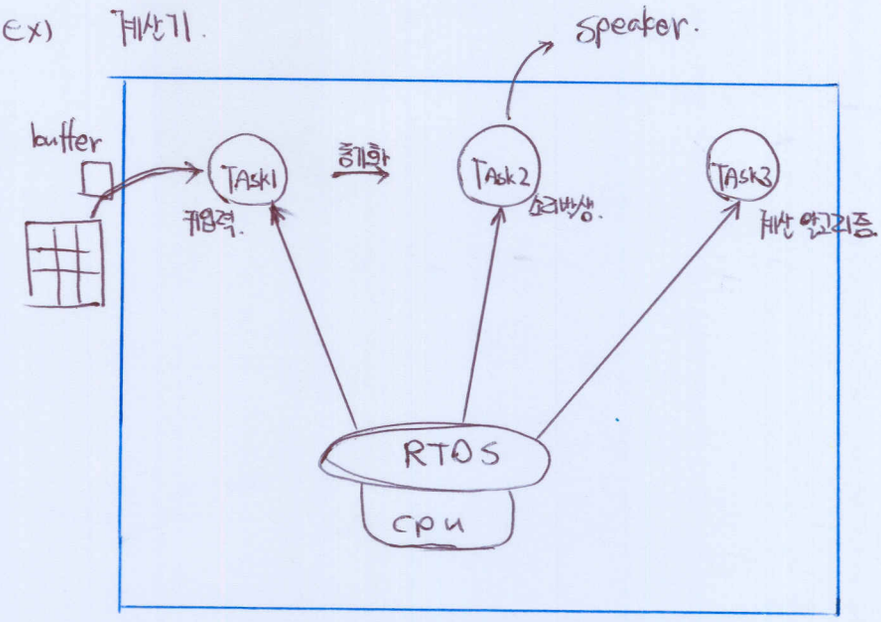
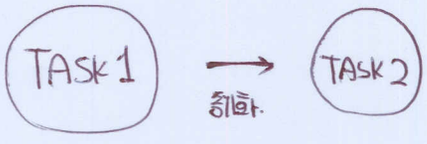
ex) 계산기.



< 계산기 >



동기화구현.

① Semaphore (counting semaphore)

- 간단한 경우에만 , 명령 1개.

② Event Flag

- 복잡한 명령어의 경우에만, 명령어 여러 개.

TASK1

post : 명령을 내린다.

pend : 명령을 기다린다.

TASK2

★ 세마포어를 이용한 동기화.

① 세마포어 생성 OSSemCreate

② 명령을 내린다. OSSemPost → Task 1.

⋮

③ 명령을 기다린다. OSSemPend → Task 2

- 예제코드.

커널적.
Void TASK1()
{ ① sem = OSSemCreate();

key input();

② OSSemPost (sem);

우선순위가 높으면 실행.
Void TASK2()
{
③ OSSemPend (sem, 0, &err);
Sound();

계산문과관련.
Void TASK3()

※ ◦ Mutual Exclusion → Binary Semaphore
◦ 동기화. → Counter Semaphore

※ { Post → +1
Pend → -1

※ 동기화 할 때는. "0"을 사용한다.

OSSemCreate(0);
↓ 이유는?

★ 공유자원을 접근X, 초기화 위해 '0'

세마포어는 ≥'0' 이다.
pend 반복시. '0'상태.
Waiting 상태.

Counter Semaphore → post 함수즉 증가. Pend 함수즉 감소.

```
/*
*********************************************************************************************************
*                                              uC/OS-II
*                                        The Real-Time Kernel
*
*                     (c) Copyright 1992-1998, Jean J. Labrosse, Plantation, FL
*                                          All Rights Reserved
*
*                                              V2.51
*
*                                            EXAMPLE #1
*********************************************************************************************************
*/
#include "stdio.h"
#include "includes.h"
/*
*********************************************************************************************************
*                                              CONSTANTS
*********************************************************************************************************
*/

//#define  TASK_STK_SIZE        512       /* Size of each task's stacks (# of WORDs)   */
#define  TASK_STK_SIZE         8192       /* Size of each task's stacks (# of WORDs)   */
#define  N_TASKS                       5         /* Number of identical tasks                */

/*
*********************************************************************************************************
*                                              VARIABLES
*********************************************************************************************************
*/

OS_STK          TaskStk[N_TASKS][TASK_STK_SIZE];     /* Tasks stacks                                  */
OS_STK          TaskStartStk[TASK_STK_SIZE];

OS_EVENT *sem;

/*
*********************************************************************************************************
*                                              USER VARIABLES
*********************************************************************************************************
*/
#define FND (*((volatile ushort *)(0x0C000002)))

#define DOT1 (*((volatile ushort *) (0x0C000004)))
#define DOT2 (*((volatile ushort *) (0x0C000008)))

#define LED (*((volatile uchar *) (0x0C000016)))

#define TEXT_LCD_INST (*((volatile ushort *) (0x0C000014)))
#define TEXT_LCD_DATA (*((volatile ushort *) (0x0C000012)))

#define FND_RUN                 1
#define LED_RUN                 1
#define TEXT_LCD_RUN    1
#define DOT_RUN                 1


#if FND_RUN
        unsigned char Getsegcode (int x);
#endif

#if TEXT_LCD_RUN
#define LCD_FUNC_SET            0x38
#define LCD_DISP_SET            0x0c
#define LCD_CLEAR_SET           0x01
#define LCD_ENTRY_SET           0x07
#define LCD_RETURN_HOME         0x02
#define LCD_FIRST_LINE_ADDRESS  0x80
#define LCD_SECOND_LINE_ADDRESS 0xc0
#endif


#if TEXT_LCD_RUN
void udelay (unsigned int count)
{
        while ( count -- ) {
                if (count == 0) break;
        }
}
#endif
```

```
#if FND_RUN
unsigned char Getsegcode(int x)
{
        char code;

        switch (x) {
                case 0x0 : code = 0x3f; break;
                case 0x1 : code = 0x06; break;
                case 0x2 : code = 0x5b; break;
                case 0x3 : code = 0x4f; break;
                case 0x4 : code = 0x66; break;
                case 0x5 : code = 0x6d; break;
                case 0x6 : code = 0x7d; break;
                case 0x7 : code = 0x07; break;
                case 0x8 : code = 0x7f; break;
                case 0x9 : code = 0x6f; break;
                case 0xA : code = 0x77; break;
                case 0xB : code = 0x7c; break;
                case 0xC : code = 0x39; break;
                case 0xD : code = 0x5e; break;
                case 0xE : code = 0x79; break;
                case 0xF : code = 0x71; break;
                                //case 0xm : code = 0x40; break;
                                //case 0xq : code = 0x48; break;
                default : code = 0; break;
        }

        return code;
}
#endif

#if DOT_RUN
        void dot1_display (ushort value);
        void dot2_display (ushort value);
#endif

#if DOT_RUN
        void dot1_display(ushort value)
        {
                OS_ENTER_CRITICAL ();
                DOT1 = value ;
                OS_EXIT_CRITICAL ();
                OSTimeDly(1);
        }

        void dot2_display(ushort value)
        {
                OS_ENTER_CRITICAL ();
                DOT2 = value ;
                OS_EXIT_CRITICAL ();
                OSTimeDly(1);
        }
#endif


#define STEP_MOTOR (*((volatile ushort *) (0x0C00000C)))

#define STEP_MOTOR_RUN  1

#if STEP_MOTOR_RUN
#define A       (unsigned short)(0x1)
#define AB      (unsigned short)(0x5)
#define B       (unsigned short)(0x4)
#define _AB     (unsigned short)(0x6)
#define _A      (unsigned short)(0x2)
#define _A_B    (unsigned short)(0xa)
#define _B      (unsigned short)(0x8)
#define A_B     (unsigned short)(0x9)
#endif


/*
*********************************************************************************************************
*                                          FUNCTION PROTOTYPES
*********************************************************************************************************
*/

void   Task1(void *data);                       /* Function prototypes of tasks                  */
void   Task2(void *data);                       /* Function prototypes of tasks                  */
void   Task3(void *data);
void   Task4(void *data);
```

```
void   TaskStart(void *data);                          /* Function prototypes of Startup task          */


/*
*********************************************************************************************************
*                                                MAIN
*********************************************************************************************************
*/

void C_Entry(void)
{
        SerialInit();
        printf("₩nC_Entry...₩n");
        OSInit();                                                  /* Initialize uC/OS-II                  */
        printf("₩nOSInit...₩n");

        OSTaskCreate(TaskStart, (void *)0,&TaskStartStk[TASK_STK_SIZE - 1], 4);
        printf("₩nOSTaskCreate...₩n");

        OSStart();                                                 /* Start multitasking                   */
        printf("₩nOSStart...₩n");

}

/*
*********************************************************************************************************
*                                            STARTUP TASK
*********************************************************************************************************
*/

void TaskStart (void *data)
{
        printf("TaskStart!!₩n");

        OSTimer0_Period_Setting();
        OSTimer0_Interrupt_Setting();                    /* Timer Enable
  */

        OSStatInit();                                    /* Initialize uC/OS-II's statistics        */
        printf("Task...!!₩n");

        OSTaskCreate(Task1,"Task1",&TaskStk[0][TASK_STK_SIZE - 1],6);

        OSTaskCreate(Task2,"Task2",&TaskStk[1][TASK_STK_SIZE - 1],5);

        OSTaskCreate(Task3,"Task3",&TaskStk[2][TASK_STK_SIZE - 1],7);

        OSTaskCreate(Task4,"Task4",&TaskStk[3][TASK_STK_SIZE - 1],8);

        sem = OSSemCreate(0);

        for (;;) {

                /*
                printf("₩n--------------------------------");
                printf("₩nRunning Task : %5d₩n", OSTaskCtr);                                    // Display
#tasks running
                printf("Cpu Usage : %3d₩n", OSCPUUsage);                                        //
Display CPU usage in %
                printf("Context Switches per Sec : %5d₩n",(int)OSCtxSwCtr);             // Display #context
switches per second
                printf("₩n--------------------------------₩n");
                OSCtxSwCtr = 0;
                */

                //OSTimeDlyHMSM(0, 0, 1, 0);
        // Wait one second
                OSTimeDly(50);

                }
}
/*
*********************************************************************************************************
*                                                TASKS
*********************************************************************************************************
*/

void Task1(void *data)
{
        for (;;) {
```

```
                printf((char *)data);
                printf("----------------------₩n");

                #if LED_RUN
                LED = 0x01;
                OSTimeDly(10);
                LED = 0x02;
                OSTimeDly(10);
                LED = 0x04;
                OSTimeDly(10);
                LED = 0x08;
                OSTimeDly(10);
                LED = 0x10;
                OSTimeDly(10);
                LED = 0x20;
                OSTimeDly(10);
                LED = 0x40;
                OSTimeDly(10);
                LED = 0x80;
                OSTimeDly(10);
                LED = 0x00;     // end value
                #endif

                printf("TASK1 : POST OK₩n");

                OSSemPost(sem);

                OSTimeDly(50);

                }
}


void Task2(void *data)
{
        for (;;) {
                printf((char *)data);
                printf("----------------------₩n");

                INT8U   err;

                OSSemPend(sem, 0, &err);

                printf("TASK3 : Pend OK₩n");

                int step;

                for(step = 0 ; step < 8; step++ ){

                #if STEP_MOTOR_RUN
                // A -> B -> _A -> _B
                STEP_MOTOR = A;
                OSTimeDly(1);
                STEP_MOTOR = AB;
                OSTimeDly(1);
                STEP_MOTOR = B;
                OSTimeDly(1);
                STEP_MOTOR = _AB;
                OSTimeDly(1);
                STEP_MOTOR = _A;
                OSTimeDly(1);
                STEP_MOTOR = _A_B;
                OSTimeDly(1);
                STEP_MOTOR = _B;
                OSTimeDly(1);
                STEP_MOTOR = A_B;
                OSTimeDly(1);
                #endif

                }


                OSTimeDly(50);

        }
}


void Task3(void *data)
{
        for (;;) {
```

```
                printf((char *)data);
                printf("----------------------₩n");


                OSTimeDly(50);



        }
}


void Task4(void *data)
{
        for (;;) {
                printf((char *)data);
                printf("----------------------₩n");

                OSTimeDly(50);

        }
}
```

" Event Flag " - 복잡한 경우 " 多 대응어"

< 이벤트 플래그를 이용한 응기화 >

① 이벤트 플래그 생성 OSFlag Create

② 명령을 내린다. OSFlag Post

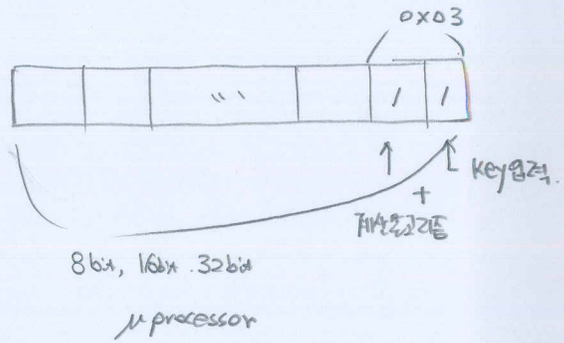
③ 명령을 기다린다. OSFlag Pend

ex) OS_FLAG_GRP *ef;

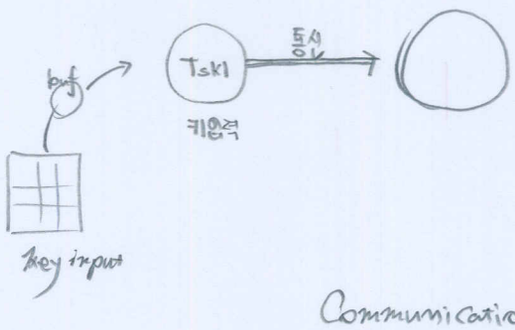
```
Void TASK1() //키입력.
{
  ef = OSFlagCreate(0, &err);

  KeyInput();

  OSFlagPost(ef, 0x01, OS_FLAG_SET, &err);
```

```
Void TASK2() // 출력, 우선순위 높다.
{ ::
  OSFlagPend(ef, 0x03, OS_FLAG_SET_ANY
          + OS_FLAG_CONSUME, &err);

  //ANY : 어느것이든 하나.
  -ALL : 모두다
```

```
Void TASK3() // 계산알고리즘.
{
  CaCulate();

  OSFlagPost(ef, 0x02, OS_FLAG_SET, &err);
```



InterTASk Communication 〈ITC〉



Communication 통신

inter Task Communication

두가지 방법이 있다.

① mail Box. (간단. 정보 1개씩만)

② Queue (복잡. 정보 多 수개)

메일박스를 이용한 통신

① 메일박스 생성 OSMboxCreate

② 정보를 보낸다 OSMboxPost

③ 정보를 받는다 OSMboxPend

Coding EX)

```
Void TASK1()
{   mb = OSMboxCreate(0);
    key = KeyInput();

    OSMboxPost(mb, (void*)key);
}
```

```
VoidTASK2()
{
    in = OSMboxPend(mb, 0, &err);

    case(in)
    1: ~   // 1번소리
    2: ~   // 2번소리
    3: ~   // 3번소리
    ;

}
```

```
/*
*********************************************************************************************************
*                                              uC/OS-II
*                                        The Real-Time Kernel
*
*                       (c) Copyright 1992-1998, Jean J. Labrosse, Plantation, FL
*                                          All Rights Reserved
*
*                                              V2.51
*
*                                            EXAMPLE #1
*********************************************************************************************************
*/
#include "stdio.h"
#include "includes.h"
/*
*********************************************************************************************************
*                                              CONSTANTS
*********************************************************************************************************
*/

//#define  TASK_STK_SIZE        512       /* Size of each task's stacks (# of WORDs)   */
#define  TASK_STK_SIZE          8192       /* Size of each task's stacks (# of WORDs)   */
#define  N_TASKS                        5         /* Number of identical tasks                */

/*
*********************************************************************************************************
*                                              VARIABLES
*********************************************************************************************************
*/

OS_STK          TaskStk[N_TASKS][TASK_STK_SIZE];     /* Tasks stacks                                  */
OS_STK          TaskStartStk[TASK_STK_SIZE];

OS_EVENT *sem;
OS_EVENT *mb;
/*
*********************************************************************************************************
*                                              USER VARIABLES
*********************************************************************************************************
*/
#define FND (*((volatile ushort *)(0x0C000002)))

#define DOT1 (*((volatile ushort *) (0x0C000004)))
#define DOT2 (*((volatile ushort *) (0x0C000008)))

#define LED (*((volatile uchar *) (0x0C000016)))

#define TEXT_LCD_INST (*((volatile ushort *) (0x0C000014)))
#define TEXT_LCD_DATA (*((volatile ushort *) (0x0C000012)))

#define FND_RUN                 1
#define LED_RUN                 1
#define TEXT_LCD_RUN    1
#define DOT_RUN                 1


#if FND_RUN
        unsigned char Getsegcode (int x);
#endif

#if TEXT_LCD_RUN
#define LCD_FUNC_SET            0x38
#define LCD_DISP_SET            0x0c
#define LCD_CLEAR_SET           0x01
#define LCD_ENTRY_SET           0x07
#define LCD_RETURN_HOME         0x02
#define LCD_FIRST_LINE_ADDRESS  0x80
#define LCD_SECOND_LINE_ADDRESS 0xc0
#endif


#if TEXT_LCD_RUN
void udelay (unsigned int count)
{
        while ( count -- ) {
                if (count == 0) break;
        }
}
#endif
```

```
#if FND_RUN
unsigned char Getsegcode(int x)
{
        char code;

        switch (x) {
                case 0x0 : code = 0x3f; break;
                case 0x1 : code = 0x06; break;
                case 0x2 : code = 0x5b; break;
                case 0x3 : code = 0x4f; break;
                case 0x4 : code = 0x66; break;
                case 0x5 : code = 0x6d; break;
                case 0x6 : code = 0x7d; break;
                case 0x7 : code = 0x07; break;
                case 0x8 : code = 0x7f; break;
                case 0x9 : code = 0x6f; break;
                case 0xA : code = 0x77; break;
                case 0xB : code = 0x7c; break;
                case 0xC : code = 0x39; break;
                case 0xD : code = 0x5e; break;
                case 0xE : code = 0x79; break;
                case 0xF : code = 0x71; break;
                                //case 0xm : code = 0x40; break;
                                //case 0xq : code = 0x48; break;
                default : code = 0; break;
        }

        return code;
}
#endif

#if DOT_RUN
        void dot1_display (ushort value);
        void dot2_display (ushort value);
#endif

#if DOT_RUN
        void dot1_display(ushort value)
        {
                OS_ENTER_CRITICAL ();
                DOT1 = value ;
                OS_EXIT_CRITICAL ();
                OSTimeDly(1);
        }

        void dot2_display(ushort value)
        {
                OS_ENTER_CRITICAL ();
                DOT2 = value ;
                OS_EXIT_CRITICAL ();
                OSTimeDly(1);
        }
#endif


#define STEP_MOTOR (*((volatile ushort *) (0x0C00000C)))

#define STEP_MOTOR_RUN  1

#if STEP_MOTOR_RUN
#define A       (unsigned short)(0x1)
#define AB      (unsigned short)(0x5)
#define B       (unsigned short)(0x4)
#define _AB     (unsigned short)(0x6)
#define _A      (unsigned short)(0x2)
#define _A_B    (unsigned short)(0xa)
#define _B      (unsigned short)(0x8)
#define A_B     (unsigned short)(0x9)
#endif


/*
*********************************************************************************************************
*                                          FUNCTION PROTOTYPES
*********************************************************************************************************
*/

void   Task1(void *data);                                /* Function prototypes of tasks                 */
void   Task2(void *data);                                /* Function prototypes of tasks                 */
void   Task3(void *data);
void   Task4(void *data);
```

```
void   TaskStart(void *data);                          /* Function prototypes of Startup task          */


/*
*********************************************************************************************************
*                                                MAIN
*********************************************************************************************************
*/

void C_Entry(void)
{
        SerialInit();
        printf("₩nC_Entry...₩n");
        OSInit();                                                  /* Initialize uC/OS-II                  */
        printf("₩nOSInit...₩n");

        OSTaskCreate(TaskStart, (void *)0,&TaskStartStk[TASK_STK_SIZE - 1], 4);
        printf("₩nOSTaskCreate...₩n");

        OSStart();                                                 /* Start multitasking                   */
        printf("₩nOSStart...₩n");

}

/*
*********************************************************************************************************
*                                            STARTUP TASK
*********************************************************************************************************
*/

void TaskStart (void *data)
{
        printf("TaskStart!!₩n");

        OSTimer0_Period_Setting();
        OSTimer0_Interrupt_Setting();                    /* Timer Enable
  */

        OSStatInit();                                    /* Initialize uC/OS-II's statistics       */
        printf("Task...!!₩n");

        OSTaskCreate(Task1,"Task1",&TaskStk[0][TASK_STK_SIZE - 1],6);

        OSTaskCreate(Task2,"Task2",&TaskStk[1][TASK_STK_SIZE - 1],5);

        OSTaskCreate(Task3,"Task3",&TaskStk[2][TASK_STK_SIZE - 1],7);

        OSTaskCreate(Task4,"Task4",&TaskStk[3][TASK_STK_SIZE - 1],8);

        sem = OSSemCreate(0);
        mb = OSMboxCreate(0);

        for (;;) {

                /*
                printf("₩n--------------------------------");
                printf("₩nRunning Task : %5d₩n", OSTaskCtr);                                    // Display
#tasks running
                printf("Cpu Usage : %3d₩n", OSCPUUsage);                                                //
Display CPU usage in %
                printf("Context Switches per Sec : %5d₩n",(int)OSCtxSwCtr);             // Display #context
switches per second
                printf("₩n--------------------------------₩n");
                OSCtxSwCtr = 0;
                */

                //OSTimeDlyHMSM(0, 0, 1, 0);
        // Wait one second
                OSTimeDly(50);

                }
}
/*
*********************************************************************************************************
*                                                TASKS
*********************************************************************************************************
*/

void Task1(void *data)
{
        for (;;) {
```

```
                printf((char *)data);
                printf("----------------------₩n");

                INT8U key = 3;

                printf(" key = %d ₩n", key);

                OSMboxPost(mb,(void*)key);

                OSTimeDly(50);

                }
}


void Task2(void *data)
{
        for (;;) {
                printf((char *)data);
                printf("----------------------₩n");

                int in = 0;

                INT8U err;

                in = OSMboxPend(mb, 0, &err);

                printf(" sound = %d ₩n", in);

                OSTimeDly(50);

        }
}


void Task3(void *data)
{

        for (;;) {
                printf((char *)data);
                printf("----------------------₩n");

                OSTimeDly(50);

        }
}


void Task4(void *data)
{
        for (;;) {
                printf((char *)data);
                printf("----------------------₩n");

                OSTimeDly(50);

        }
}
```