T-Engine フォーラム主催

【実習】μT-Kernel(初級編)

プログラミング演習テキスト

T-Engine フォーラム 学術・教育 WG 委員

スパンション・イノベイツ株式会社

長濱　みほ

目次

演習 1 タスク起動と優先度

■ プロジェクトフォルダ

C:¥ut_realos¥utkernel¥7-M¥uT-Kernel_Samples¥sta_pri

■ プロゴラム概要

- タスクは，led1(優先度 1)，led2(優先度 2)，led3(優先度 3) から構成される
- 初期化タスク uinit_task はタスクを動的に生成し，led3 を開始する
- 各タスクがより優先順位の高いタスクの起動と字タスクの終了を繰り返す

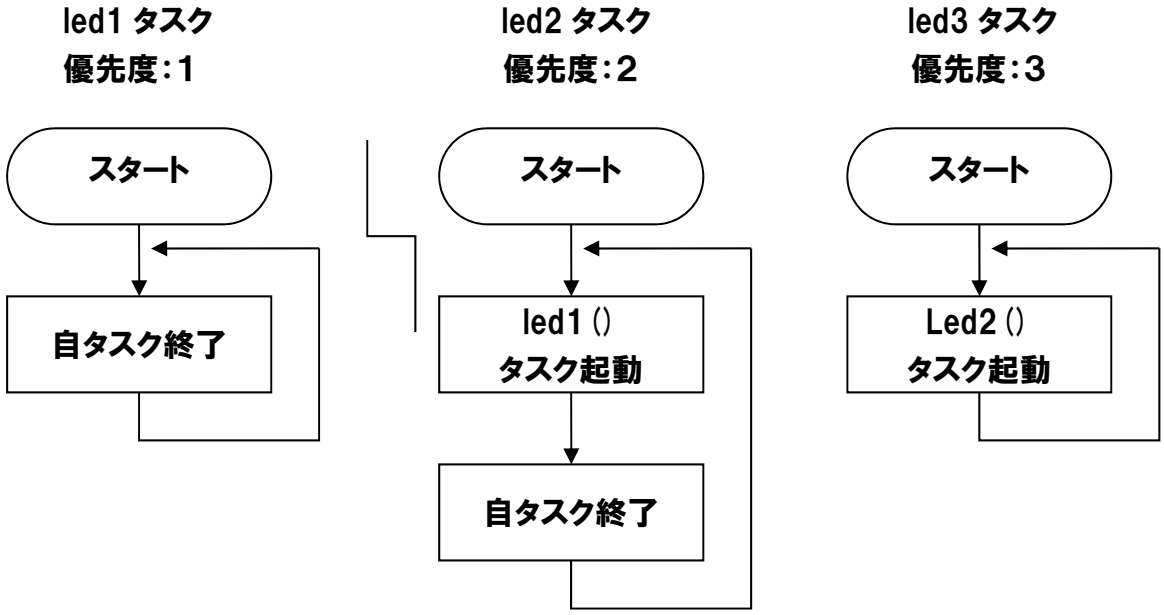


図 1.1 sta_pri の各タスクのフローチャート

```
/*
 *----------------------------------------------------------------------
 *      micro T-Kernel
 *
 *      Copyright (C) 2006-2008 by Ken Sakamura. All rights reserved.
 *      micro T-Kernel is distributed under the micro T-License.
 *----------------------------------------------------------------------
 *      micro T-REALOS/M3
 *
 *      Copyright (C) FUJITSU SEMICONDUCTOR LIMITED 2010 All rights reserved.
 *      This product uses the Source Code of micro T-Kernel under the micro
 *      T-License granted by the T-Engine forum (http://www.t-engine.org).
 *                                                  Last update : 2010/05/11
 *----------------------------------------------------------------------
 *
 *      Version:    1.01.00
 *      Released by T-Engine Forum(http://www.t-engine.org) at 2008/02/25.
 *
 *----------------------------------------------------------------------
*/

#include <tk/tkernel.h>
#include "init_task.h"
#include "mcu.h"

#define USE_WDG 0

#if USE_WDG == 0
#define WDG_CTL   (*(volatile UW *)0x40011008)
#define WDG_LCK   (*(volatile UW *)0x40011C00)
#endif

ID LED1_ID;
ID LED2_ID;
ID LED3_ID;

extern void _reset( void );

EXPORT int main( void )
```

```
{
#if USE_WDG == 0
        WDG_LCK = 0x1acce551;
        WDG_LCK = 0xe5331aae;
        WDG_CTL = 0x00000000;
#endif

        _reset();
        while(1);
}

void delay (void)
{
    Int i;
    For(i=0;i<200000;i++){}
}

/*****************************************************************************/
/* LED1 task                                                                 */
/*****************************************************************************/

Static void led1( INT stacd, VP exinf )
{
    printf("===Start led1()¥n");
    FM3_GPIO->PDOR3 ^= 0x0004;

    printf("led1: goes into DORMANT state¥n");

    printf("===End led1()¥n");
    FM3_GPIO->PDOR3 ^= 0x0004;

    ①                          /* Exit Task */
}

/*****************************************************************************/
/* LED2 task                                                                 */
/*****************************************************************************/

Static void led2( INT stacd, VP exinf )
```

```
79  {
80      ER ercd;
81
82      printf("===Start led2()¥n");
83
84      FM3_GPIO->PDOR3 ^= 0x0008;
85
86      printf("led2: starts led1¥n");
87
88      Ercd =      ②          /* Start Task LED1 */
89      if(ercd != E_OK){while(1){}}
90
91      printf("led2: goes into DORMANT state¥n");
92
93      printf("===End led2()¥n");
94
95      FM3_GPIO->PDOR3 ^= 0x0008;
96
97          ①              /* Exit Task */
98      }
99
100 /*********************************************************************/
101 /* LED3 task                                                         */
102 /*********************************************************************/
103
104 Static void led3( INT stacd, VP exinf )
105 {
106     ER ercd;
107
108     printf("===Start led3()¥n");
109
110     FM3_GPIO->PDOR3 ^= 0x0010;
111
112     While(1)
113     {
114         printf("led3: starts led2¥n");
115
116         ercd =      ③          /* Start Task LED2 */
117         if(ercd != E_OK){while(1){}}
```

```
118     }
119 }
120
121 /*************************************************************************/
122 /* Initialize task                                                       */
123 /*************************************************************************/
124
125 void uinit_task(void)
126 {
127     ER ercd;
128
129     T_CTSK ctsk;
130
131     static INT led1_stack[0x400/4];
132     static INT led2_stack[0x400/4];
133     static INT led3_stack[0x400/4];
134
135     SysTick_init();
136     sys_init();
137
138     LCD_sys_init();
139     LCD_init();
140     LCD_reset();
141
142     LCD_gotoxy(0,0);
143     LCD_sendstring("uT-Kernel");
144     LCD_gotoxy(0,1);
145     LCD_sendstring("start&priority");
146
147     ctsk.exinf     = (VP)1;
148     ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
149     ctsk.task      = led1;
150     ctsk.itskpri = 1;
151     ctsk.stksz     = (W)sizeof(led1_stack);
152     ctsk.bufptr    = led1_stack;
153     LED1_ID = tk_cre_tsk(&ctsk);
154
155     ctsk.exinf     = (VP)2;
156     ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
```

```
157     ctsk.task       = led2;
158     ctsk.itskpri = 2;
159     ctsk.stksz      = (W)sizeof(led2_stack);
160     ctsk.bufptr     = led2_stack;
161     LED2_ID = tk_cre_tsk(&ctsk);
162
163     ctsk.exinf      = (VP)3;
164     ctsk.tskatr     = TA_HLNG | TA_RNG0 | TA_USERBUF;
165     ctsk.task       = led3;
166     ctsk.itskpri = 3;
167     ctsk.stksz      = (W)sizeof(led3_stack);
168     ctsk.bufptr     = led3_stack;
169     LED3_ID = tk_cre_tsk(&ctsk);
170
171     ercd =  [ ④ ]              /* Start Task LED3 */
172     if(ercd != E_OK){while(1){}}
173 }
```

演習2 起床待ちと起床

■ プロジェクトフォルダ

C:¥ut_realos¥utkernel¥7-M¥uT-Kernel_Samples¥slp_wu

■ プロゴラム概要

- タスクは，led1(優先度 6)，led2(優先度 5)，led3(優先度 4) から構成される
- 初期化タスク uinit_task はタスクを動的に生成し，
  他のタスクに起床されるまで待つ.
  他タスクを起床したら，志度タスク起床待ち状態に遷移する.

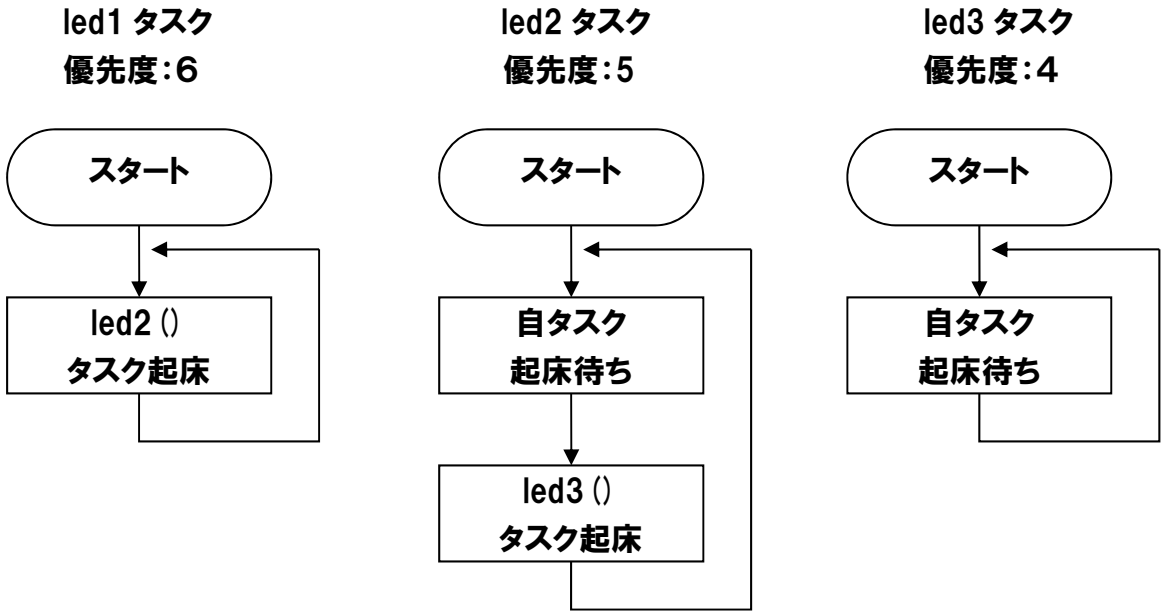


図 2 slp_wup の各タスクのフローチャート

```
/*
 *----------------------------------------------------------------------
 *      micro T-Kernel
 *
 *      Copyright (C) 2006-2008 by Ken Sakamura. All rights reserved.
 *      micro T-Kernel is distributed under the micro T-License.
 *----------------------------------------------------------------------
 *      micro T-REALOS/M3
 *
 *      Copyright (C) FUJITSU SEMICONDUCTOR LIMITED 2010 All rights reserved.
 *      This product uses the Source Code of micro T-Kernel under the micro
 *      T-License granted by the T-Engine forum (http://www.t-engine.org).
 *                                                  Last update : 2010/05/11
 *----------------------------------------------------------------------
 *
 *      Version:    1.01.00
 *      Released by T-Engine Forum(http://www.t-engine.org) at 2008/02/25.
 *
 *----------------------------------------------------------------------
*/

#include <tk/tkernel.h>
#include "init_task.h"
#include "mcu.h"

#define USE_WDG 0

#if USE_WDG == 0
#define WDG_CTL   (*(volatile UW *)0x40011008)
#define WDG_LCK   (*(volatile UW *)0x40011C00)
#endif

ID LED1_ID;
ID LED2_ID;
ID LED3_ID;

extern void _reset( void );

EXPORT int main( void )
```

```
40  {
41  #if USE_WDG == 0
42          WDG_LCK = 0x1acce551;
43          WDG_LCK = 0xe5331aae;
44          WDG_CTL = 0x00000000;
45  #endif
46
47          _reset();
48          while(1);
49  }
50
51  /*********************************************************************/
52  /* LED1 task                                                         */
53  /*********************************************************************/
54
55  static void led1( INT stacd, VP exinf )
56  {
57      ER ercd;
58
59      while(1)
60      {
61          printf("led1: RUNNING¥n");
62          FM3_GPIO->PDOR3 ^= 0x0004;
63
64          printf("led1: WAKEUP led2¥n");
65          FM3_GPIO->PDOR3 ^= 0x0004;
66
67          ercd =      ①                   /* Wake Up Task LED2 */
68          if(ercd != E_OK){while(1){}}
69      }
70  }
71
72  /*********************************************************************/
73  /* LED2 task                                                         */
74  /*********************************************************************/
75
76  static void led2( INT stacd, VP exinf )
77  {
78      ER ercd;
```

```
79
80      while(1)
81      {
82          printf("led2: RUNNING¥n");
83          FM3_GPIO->PDOR3 ^= 0x0008;
84
85          printf("led2: SLEEP¥n");
86          FM3_GPIO->PDOR3 ^= 0x0008;
87
88          ercd =  ②                       /* Sleep Task (No Timeout) */
89          if(ercd != E_OK){while(1){}}
90
91          printf("led2: RUNNING¥n");
92          FM3_GPIO->PDOR3 ^= 0x0008;
93
94          printf("led2: WAKEUP led3¥n");
95          FM3_GPIO->PDOR3 ^= 0x0008;
96
97          ercd =  ③                       /* Wake Up LED3 */
98          if(ercd != E_OK){while(1){}}
99      }
100 }
101 /*********************************************************************/
102 /* LED3 task                                                         */
103 /*********************************************************************/
104
105 static void led3( INT stacd, VP exinf )
106 {
107     ER ercd;
108
109     while(1)
110     {
111         printf("led3: RUNNING¥n");
112         FM3_GPIO->PDOR3 ^= 0x0010;
113
114         printf("led3: SLEEP¥n");
115         FM3_GPIO->PDOR3 ^= 0x0010;
116
117         ercd =  ②                       /* Sleep Task (No Timeout) */
```

```
118         if(ercd != E_OK){while(1){}}
119     }
120 }
121
122 /*************************************************************************/
123 /* Initialize task                                                       */
124 /*************************************************************************/
125
126 void uinit_task(void)
127 {
128             ER ercd;
129
130             T_CTSK ctsk;
131
132             static INT led1_stack[0x400/4];
133             static INT led2_stack[0x400/4];
134             static INT led3_stack[0x400/4];
135
136             SysTick_init();
137             sys_init();
138
139             LCD_sys_init();
140             LCD_init();
141             LCD_reset();
142
143             LCD_gotoxy(0,0);
144             LCD_sendstring("uT-Kernel");
145             LCD_gotoxy(0,1);
146             LCD_sendstring("sleep&wake-up");
147
148           ctsk.exinf     = (VP)1;
149           ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
150           ctsk.task       = led1;
151           ctsk.itskpri = 6;
152           ctsk.stksz     = (W)sizeof(led1_stack);
153           ctsk.bufptr    = led1_stack;
154           LED1_ID = tk_cre_tsk(&ctsk);
155
156           ctsk.exinf     = (VP)2;
```

```
157             ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
158             ctsk.task      = led2;
159             ctsk.itskpri = 5;
160             ctsk.stksz     = (W)sizeof(led2_stack);
161             ctsk.bufptr    = led2_stack;
162             LED2_ID = tk_cre_tsk(&ctsk);
163
164             ctsk.exinf     = (VP)3;
165             ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
166             ctsk.task      = led3;
167             ctsk.itskpri   = 4;
168             ctsk.stksz     = (W)sizeof(led3_stack);
169             ctsk.bufptr    = led3_stack;
170             LED3_ID = tk_cre_tsk(&ctsk);
171
172             ercd = tk_sta_tsk(LED3_ID, NULL);
173             if(ercd != E_OK){while(1){}}
174
175             ercd = tk_sta_tsk(LED2_ID, NULL);
176             if(ercd != E_OK){while(1){}}
177
178             ercd = tk_sta_tsk(LED1_ID, NULL);
179             if(ercd != E_OK){while(1){}}
180     }
```

演習3 セマフォ

■ プロジェクトフォルダ

C:¥ut_realos¥utkernel¥7-M¥uT-Kernel_Samples¥sem

■ プロゴラム概要

- タスクは，led1(優先度 1)，led2(優先度 2)，led3(優先度 3)，psw1 から構成される
- 初期化タスク uinit_task はタスクを動的に生成し，各タスクを順番に起動する
- led1，led2，led3 がそれぞれ 1 つ，2 つ，3 つのセマフォを要求し，待ち状態に遷移する．その後，psw1 が押されるたびにセマフォが返却され，必要なセマフォ数が確保できるようになったら，led1，led2，led3 の待ち状態が解除される．

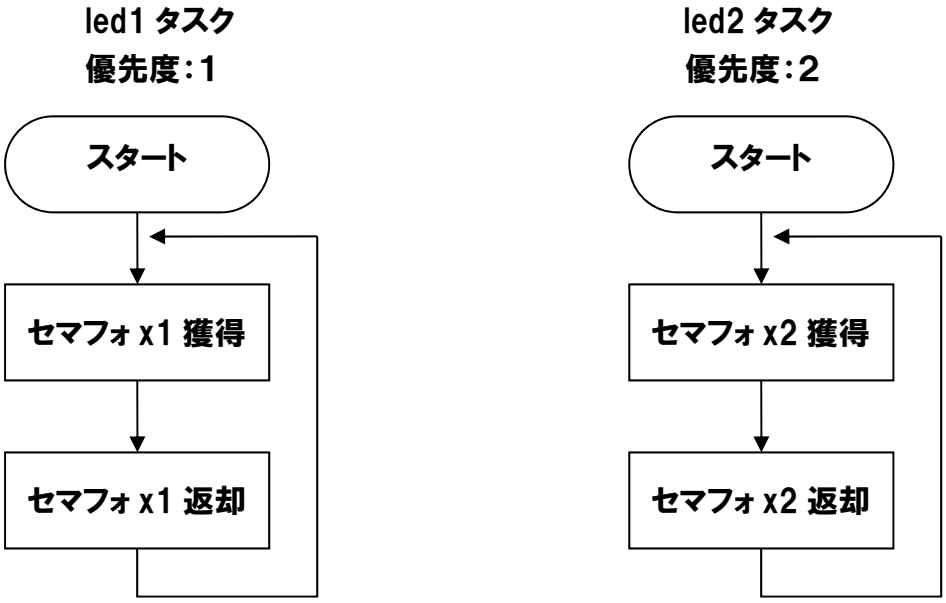


図 3 演習 3 の led1，led2 タスクのフローチャート

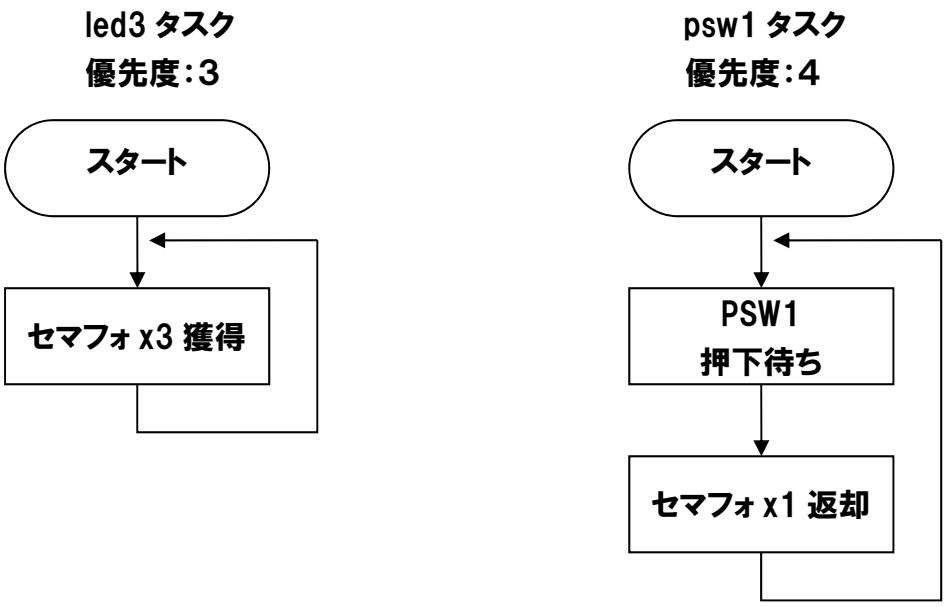


図 4 演習 3 の led3, psw1 タスクのフローチャート

```
/*
 *----------------------------------------------------------------------
 *      micro T-Kernel
 *
 *      Copyright (C) 2006-2008 by Ken Sakamura. All rights reserved.
 *      micro T-Kernel is distributed under the micro T-License.
 *----------------------------------------------------------------------
 *      micro T-REALOS/M3
 *
 *      Copyright (C) FUJITSU SEMICONDUCTOR LIMITED 2010 All rights reserved.
 *      This product uses the Source Code of micro T-Kernel under the micro
 *      T-License granted by the T-Engine forum (http://www.t-engine.org).
 *                                                      Last update : 2010/05/11
 *----------------------------------------------------------------------
 *
 *      Version:    1.01.00
 *      Released by T-Engine Forum(http://www.t-engine.org) at 2008/02/25.
 *
 *----------------------------------------------------------------------
*/

/*
 *                                              Init_task.c (usermain)
 *                                              User Main
*/

#include <tk/tkernel.h>

#include "init_task.h"

#include "mcu.h"

#define USE_WDG 0

#if USE_WDG == 0
#define WDG_CTL   (*(volatile UW *)0x40011008)
#define WDG_LCK   (*(volatile UW *)0x40011C00)
#endif

```

```
40  ID SEM1_ID;
41  ID LED1_ID;
42  ID LED2_ID;
43  ID LED3_ID;
44  ID PSW1_ID;
45
46  extern void _reset( void );
47
48  EXPORT int main( void )
49  {
50  #if USE_WDG == 0
51      WDG_LCK = 0x1acce551;
52      WDG_LCK = 0xe5331aae;
53      WDG_CTL = 0x00000000;
54  #endif
55
56      _reset();
57      while(1);
58  }
59
60  /*****************************************************************************/
61  /* LED1 task                                                                 */
62  /*****************************************************************************/
63
64  static void led1( INT stacd, VP exinf )
65  {
66     ER ercd;
67
68     while(1){
69        ercd =      ①              /* Wait Semaphore x 1 */
70        if(ercd != E_OK){while(1){}}
71
72        FM3_GPIO->PDOR3   = 0x001c;
73        FM3_GPIO->PDOR3 ^= 0x0004;
74
75        ercd =      ②              /* Signal Semaphore x 1 */
76        if(ercd != E_OK){while(1){}}
77     }
78  }
```

```
79
80  /*************************************************************************/
81  /* LED2 task                                                             */
82  /*************************************************************************/
83
84  static void led2( INT stacd, VP exinf )
85  {
86      ER ercd;
87
88      while(1){
89
90          ercd =      ③              /* Wait Semaphore x 2 */
91          if(ercd != E_OK){while(1){}}
92
93          FM3_GPIO->PDOR3 ^= 0x000c;
94
95          ercd =      ④              /* Signal Semaphore x 2 */
96          if(ercd != E_OK){while(1){}}
97      }
98  }
99
100 /*************************************************************************/
101 /* LED3 task                                                             */
102 /*************************************************************************/
103
104 static void led3( INT stacd, VP exinf )
105 {
106     ER ercd;
107
108     while(1){
109
110         ercd =      ⑤              /* Wait Semaphore x 3 */
111         if(ercd != E_OK){while(1){}}
112
113         FM3_GPIO->PDOR3 ^= 0x0018;
114     }
115 }
116
117 /*************************************************************************/
```

```
118  /* Switch1 task                                                          */
119  /************************************************************************/
120
121  static void psw1( INT stacd, VP exinf )
122  {
123     ER ercd;
124
125     int sw1=0;
126     int count;
127     while(1){
128        if(sw1)
129        {
130           if(FM3_GPIO->PDIR5_f.P0)
131           {
132              count++;
133              if(count>3)
134              {
135                 Count=0;
136                 sw1=0;
137              }
138           }
139           Else
140           {
141              count=0;
142           }
143        }
144        Else
145        {
146           if(!FM3_GPIO->PDIR5_f.P0)
147           {
148              count++;
149              if(count>3)
150              {
151                 ercd =      ②          /* Signal Semaphore x 1 */
152                 if(ercd != E_OK){while(1){}}
153                 count=0;
154                 sw1=1;
155              }
156           }
```

```
157             Else
158             {
159                count=0;
160             }
161
162          }
163       }
164    }
165
166    /*************************************************************************/
167    /* Initialize task                                                       */
168    /*************************************************************************/
169
170    void uinit_task(void)
171    {
172                T_CTSK ctsk;
173                T_CSEM csem;
174                static INT led1_stack[0x400/4];
175                static INT led2_stack[0x400/4];
176                static INT led3_stack[0x400/4];
177                static INT psw1_stack[0x400/4];
178
179                /*system clock set*/
180                SysTick_init();
181                sys_init();
182
183                LCD_sys_init();
184                LCD_init();
185                LCD_reset();
186
187                LCD_gotoxy(0,0);
188                LCD_sendstring("ut-Kernel");
189                LCD_gotoxy(0,1);
190                LCD_sendstring("semaphore");
191
192                csem.sematr    = TA_TFIFO | TA_FIRST;
193                csem.isemcnt = 0;
194                csem.maxsem    = 3;
195                SEM1_ID =      ⑥          /* Create Semaphore */
```

```
196
197             ctsk.exinf     = (VP)1;
198             ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
199             ctsk.task      = led1;
200             ctsk.itskpri   = 1;
201             ctsk.stksz     = (W)sizeof(led1_stack);
202             ctsk.bufptr    = led1_stack;
203             LED1_ID = tk_cre_tsk(&ctsk);
204
205             ctsk.exinf     = (VP)2;
206             ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
207             ctsk.task      = led2;
208             ctsk.itskpri = 2;
209             ctsk.stksz     = (W)sizeof(led2_stack);
210             ctsk.bufptr    = led2_stack;
211             LED2_ID = tk_cre_tsk(&ctsk);
212
213             ctsk.exinf     = (VP)3;
214             ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
215             ctsk.task      = led3;
216             ctsk.itskpri = 3;
217             ctsk.stksz     = (W)sizeof(led3_stack);
218             ctsk.bufptr    = led3_stack;
219             LED3_ID = tk_cre_tsk(&ctsk);
220
221             ctsk.exinf     = (VP)4;
222             ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
223             ctsk.task      = psw1;
224             ctsk.itskpri = 4;
225             ctsk.stksz     = (W)sizeof(psw1_stack);
226             ctsk.bufptr    = psw1_stack;
227             PSW1_ID = tk_cre_tsk(&ctsk);
228
229             Tk_sta_tsk(LED1_ID, NULL);
230             Tk_sta_tsk(LED2_ID, NULL);
231             tk_sta_tsk(LED3_ID, NULL);
232             tk_sta_tsk(PSW1_ID, NULL);
233 }
```

演習４ イベントフラグ

■ プロジェクトフォルダ

C:¥ut_realos¥utkernel¥7-M¥uT-Kernel_Samples¥ev_flag

■ プロゴラム概要

- タスクは，led1(優先度 1)，psw1(優先度 2) から構成される
- 初期化タスク uinit_task はタスクを動的に生成し，各タスクを順番に起動する
- psw1 タスクは psw1 スイッチの状態をサンプリングし，psw1 スイッチが押されたら
  フラグパターンを 0x00 から増加させる（0x00，0x01，0x02，0x03，・・・）
- led1 は要求するフラグパターンを設定し，パターンが実行されるまで待状態に遷移する．
  psw1 によって発行されるパターンが要求パターンと一致すれば，待ち状態が解除される．

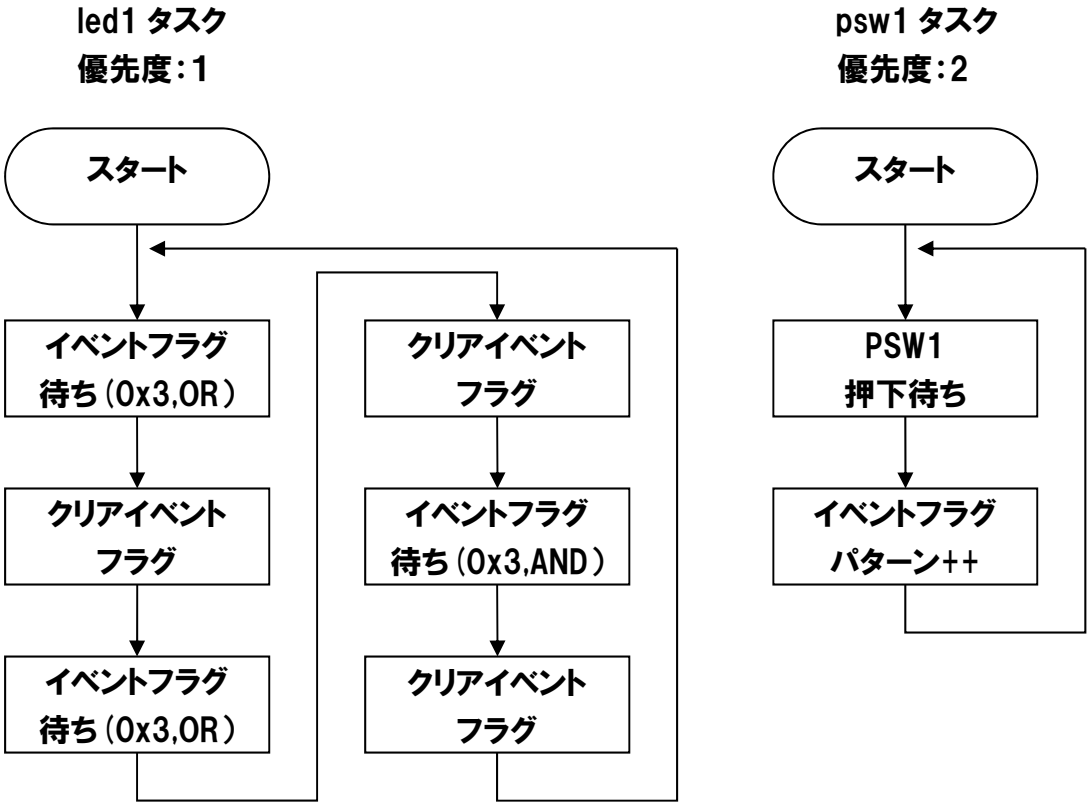


図 5 演習４の各タスクのフローチャート

```
/*
 *----------------------------------------------------------------------
 *      micro T-Kernel
 *
 *      Copyright (C) 2006-2008 by Ken Sakamura. All rights reserved.
 *      micro T-Kernel is distributed under the micro T-License.
 *----------------------------------------------------------------------
 *      micro T-REALOS/M3
 *
 *      Copyright (C) FUJITSU SEMICONDUCTOR LIMITED 2010 All rights reserved.
 *      This product uses the Source Code of micro T-Kernel under the micro
 *      T-License granted by the T-Engine forum (http://www.t-engine.org).
 *                                                      Last update : 2010/05/11
 *----------------------------------------------------------------------
 *
 *      Version:    1.01.00
 *      Released by T-Engine Forum(http://www.t-engine.org) at 2008/02/25.
 *
 *----------------------------------------------------------------------
*/

/*
 *                                      init_task.c (usermain)
 *                                      User Main
*/

#include <tk/tkernel.h>

#include "init_task.h"

#include "mcu.h"

#define USE_WDG 0

#if USE_WDG == 0
#define WDG_CTL   (*(volatile UW *)0x40011008)
#define WDG_LCK   (*(volatile UW *)0x40011C00)
#endif

```

```
40  ID FLG1_ID;
41  ID LED1_ID;
42  ID PSW1_ID;
43
44  UINT flg=0;
45
46  Extern void _reset( void );
47
48  EXPORT int main( void )
49  {
50  #if USE_WDG == 0
51          WDG_LCK = 0x1acce551;
52          WDG_LCK = 0xe5331aae;
53          WDG_CTL = 0x00000000;
54  #endif
55
56          _reset();
57          While(1);
58  }
59
60  /*************************************************************************/
61  /* LED1 task                                                             */
62  /*************************************************************************/
63
64  Static void led1( INT stacd, VP exinf )
65  {
66     UINT ptn;
67     ER ercd;
68
69     while(1){
70
71        /* wait flag 0x3 OR*/
72        printf("WAIT FLAG PTN 0x3 OR¥n");
73
74        Ercd =  [ ① ]                    /* Wait Flag 3 OR */
75        If(ercd != E_OK){while(1){}}
76
77        printf("RECEIVE FLAG 0x%0x¥n",flg);
78
```

```
79        FM3_GPIO->PDOR3    = 0x001c;
80        FM3_GPIO->PDOR3 ^= 0x0004;
81
82        Ercd =  [ ② ]                    /* Clear Flag */
83        If(ercd != E_OK){while(1){}}
84
85        /* wait flag 0x3 OR*/
86        printf("WAIT FLAG PTN 0x3 OR¥n");
87
88        Ercd =  [ ① ]                    /* Wait Flag 3 OR */
89        If(ercd != E_OK){while(1){}}
90
91        Printf("RECEIVE FLAG 0x%0x¥n",flg);
92
93        FM3_GPIO->PDOR3 ^= 0x0008;
94
95        Ercd =  [ ② ]                    /* Clear Flag */
96        If(ercd != E_OK){while(1){}}
97
98        Printf("<RESET FLAG PTN>¥n");
99        flg=0;
100
101       /* wait flag 0x3 AND*/
102       Printf("WAIT FLAG PTN 0x3 AND¥n");
103
104       Ercd =  [ ③ ]                    /* Wait Flag 3 AND */
105       if(ercd != E_OK){while(1){}}
106
107       Printf("RECEIVE FLAG 0x%0x¥n",flg);
108
109       FM3_GPIO->PDOR3 ^= 0x0010;
110
111       Ercd =  [ ② ]                    /* Clear Flag */
112       if(ercd != E_OK){while(1){}}
113
114       Printf("<RESET FLAG PTN>¥n");
115       flg=0;
116    }
117 }
```

```
118
119 /*************************************************************************/
120 /* Switch1 task                                                          */
121 /*************************************************************************/
122
123 static void psw1( INT stacd, VP exinf )
124 {
125     ER ercd;
126
127     Int sw1=0;
128     Int count;
129
130     while(1){
131         if(sw1)
132         {
133             if(FM3_GPIO->PDIR5_f.P0)
134             {
135                 count++;
136                 if(count>3)
137                 {
138                     count=0;
139                     sw1=0;
140                 }
141             }
142             Else
143             {
144                 count=0;
145             }
146         }
147         Else
148         {
149             If(!FM3_GPIO->PDIR5_f.P0)
150             {
151                 count++;
152                 if(count>3)
153                 {
154                     flg++;
155                     ercd =  ②                    /* Clear Flag */
156                     if(ercd != E_OK){while(1){}}
```

```
157
158                 Printf("<FLAG PTN 0x%0x>¥n",flg);
159
160                 ercd =  ④                   /* Set Flag to flg */
161                 if(ercd != E_OK){while(1){}}
162
163                 count=0;
164                 sw1=1;
165               }
166           }
167           Else
168           {
169              count=0;
170           }
171
172         }
173     }
174 }
175
176 /*************************************************************************/
177 /* Initialize task                                                       */
178 /*************************************************************************/
179
180 void uinit_task(void)
181 {
182             T_CTSK ctsk;
183             T_CFLG cflg;
184             static INT led1_stack[0x400/4];
185             static INT psw1_stack[0x400/4];
186
187 /*symtem clock set*/
188             SysTick_init();
189             sys_init();
190
191             LCD_sys_init();
192             LCD_init();
193             LCD_reset();
194
195             LCD_gotoxy(0,0);
```

```
196             LCD_sendstring("ut-Kernel");
197             LCD_gotoxy(0,1);
198             LCD_sendstring("event flag");
199
200             cflg.exinf     = (VP)1;
201             cflg.flgatr    = TA_TFIFO;
202             cflg.iflgptn = 0;
203             FLG1_ID =      ⑤          /* Create Event Flag */
204
205             ctsk.exinf     = (VP)1;
206             ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
207             ctsk.task      = led1;
208             ctsk.itskpri   = 1;
209             ctsk.stksz     = (W)sizeof(led1_stack);
210             ctsk.bufptr    = led1_stack;
211             LED1_ID = tk_cre_tsk(&ctsk);
212
213             ctsk.exinf     = (VP)2;
214             Ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
215             ctsk.task       = psw1;
216             ctsk.itskpri = 2;
217             ctsk.stksz     = (W)sizeof(psw1_stack);
218             ctsk.bufptr    = psw1_stack;
219             PSW1_ID = tk_cre_tsk(&ctsk);
220
221             tk_sta_tsk(LED1_ID, NULL);
222             tk_sta_tsk(PSW1_ID, NULL);
223  }
```

演習５ メールボックス

■ プロジェクトフォルダ

C:¥ut_realos¥utkernel¥7-M¥uT-Kernel_Samples¥mailbox

■ プロゴラム概要

- タスクは，rcv1(優先度 1)，snd1(優先度 2) から構成される
- 初期化タスク uinit_task はタスクを動的に生成し，各タスクを順番に起動する
- rcv1 が実行状態になったら受信できるメッセージが発生するまで待ちます.
  そこで，snd1 からメッセージが送信されたら，rcv1 の待ち状態が解除され，
  rcv1 が受信したメッセージが LCD とターミナル I/O に表示される.

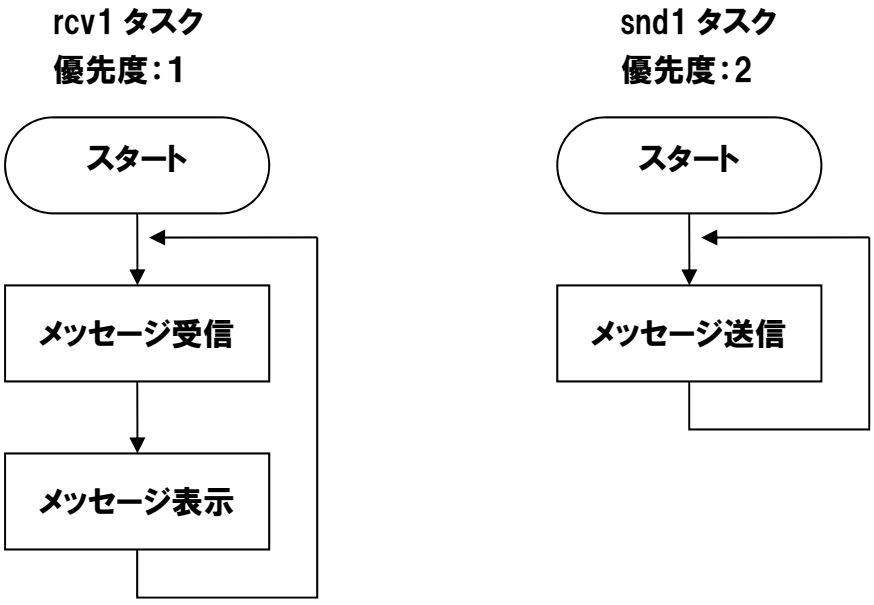


図 6 演習５の各タスクのフローチャート

```
/*
 *----------------------------------------------------------------------
 *      micro T-Kernel
 *
 *      Copyright (C) 2006-2008 by Ken Sakamura. All rights reserved.
 *      micro T-Kernel is distributed under the micro T-License.
 *----------------------------------------------------------------------
 *      micro T-REALOS/M3
 *
 *      Copyright (C) FUJITSU SEMICONDUCTOR LIMITED 2010 All rights reserved.
 *      This product uses the Source Code of micro T-Kernel under the micro
 *      T-License granted by the T-Engine forum (http://www.t-engine.org).
 *                                                      Last update : 2010/05/11
 *----------------------------------------------------------------------
 *
 *      Version:    1.01.00
 *      Released by T-Engine Forum(http://www.t-engine.org) at 2008/02/25.
 *
 *----------------------------------------------------------------------
*/

/*
 *                                              init_task.c (usermain)
 *                                              User Main
*/

#include <tk/tkernel.h>

#include "init_task.h"

#include "mcu.h"

#define USE_WDG 0

#if USE_WDG == 0
#define WDG_CTL   (*(volatile UW *)0x40011008)
#define WDG_LCK   (*(volatile UW *)0x40011C00)
#endif

```

```
40  ID MBX1_ID;
41  ID RCV1_ID;
42  ID SND1_ID;
43
44  UINT flg=0;
45
46  extern void _reset( void );
47
48  EXPORT int main( void )
49  {
50  #if USE_WDG == 0
51          WDG_LCK = 0x1acce551;
52          WDG_LCK = 0xe5331aae;
53          WDG_CTL = 0x00000000;
54  #endif
55
56          _reset();
57          while(1);
58  }
59
60  /*************************************************************************/
61  /* RECEIVE MESSAGE TASK                                                  */
62  /*************************************************************************/
63
64  static void rcv1( INT stacd, VP exinf )
65  {
66      ER ercd;
67
68      typedef struct
69      {
70          T_MSG header;
71          UB msg_cont[64];
72      } mbx_tag;
73      mbx_tag* rx_mail;
74
75
76      while(1){
77          ercd =          ①          /* Recieve Mailbox */
78          if(ercd != E_OK){while(1){}}
```

```
79
80          printf("#Receive Message = %s¥n", rx_mail->msg_cont);
81
82          LCD_gotoxy(0,0);
83          LCD_sendstring("Message: ");
84          LCD_gotoxy(0,1);
85          LCD_sendstring(rx_mail->msg_cont);
86      }
87  }
88
89  /*************************************************************************/
90  /* SEND MESSAGE TASK                                                     */
91  /*************************************************************************/
92
93  static void snd1( INT stacd, VP exinf )
94  {
95      ER ercd;
96      UB str[] = "T-Engine Forum";
97
98      Typedef struct   {
99          T_MSG header;
100         UB msg_cont[64];
101     } mbx_tag;
102     mbx_tag tx_mail;
103
104     strcpy(tx_mail.msg_cont, str);
105
106     while(1){
107         printf("#Send Message¥n");
108         Ercd =          ②          /* Send Mailbox */
109         if(ercd != E_OK){while(1){}}
110     }
111 }
112
113 /*************************************************************************/
114 /* Initialize task                                                       */
115 /*************************************************************************/
116
117 void uinit_task(void)
```

```
118  {
119          ER ercd;
120
121          T_CTSK ctsk;
122          T_CMBX cmbx;
123          static INT rcv1_stack[0x400/4];
124          static INT snd1_stack[0x400/4];
125
126          /*system clock set*/
127          SysTick_init();
128          sys_init();
129
130          LCD_sys_init();
131          LCD_init();
132          LCD_reset();
133
134          LCD_gotoxy(0,0);
135          LCD_sendstring("ut-Kernel");
136          LCD_gotoxy(0,1);
137          LCD_sendstring("mailbox");
138
139          cmbx.exinf     = (VP)1;
140          cmbx.mbxatr    = TA_TFIFO;
141          MBX1_ID =        ③          /* Create Mailbox */
142
143          ctsk.exinf     = (VP)1;
144          ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
145          ctsk.task      = rcv1;
146          ctsk.itskpri = 1;
147          ctsk.stksz     = (W)sizeof(rcv1_stack);
148          ctsk.bufptr    = rcv1_stack;
149          RCV1_ID = tk_cre_tsk(&ctsk);
150
151          ctsk.exinf     = (VP)2;
152          ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
153          ctsk.task      = snd1;
154          ctsk.itskpri = 2;
155          ctsk.stksz     = (W)sizeof(snd1_stack);
156          ctsk.bufptr    = snd1_stack;
```

```
157             SND1_ID = tk_cre_tsk(&ctsk);
158
159             ercd = tk_sta_tsk(RCV1_ID, NULL);
160             if(ercd != E_OK){while(1){}}
161             ercd = tk_sta_tsk(SND1_ID, NULL);
162             if(ercd != E_OK){while(1){}}
163 }
```

演習６ 割込みハンドラ

■ プロジェクトフォルダ

C:¥ut_realos¥utkernel¥7-M¥uT-Kernel_Samples¥int

■ プロゴラム概要

・ タスクは，led1(優先度 1)，led2(優先度 2) と割込みハンドラ int1 から構成される

・ 初期化タスク uinit_task は割込みハンドラ int1 の登録を行った後，
タスクを動的に生成し，各タスクを順番に起動する

・ led1 と led2 が実行状態からすぐに起床待ち状態に遷移する．
psw1 または psw2 が押されたら，割込みハンドラが起動し，led1 または led2 を起床する．
その後，起床されたタスクが該当する LED を点灯/消灯し，再度起床待ち状態に遷移する．

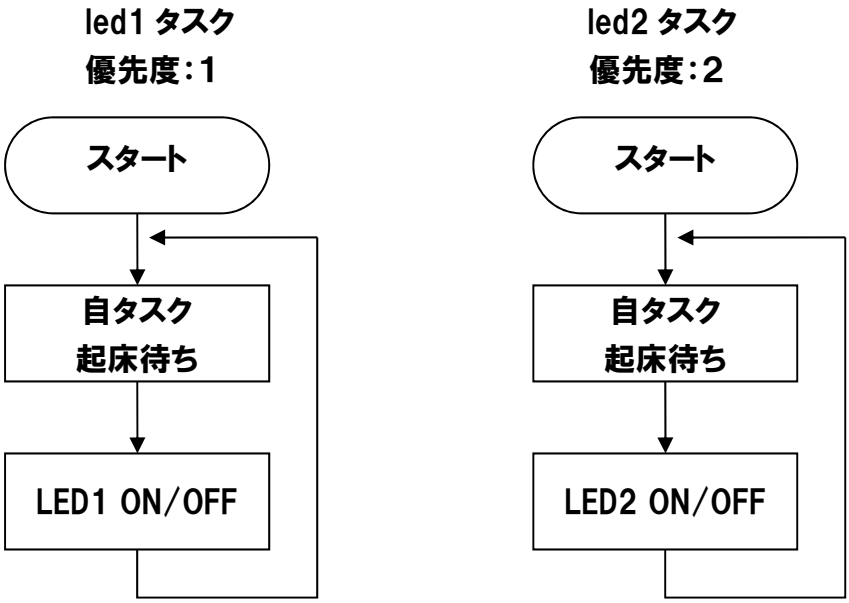


図 7 演習６の led1，led2 タスクのフローチャート

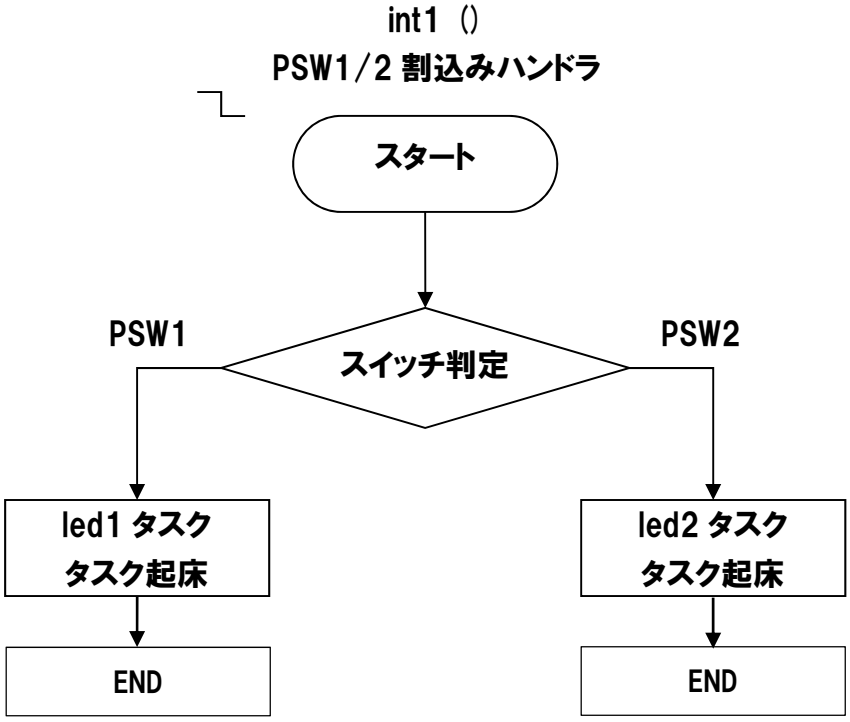


図 8 演習６の割り込みハンドラのフローチャート

```
/*
 *----------------------------------------------------------------------
 *      micro T-Kernel
 *
 *      Copyright (C) 2006-2008 by Ken Sakamura. All rights reserved.
 *      micro T-Kernel is distributed under the micro T-License.
 *----------------------------------------------------------------------
 *      micro T-REALOS/M3
 *
 *      Copyright (C) FUJITSU SEMICONDUCTOR LIMITED 2010 All rights reserved.
 *      This product uses the Source Code of micro T-Kernel under the micro
 *      T-License granted by the T-Engine forum (http://www.t-engine.org).
 *                                                      Last update : 2010/05/11
 *----------------------------------------------------------------------
 *
 *      Version:    1.01.00
 *      Released by T-Engine Forum(http://www.t-engine.org) at 2008/02/25.
 *
 *----------------------------------------------------------------------
*/

/*
 *                                              init_task.c (usermain)
 *                                              User Main
*/

#include <tk/tkernel.h>

#include "init_task.h"

#include "mcu.h"

#define USE_WDG                                                         0

#if USE_WDG == 0
#define WDG_CTL   (*(volatile UW *)0x40011008)
#define WDG_LCK   (*(volatile UW *)0x40011C00)
#endif

```

```
40  ID LED1_ID;
41  ID LED2_ID;
42
43  extern void _reset( void );
44
45  EXPORT int main( void )
46  {
47  #if USE_WDG == 0
48          WDG_LCK = 0x1acce551;
49          WDG_LCK = 0xe5331aae;
50          WDG_CTL = 0x00000000;
51  #endif
52
53          _reset();
54          While(1);
55  }
56
57  /*********************************************************************/
58  /* LED1 task                                                         */
59  /*********************************************************************/
60
61  static void led1( INT stacd, VP exinf )
62  {
63     ER ercd;
64
65     while(1)
66     {
67        Ercd = tk_slp_tsk(TMO_FEVR);
68        if(ercd != E_OK){while(1){}}
69
70        FM3_GPIO->PDOR3 ^= 0x0004;
71     }
72  }
73
74  /*********************************************************************/
75  /* LED2 task                                                         */
76  /*********************************************************************/
77
78  static void led2( INT stacd, VP exinf )
```

```
79  {
80      ER ercd;
81
82      while(1)
83      {
84          Ercd = tk_slp_tsk(TMO_FEVR);
85          if(ercd != E_OK){while(1){}}
86
87          FM3_GPIO->PDOR3 ^= 0x0008;
88      }
89  }
90
91
92  /*********************************************************************/
93  /* SWITCH1 task                                                      */
94  /*********************************************************************/
95
96  static void psw1(void)
97  {
98      ER ercd;
99
100     ercd = tk_wup_tsk(LED1_ID);
101     if(ercd != E_OK){while(1){}}
102 }
103
104 /*********************************************************************/
105 /* SWITCH2 task                                                      */
106 /*********************************************************************/
107
108 static void psw2(void)
109 {
110     ER ercd;
111
112     ercd = tk_wup_tsk(LED2_ID);
113     if(ercd != E_OK){while(1){}}
114 }
115
116
117
```

```
118 /************************************************************************/
119 /* INTERRUPT HANDLER                                                    */
120 /************************************************************************/
121
122 void int1(void)
123 {
124     if(FM3_EXTI->EIRR_f.ER0)
125     {
126         FM3_EXTI->EICL_f.ECL0 = 0;
127         Psw1();
128     }
129     else if(FM3_EXTI->EIRR_f.ER1)
130     {
131         FM3_EXTI->EICL_f.ECL1 = 0;
132         Psw2();
133     }
134     Else
135     {
136        FM3_EXTI->EIRR = 0x0000;
137        FM3_GPIO->PDOR3 ^= 0x000c;
138     }
139 }
140
141 /************************************************************************/
142 /* Initialize task                                                      */
143 /************************************************************************/
144
145 void uinit_task(void)
146 {
147             ER ercd;
148             UINT intst;
149
150             T_CTSK ctsk;
151             T_DINT dint;
152
153              static INT led1_stack[0x400/4];
154              static INT led2_stack[0x400/4];
155
156              /*system clock set*/
```

```
157             DI(intst);
158
159              SysTick_init();
160             sys_init();
161
162             dint.intatr    = TA_HLNG;
163             dint.inthdr    = int1;
164             ercd =          ①           /* Define Interrupt */
165             if(ercd != E_OK){while(1){}}
166
167             EI(intst);
168
169             LCD_sys_init();
170             LCD_init();
171             LCD_reset();
172
173             LCD_gotoxy(0,0);
174             LCD_sendstring("uT-Kernel");
175
176             LCD_gotoxy(0,1);
177             LCD_sendstring("Int Handler");
178
179             ctsk.exinf     = (VP)1;
180             ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
181             ctsk.task       = led1;
182             ctsk.itskpri = 1;
183             ctsk.stksz     = (W)sizeof(led1_stack);
184             ctsk.bufptr    = led1_stack;
185             LED1_ID = tk_cre_tsk(&ctsk);
186
187             ctsk.exinf     = (VP)2;
188             ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
189             ctsk.task       = led2;
190             ctsk.itskpri = 2;
191             ctsk.stksz     = (W)sizeof(led2_stack);
192             ctsk.bufptr    = led2_stack;
193             LED2_ID = tk_cre_tsk(&ctsk);
194
195             tk_sta_tsk(LED1_ID, NULL);
```

```
196             tk_sta_tsk(LED2_ID, NULL);
197 }
```

演習7 周期ハンドラ

■ プロジェクトフォルダ

C:¥ut_realos¥utkernel¥7-M¥uT-Kernel_Samples¥cyclic

■ プロゴラム概要

・タスクは, lcd1(優先度 1), 周期ハンドラ cyc1 から構成される

・初期化タスク uinit_task は周期ハンドラとタスクを動的に生成し, タスクを起動させる.
周期ハンドラは lcd1 タスクにより動作開始される.

・lcd1 タスクは時間を表示したら, 起床待ち状態に遷移する.
また,周期ハンドラ cyc1 は1s毎に起動しており,周期ハンドラの中で時間カウントされている.
時間カウントが終了したら lcd1 タスクを起床させる.
起床した lcd1 タスクは時間を表示する. 以降この処理を繰り返す.

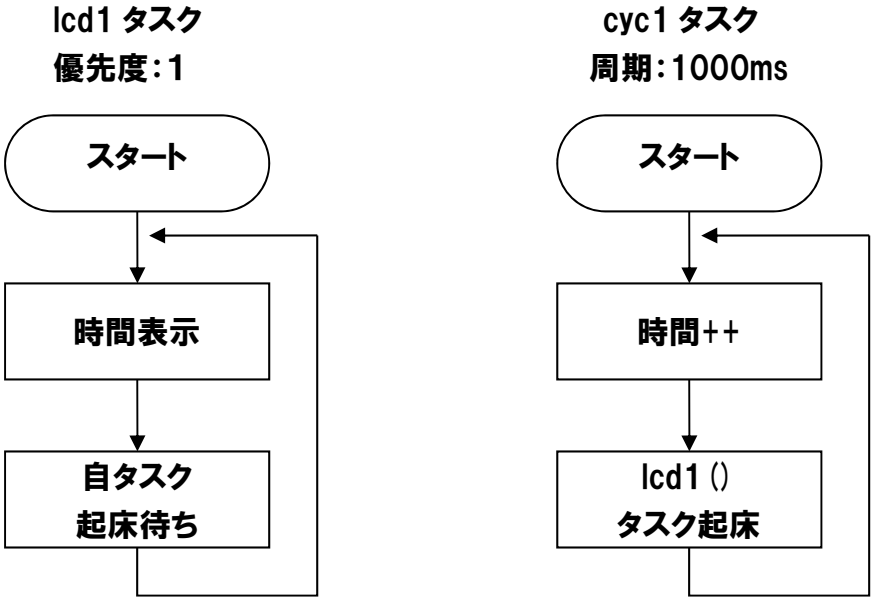


図 9 演習７の各タスクのフローチャート

```
/*
 *----------------------------------------------------------------------
 *      micro T-Kernel
 *
 *      Copyright (C) 2006-2008 by Ken Sakamura. All rights reserved.
 *      micro T-Kernel is distributed under the micro T-License.
 *
 *----------------------------------------------------------------------
 *      micro T-REALOS/M3
 *
 *      Copyright (C) FUJITSU SEMICONDUCTOR LIMITED 2010 All rights reserved.
 *      This product uses the Source Code of micro T-Kernel under the micro
 *      T-License granted by the T-Engine forum (http://www.t-engine.org).
 *                                                      Last update : 2010/05/11
 *
 *----------------------------------------------------------------------
 *
 *      Version:    1.01.00
 *      Released by T-Engine Forum(http://www.t-engine.org) at 2008/02/25.
 *
 *----------------------------------------------------------------------
 */

/*
 *                              init_task.c (usermain)
 *                              User Main
 */

#include <tk/tkernel.h>

#include "init_task.h"

#include "mcu.h"

#define USE_WDG 0

#if USE_WDG == 0
```

```
36  #define WDG_CTL   (*(volatile UW *)0x40011008)
37  #define WDG_LCK   (*(volatile UW *)0x40011C00)
38  #endif
39
40  ID LCD1_ID;
41  ID CYC1_ID;
42
43  unsigned int sec  = 0;
44  unsigned int dsec = 0;
45  unsigned int min  = 0;
46  unsigned int dmin = 0;
47
48  extern void _reset( void );
49
50  EXPORT int main( void )
51  {
52  #if USE_WDG == 0
53        WDG_LCK = 0x1acce551;
54        WDG_LCK = 0xe5331aae;
55        WDG_CTL = 0x00000000;
56  #endif
57
58        _reset();
59        while(1);
60  }
61
62  /*************************************************************************/
63  /* LCD TASK                                                              */
64  /*************************************************************************/
65
66  static void lcd1( INT stacd, VP exinf )
67  {
68        ER ercd;
69
70        ercd =  [ ① ]            /* Start Cyclic Handler */
71        if(ercd != E_OK){while(1){}}
72
73        LCD_gotoxy(2,1);
74        LCD_sendstring(":");
```

```
75
76          while(1)
77          {
78              LCD_gotoxy(0,1);
79              LCD_sendchar(dmin+'0');
80
81              LCD_gotoxy(1,1);
82              LCD_sendchar(min+'0');
83
84              LCD_gotoxy(3,1);
85              LCD_sendchar(dsec+'0');
86
87              LCD_gotoxy(4,1);
88              LCD_sendchar(sec+'0');
89
90              ercd = tk_slp_tsk(TMO_FEVR);
91              if(ercd != E_OK){while(1){}}
92          }
93  }
94
95  /*************************************************************************/
    /* CYCLIC HANDLER
96  */
97  /*************************************************************************/
98
99  static void cyc1( INT stacd, VP exinf )
100 {
101     ER ercd;
102
103     sec++;
104
105     if(sec==10)
106     {
107         sec=0;
108         dsec++;
109
110         if(dsec==6)
111         {
112
```

```
113             dsec=0;
114             min++;
115
116             if(min==10)
117             {
118                 min=0;
119                 dmin++;
120
121                 if(dmin==6)
122
123                     dmin=0;
124             }
125         }
126     }
127
128     ercd = tk_wup_tsk(LCD1_ID);
129     if(ercd != E_OK){while(1){}}
130 }
131
132
133 /*************************************************************************/
134 /* Initialize task                                                       */
135 /*************************************************************************/
136
137 void uinit_task(void)
138 {
139             T_CTSK ctsk;
140             T_CCYC ccyc;
141             static INT lcd1_stack[0x400/4];
142
143             /*system clock set*/
144             SysTick_init();
145             sys_init();
146
147             LCD_sys_init();
148             LCD_init();
149             LCD_reset();
150
151             LCD_gotoxy(0,0);
```

```
152             LCD_sendstring("cyclic handler");
153
154             ccyc.exinf     = (VP)1;
155             ccyc.cycatr    = TA_HLNG;
156             ccyc.cychdr    = cyc1;
157             ccyc.cyctim    =        ②        /* Cycle Time = 1s */
158             ccyc.cycphs    = 0;
159             CYC1_ID =        ③          /* Create Cyclic Handler */
160
161             ctsk.exinf     = (VP)1;
162             ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
163             ctsk.task      = lcd1;
164             ctsk.itskpri = 1;
165             ctsk.stksz     = (W)sizeof(lcd1_stack);
166             ctsk.bufptr    = lcd1_stack;
167             LCD1_ID = tk_cre_tsk(&ctsk);
168
169             tk_sta_tsk(LCD1_ID, NULL);
170     }
```

演習８ タイムシェアリング

■ プロジェクトフォルダ

C:¥ut_realos¥utkernel¥7-M¥uT-Kernel_Samples¥time_share

■ プロゴラム概要

・タスクは，led1(優先度 2)，led2(優先度 2) ，led3(優先度 2) と
周期ハンドラ cyc1 から構成される

・初期化タスク uinit_task は周期ハンドラとタスクを動的に生成し，各タスクを起動させる.

・各タスクは実行状態になったら該当する LED を点灯.
また，周期ハンドラ cyc1 は 1s 毎に起動しており，周期ハンドラの中で各タスクの優先順位を回転させている．これにより全タスクの優先度が同じにもかかわらず，実行中のタスクが順番に切り替わる.

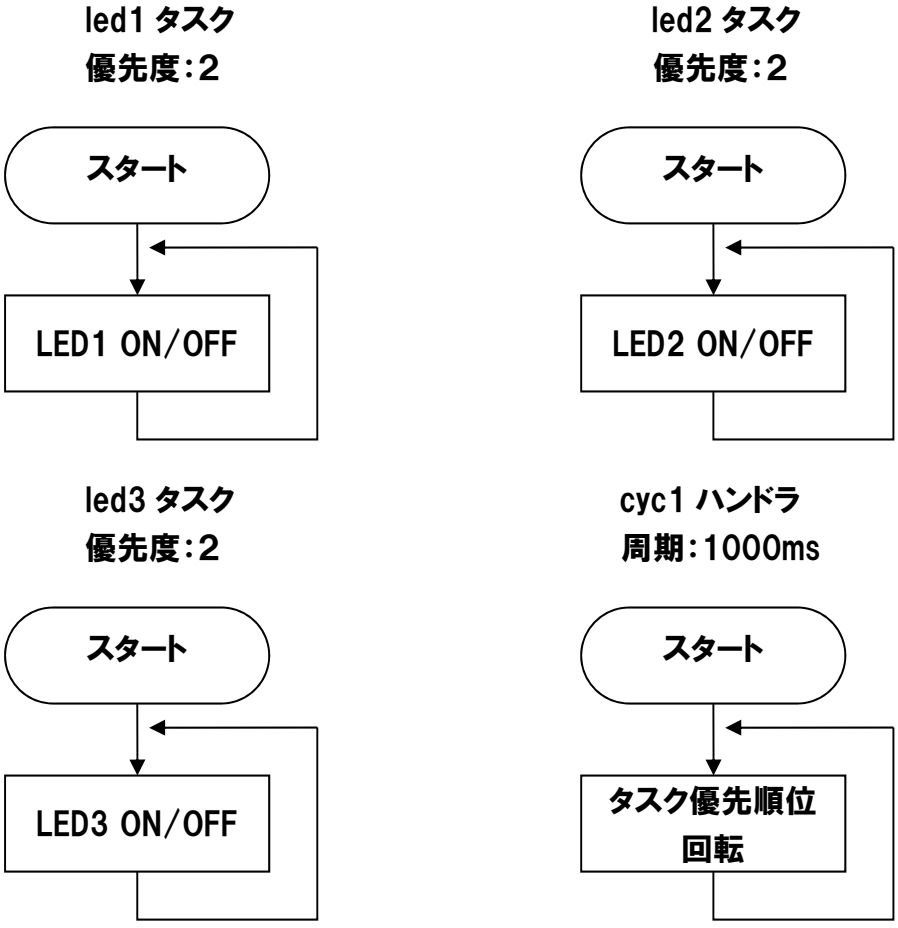


図 10 演習８の各タスクのフローチャート

```
/*
 *----------------------------------------------------------------------
 *      micro T-Kernel
 *
 *      Copyright (C) 2006-2008 by Ken Sakamura. All rights reserved.
 *      micro T-Kernel is distributed under the micro T-License.
 *----------------------------------------------------------------------
 *      micro T-REALOS/M3
 *
 *      Copyright (C) FUJITSU SEMICONDUCTOR LIMITED 2010 All rights reserved.
 *      This product uses the Source Code of micro T-Kernel under the micro
 *      T-License granted by the T-Engine forum (http://www.t-engine.org).
 *                                                  Last update : 2010/05/11
 *----------------------------------------------------------------------
 *
 *      Version:    1.01.00
 *      Released by T-Engine Forum(http://www.t-engine.org) at 2008/02/25.
 *
 *----------------------------------------------------------------------
 */

/*
 *                                      init_task.c (usermain)
 *                                      User Main
 */

#include <tk/tkernel.h>

#include "init_task.h"

#include "mcu.h"

#define USE_WDG                                                 0

#if USE_WDG == 0
```

```
36  #define WDG_CTL   (*(volatile UW *)0x40011008)
37  #define WDG_LCK   (*(volatile UW *)0x40011C00)
38  #endif
39
40  ID LED1_ID;
41  ID LED2_ID;
42  ID LED3_ID;
43  ID CYC1_ID;
44
45  extern void _reset( void );
46
47  EXPORT int main( void )
48  {
49  #if USE_WDG == 0
50      WDG_LCK = 0x1acce551;
51      WDG_LCK = 0xe5331aae;
52      WDG_CTL = 0x00000000;
53  #endif
54
55      _reset();
56      while(1);
57  }
58
59  /*********************************************************************/
    /* LED1 TASK
60  */
61  /*********************************************************************/
62
63  static void led1( INT stacd, VP exinf )
64  {
65      while(1)
66      {
67          FM3_GPIO->PDOR3   = 0x001c;
68          FM3_GPIO->PDOR3 ^= 0x0004;
69      }
70  }
71
72  /*********************************************************************/
73  /* LED2 TASK
```

```
    */
74  /*************************************************************************/
75
76  static void led2( INT stacd, VP exinf )
77  {
78      while(1)
79      {
80          FM3_GPIO->PDOR3   = 0x001c;
81          FM3_GPIO->PDOR3 ^= 0x0008;
82      }
83  }
84
85  /*************************************************************************/
    /* LED3 TASK
86  */
87  /*************************************************************************/
88
89  static void led3( INT stacd, VP exinf )
90  {
91      while(1)
92      {
93          FM3_GPIO->PDOR3   = 0x001c;
94          FM3_GPIO->PDOR3 ^= 0x0010;
95      }
96  }
97
98
99  /*************************************************************************/
    /* CYCLIC HANDLER
100 */
101 /*************************************************************************/
102
103 static void cyc1( INT stacd, VP exinf )
104 {
105   ER ercd;
106
107   ercd =      ①          /* Rotate Ready Queue */
108   if(ercd != E_OK){while(1){}}
109 }
```

```
110
111
112 /*************************************************************************/
113 /* Initialize task                                                       */
114 /*************************************************************************/
115
116 void uinit_task(void)
117 {
118             ER ercd;
119
120             T_CTSK ctsk;
121             T_CCYC ccyc;
122             static INT led1_stack[0x400/4];
123             static INT led2_stack[0x400/4];
124             static INT led3_stack[0x400/4];
125
126             /*system clock set*/
127             SysTick_init();
128             sys_init();
129
130             LCD_sys_init();
131             LCD_init();
132             LCD_reset();
133
134             LCD_gotoxy(0,0);
135             LCD_sendstring("uT-Kernel");
136             LCD_gotoxy(0,1);
137             LCD_sendstring("time sharing");
138
139             ccyc.exinf     = (VP)1;
140             ccyc.cycatr    = TA_HLNG | TA_STA;
141             ccyc.cychdr    = cyc1;
142             ccyc.cyctim    =      ②      /* Cycle Time = 1s */
143             ccyc.cycphs    = 0;
144             CYC1_ID =      ③           /* Create Cyclic Handler */
145
146             ctsk.exinf     = (VP)1;
147             ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
148             ctsk.task      = led1;
```

```
149             ctsk.itskpri = 2;
150             ctsk.stksz     = (W)sizeof(led1_stack);
151             ctsk.bufptr    = led1_stack;
152             LED1_ID = tk_cre_tsk(&ctsk);
153
154             ctsk.exinf     = (VP)1;
155             ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
156             ctsk.task      = led2;
157             ctsk.itskpri = 2;
158             ctsk.stksz     = (W)sizeof(led2_stack);
159             ctsk.bufptr    = led2_stack;
160             LED2_ID = tk_cre_tsk(&ctsk);
161
162             ctsk.exinf     = (VP)1;
163             ctsk.tskatr    = TA_HLNG | TA_RNG0 | TA_USERBUF;
164             ctsk.task      = led3;
165             ctsk.itskpri = 2;
166             ctsk.stksz     = (W)sizeof(led3_stack);
167             ctsk.bufptr    = led3_stack;
168             LED3_ID = tk_cre_tsk(&ctsk);
169
170             ercd = tk_sta_tsk(LED1_ID, NULL);
171             if(ercd != E_OK){while(1){}}
172
173             ercd = tk_sta_tsk(LED2_ID, NULL);
174             if(ercd != E_OK){while(1){}}
175
176             ercd = tk_sta_tsk(LED3_ID, NULL);
177             if(ercd != E_OK){while(1){}}
178 }
```

演習 9 総合演習1 スロットゲーム

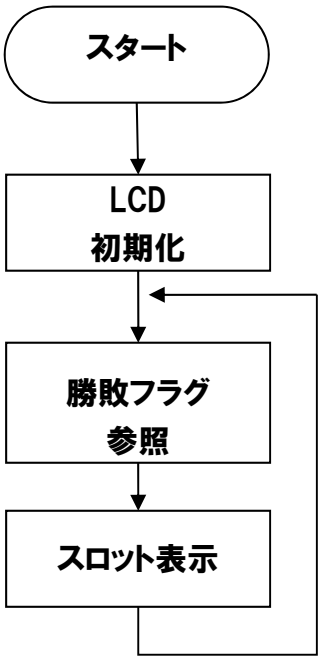
■ プロジェクトフォルダ

C:¥ut_realos¥utkernel¥7-M¥uT-Kernel_Samples¥app_slot

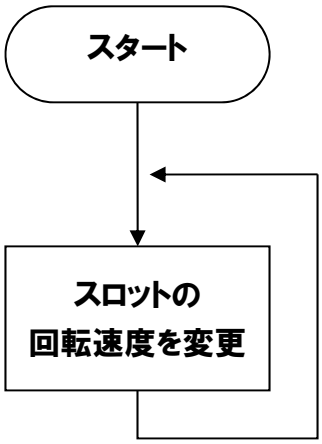
■ プロゴラム概要

- タスクは,「LCD 表示タスク(優先度 2)」,「スロット回転制御用タスク (優先度 1)」 , 周期ハンドラ は「スライダーの ADC 値を利用しスピードをつくるハンドラ」, 「LCD 表示を制御する, スロット回転ハンドラ」,「LCD の wait を制御するハンドラ」 から構成される
- 初期化タスク uinit_task は各タスクを起動させる.
- 「LCD 表示タスク」は「スロット回転ハンドラ」により起こされる. 「スロット回転制御用タスク」は「スピードをつくるハンドラ」により起こされる.
- ADC の値は 10ms 毎に検出する.
  - 回転速度を算出
  - LED の表示を設定(LED8 点灯~LED1-8 点灯)
  - ADC の値に基づいてスロットの回転スピードを設定(100ms~1s)
- PSW1 の状態をウォッチ
  - PSW1 が押されたら, スロットの回転を開始/停止
- スロットが完全に停止したら, 勝敗を判定し, 結果を LCD に表示

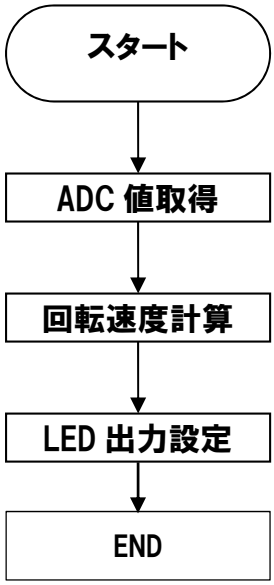
LCD 表示タスク(優:2)
lcd_display()



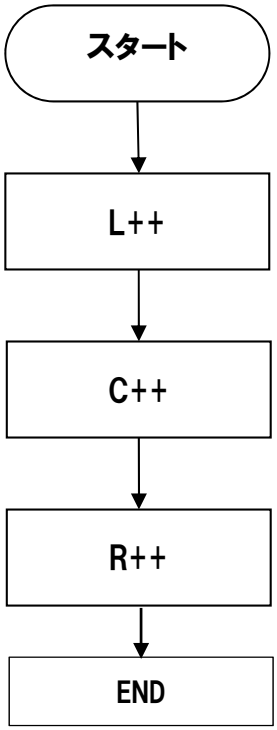
スロット回転制御用タスク(優:1)
slot_speed()



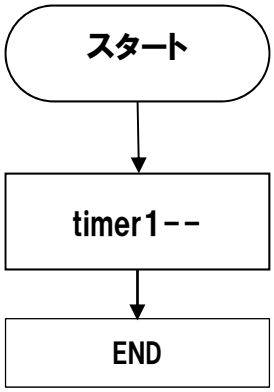
ADC 値取得ハンドラ(10ms)
adc_scan()



スロット回転ハンドラ
turn_slot()



LCD の wait を制御するハンドラ
lcd_wait()



PSW1 ハンドラ関数
psw1()

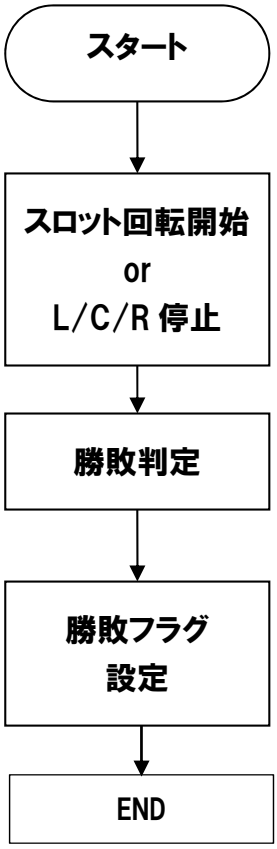


図 11 演習９の各関数フローチャート

演習 10 総合演習2 音楽プレーヤ

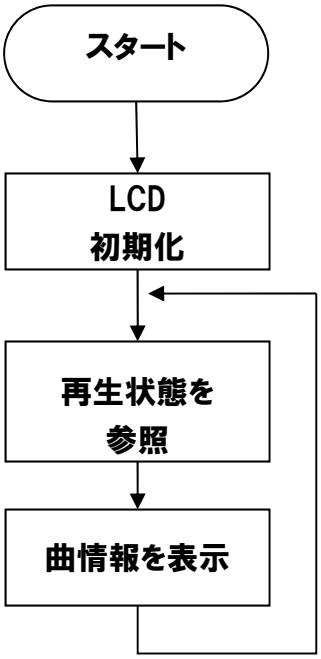
10.1 プロジェクトフォルダ

C:¥ut_realos¥utkernel¥7-M¥uT-Kernel_Samples¥app_music_player

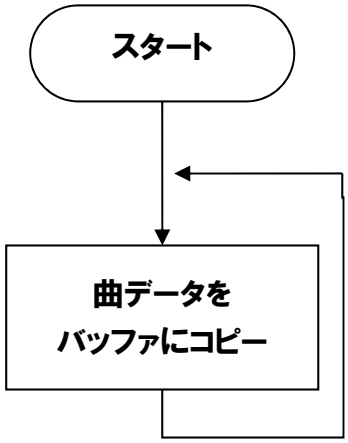
10.2 app_music_player プログラム概要

1. 先頭の曲情報を LCD に表示
2. ジョイスティック(SW6)で曲を選択
3. PSW1 押下で曲を再生
   - 再生時間が 00:00 からカウントアップされる
   - 曲の長さ分再生したら再生停止
   - 再度 PSW を押したら一時停止
   - 曲データは LED1～LED8 に表示される
4. PSW2 押下で再生を停止

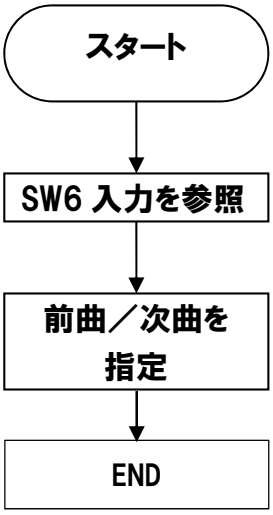
LCD 表示関数
lcd_display()



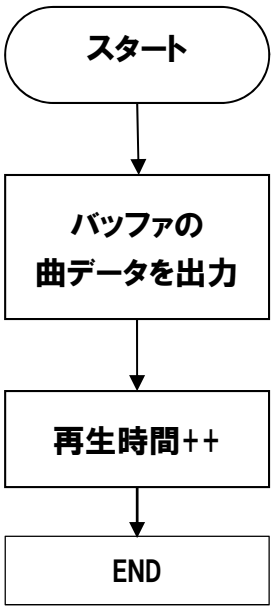
曲データ読込み関数
data_read()



入力処理関数
input_scan()



曲再生関数
play_count()



PSW1/PSW2 ハンドラ関数
psw()

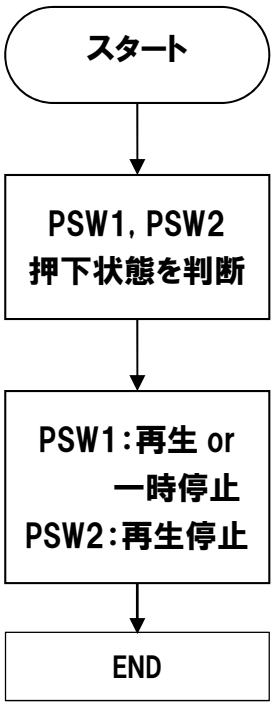


図 12 演習１０の各関数フローチャート

T-Engine フォーラム

T-Engine 講習会

【実習】μT-Kernel 入門(初級編)

プログラミング演習テキスト

2013 年 12 月 18 日発行

発行所

T-Engine フォーラム

(YRP ユビキタス・ネットワーキング研究所内)

〒141-0031 東京都品川区五反田 2-20-1 第 28 興和ビル

URL: http://www.t-engine.org/ja/

TEL:03-5437-0572(代表)　FAX:03-5437-2399(代表)

Printed in Japan

回答例

演習１ タスク起動と優先度

① tk_ext_tsk();　　② tk_sta_tsk(LED1_ID,NULL);
③ tk_sta_tsk(LED2_ID,NULL);　④ tk_sta_tsk(LED3_ID, NULL);

演習２ 起床待ちと起床

① tk_wup_tsk(LED2_ID);　　② tk_slp_tsk(TMO_FEVR);
② tk_wup_tsk(LED3_ID);

演習３ セマフォ

① tk_wai_sem(SEM1_ID,1,TMO_FEVR);　② tk_sig_sem(SEM1_ID,1);
③ tk_wai_sem(SEM1_ID,2,TMO_FEVR);　④ tk_sig_sem(SEM1_ID,2);
⑤ tk_wai_sem(SEM1_ID,3,TMO_FEVR);　⑥ tk_cre_sem(&csem);

演習４ イベントフラグ

① tk_wai_flg(FLG1_ID,3,TWF_ORW,&ptn,TMO_FEVR);
② tk_clr_flg(FLG1_ID,0);
③ tk_wai_flg(FLG1_ID,3,TWF_ANDW,&ptn,TMO_FEVR);
④ tk_set_flg(FLG1_ID,flg);
⑤ tk_cre_flg(&cflg);

演習５ メールボックス

① tk_rcv_mbx(MBX1_ID,(T_MSG**)&rx_mail,TMO_FEVR);
② tk_snd_mbx(MBX1_ID,(T_MSG*)&tx_mail);
③ tk_cre_mbx(&cmbx);

演習６ 割込みハンドラ

① tk_def_int(20,&dint);

演習７ 周期ハンドラ

① tk_sta_cyc(CYC1_ID);　② 1000　③ tk_cre_cyc(&ccyc);

演習８　タイムシェアリング

① tk_rot_rdq(TPRI_RUN);