);
extern void way_out(void);
extern void resume_lc(void);
```

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

#define CR_START(cr)        \
    do { \
         {retcode_temp=CR_GETWAYOUT; way_out();} \
         if (retcode_temp != CR_GETWAYOUT) { (cr)->lc = lc_addr; return retcode_temp; } \
         if ((cr)->lc != 0) { lc_addr = (cr)->lc; resume_lc(); } \
        }while(0)

#define CR_WAIT_UNTIL(condition)    while(!(condition)) { retcode_temp=CR_WAITING; yield(); }

#define CR_YIELD()   { retcode_temp=CR_YIELDED; yield(); }

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

汇编语言文件 （适用于：KEIL）

CR_func_K.asm

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

            NAME    CR_func_K

?DT?CR_func_K     SEGMENT DATA
   RSEG    ?DT?CR_func_K
   PUBLIC lc_addr
   PUBLIC retcode_temp

lc_addr:              DS     2
cr_quit_addr:       DS     2
retcode_temp:     DS     1

;for KEIL, Tasking-EDE, RK51
;(unsigned) short lc_addr  高位在 lc_addr，低位在 lc_addr+1
;for IAR, SDCC
;(unsigned) short lc_addr  高位在 lc_addr+1，低位在 lc_addr

?PR?CR_func_K       SEGMENT CODE
   RSEG    ?PR?CR_func_K
   PUBLIC     resume_lc
```

```
    PUBLIC    yield
    PUBLIC    way_out

;void resume_lc(void);
;lcall      resume_lc(), Now SP:
;     (next-inst_LOW)
;     (next-inst_HIGH)  <-SP
resume_lc:
    DEC     SP                  ;SP←(SP)-1, 不影响 ACC
    DEC     SP                  ;2 字节,1 周期
;for KEIL
    PUSH lc_addr+1              ;lc_addr+1 lc_LOW
    PUSH lc_addr                ;lc_addr lc_HIGH
    RET         ;=LJMP lc_addr=(cr->lc)

;void yield(void);
;lcall      yield(), Now SP:
;     (lc_LOW)
;     (lc_HIGH)     <-SP
yield:
;for KEIL
    POP   lc_addr             ;(lc_HIGH)
    POP   lc_addr+1           ;(lc_LOW)
    PUSH cr_quit_addr+1    ;(quit_addr_LOW)
    PUSH cr_quit_addr       ;(quit_addr_HIGH)
    RET         ;=LJMP quit_addr

;void way_out(void);
;lcall      way_out(), Now SP:
;     (quit_addr_LOW)
;     (quit_addr_HIGH)<-SP
way_out:
;for KEIL
    POP     cr_quit_addr       ;(quit_addr_HIGH)
    POP     cr_quit_addr+1   ;(quit_addr_LOW)
    PUSH cr_quit_addr+1    ;(quit_addr_LOW)
    PUSH cr_quit_addr       ;(quit_addr_HIGH)
    RET         ;=LJMP quit_addr

      END
```

汇编语言文件 （适用于：IAR）

CR_func_I.asm

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

        NAME    CR_func_I

  RSEG   DATA_I:DATA

  PUBLIC lc_addr
  PUBLIC retcode_temp

lc_addr:            DS    2
cr_quit_addr:       DS    2
retcode_temp:       DS    1

;for KEIL, Tasking-EDE, RK51
;(unsigned) short lc_addr  高位在 lc_addr，低位在 lc_addr+1
;for IAR, SDCC
;(unsigned) short lc_addr  高位在 lc_addr+1，低位在 lc_addr

  RSEG   NEAR_CODE:CODE
;     RSEG   ?PR?CR_func_I:CODE

  PUBLIC    resume_lc
  PUBLIC    yield
  PUBLIC    way_out

;void resume_lc(void);
;lcall      resume_lc(), Now SP:
;     (next-inst_LOW)
;     (next-inst_HIGH)  <-SP
resume_lc:
  DEC    SP                   ;SP←(SP)-1, 不影响 ACC
  DEC    SP                   ;2 字节,1 周期
;for IAR
  PUSH   lc_addr             ;lc_addr lc_LOW
  PUSH   lc_addr+1           ;lc_addr+1 lc_HIGH
  RET          ;=LJMP lc_addr=(cr->lc)

;void yield(void);
;lcall      yield(), Now SP:
;     (lc_LOW)
```

```
;     (lc_HIGH)     <-SP
yield:
;for IAR
   POP     lc_addr+1          ;(lc_HIGH)
   POP     lc_addr            ;(lc_LOW)
   PUSH    cr_quit_addr       ;(quit_addr_LOW)
   PUSH    cr_quit_addr+1 ;(quit_addr_HIGH)
   RET           ;=LJMP quit_addr

;void way_out(void);
;lcall       way_out(), Now SP:
;     (quit_addr_LOW)
;     (quit_addr_HIGH)<-SP
way_out:
;for IAR
   POP     cr_quit_addr+1 ;(quit_addr_HIGH)
   POP     cr_quit_addr       ;(quit_addr_LOW)
   PUSH    cr_quit_addr       ;(quit_addr_LOW)
   PUSH    cr_quit_addr+1 ;(quit_addr_HIGH)
   RET     ;=LJMP quit_addr

   END
```

汇编语言文件 （适用于：Tasking）

CR_func_T.asm

```
;/*---------------------------------------------------------------------------*/
;/* CR_func_T.asm                                                             */
;/* for Tasking-EDE                                                           */
;/* by fy_zhu , 2013-01                                                       */
;/*---------------------------------------------------------------------------*/

$CASE

            name    CR_func_T

CR_func_T_DT     segment data
   rseg    CR_func_T_DT
   public _lc_addr
   public _retcode_temp

_lc_addr:              ds     2
cr_quit_addr:        ds     2
_retcode_temp:     ds     1
```

```
;for KEIL, Tasking-EDE, RK51
;(unsigned) short lc_addr 高位在 lc_addr，低位在 lc_addr+1
;for IAR, SDCC
;(unsigned) short lc_addr 高位在 lc_addr+1，低位在 lc_addr

CR_func_T_PR      segment code
   rseg   CR_func_T_PR
   public _?resume_lc
   public _?yield
   public _?way_out

;void resume_lc(void);
;lcall      resume_lc(), Now SP:
;     (next-inst_LOW)
;     (next-inst_HIGH)  <-SP
_?resume_lc:
   DEC     SP                    ;SP←(SP)-1, 不影响 ACC
   DEC     SP                    ;2 字节,1 周期
;for Tasking-EDE
   PUSH    _lc_addr+1        ;lc_addr+1 lc_LOW
   PUSH    _lc_addr          ;lc_addr lc_HIGH
      RET            ;=LJMP lc_addr=(cr->lc)

;void yield(void);
;lcall      yield(), Now SP:
;     (lc_LOW)
;     (lc_HIGH)     <-SP
_?yield:
;for Tasking-EDE
   POP     _lc_addr          ;(lc_HIGH)
   POP     _lc_addr+1        ;(lc_LOW)
   PUSH    cr_quit_addr+1 ;(quit_addr_LOW)
   PUSH    cr_quit_addr     ;(quit_addr_HIGH)
   RET            ;=LJMP quit_addr

;void way_out(void);
;lcall      way_out(), Now SP:
;     (quit_addr_LOW)
;     (quit_addr_HIGH)<-SP
_?way_out:
;for Tasking-EDE
   POP     cr_quit_addr     ;(quit_addr_HIGH)
   POP     cr_quit_addr+1 ;(quit_addr_LOW)
```

```
   PUSH   cr_quit_addr+1 ;(quit_addr_LOW)
   PUSH   cr_quit_addr    ;(quit_addr_HIGH)
   RET          ;=LJMP quit_addr

   end
```

汇编语言文件 （适用于：Raisonance）

CR_func_R.asm

```
;/*---------------------------------------------------------------------------*/
;/* CR_func_R.asm                                                             */
;/* for Raisonance RKit-51                                                    */
;/* by fy_zhu , 2013-01                                                       */
;/*---------------------------------------------------------------------------*/

$include (silabs/c8051f300.inc)                    ; Include register definition file.

          NAME    CR_func_R

?DT?CR_func_R     SEGMENT DATA
   RSEG    ?DT?CR_func_R
   PUBLIC lc_addr
   PUBLIC retcode_temp

lc_addr:              DS     2
cr_quit_addr:       DS     2
retcode_temp:     DS     1

;for KEIL, Tasking-EDE, RK51
;(unsigned) short lc_addr 高位在 lc_addr，低位在 lc_addr+1
;for IAR, SDCC
;(unsigned) short lc_addr 高位在 lc_addr+1，低位在 lc_addr

?PR?CR_func_R       SEGMENT CODE
   RSEG    ?PR?CR_func_R
   PUBLIC    resume_lc
   PUBLIC    yield
   PUBLIC    way_out

;void resume_lc(void);
;lcall       resume_lc( ), Now SP:
;      (next-inst_LOW)
;      (next-inst_HIGH)  <-SP
resume_lc:
```

```
   DEC     SP                  ;SP←(SP)-1, 不影响 ACC
   DEC     SP                  ;2 字节,1 周期
;for RK51
   PUSH    lc_addr+1           ;lc_addr+1 lc_LOW
   PUSH    lc_addr             ;lc_addr lc_HIGH
   RET             ;=LJMP lc_addr=(cr->lc)

;void yield(void);
;lcall      yield(), Now SP:
;      (lc_LOW)
;      (lc_HIGH)     <-SP
yield:
;for RK51
   POP     lc_addr             ;(lc_HIGH)
   POP     lc_addr+1           ;(lc_LOW)
   PUSH    cr_quit_addr+1 ;(quit_addr_LOW)
   PUSH    cr_quit_addr     ;(quit_addr_HIGH)
   RET             ;=LJMP quit_addr

;void way_out(void);
;lcall      way_out(), Now SP:
;      (quit_addr_LOW)
;      (quit_addr_HIGH)<-SP
way_out:
;for RK51
   POP     cr_quit_addr     ;(quit_addr_HIGH)
   POP     cr_quit_addr+1 ;(quit_addr_LOW)
   PUSH    cr_quit_addr+1 ;(quit_addr_LOW)
   PUSH    cr_quit_addr     ;(quit_addr_HIGH)
   RET             ;=LJMP quit_addr

      END
```

本程序清单（8051）是

《实现协程多任务的无标号单步跳转方法(8051)》

[-- 微控制器中基于协程的实时协作多任务方法 (4)]

的附件。见 http://blog.chinaaet.com/detail/31908.html

有关 ARM Cortex-M 程序清单，见

《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。 http://blog.chinaaet.com/detail/31858.html