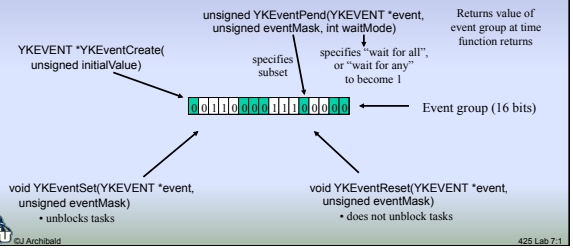
Lab 7: Event flags

- You add functions to create, pend on, set, and reset event flags.



Lab 7 application code

```
/* File: lab7app.c
   Application code for ECEn 425 lab 7 (Event flags) */

#include "clib.h"
#include "yakk.h"              /* kernel definitions */
#include "lab7defs.h"
#define TASK_STACK_SIZE 512

YKEVENT *charEvent;
YKEVENT *numEvent;

int CharTaskStk[TASK_STACK_SIZE];
int AllCharsTaskStk[TASK_STACK_SIZE];
int AllNumsTaskStk[TASK_STACK_SIZE];
int STaskStk[TASK_STACK_SIZE];

void main(void) {
   YKInitialize();
   charEvent = YKEventCreate(0);
   numEvent = YKEventCreate(0);
   YKNewTask(STask, (void *)
         &STaskStk[TASK_STACK_SIZE], 0);
   YKRun();
}
```

```
/* File: lab7defs.h
   Required definitions for ECEn 425 lab 7 (Event flags) */

#define EVENT_A_KEY 0x1
#define EVENT_B_KEY 0x2
#define EVENT_C_KEY 0x4

#define EVENT_1_KEY 0x1
#define EVENT_2_KEY 0x2
#define EVENT_3_KEY 0x4
```

```
/* interrupt handler code from myinth.c */

void mykeybrd(void) {
   char c;
   c = KeyBuffer;
   if      (c == 'a') YKEventSet(charEvent, EVENT_A_KEY);
   else if (c == 'b') YKEventSet(charEvent, EVENT_B_KEY);
   else if (c == 'c') YKEventSet(charEvent, EVENT_C_KEY);
   else if (c == 'd') YKEventSet(charEvent, EVENT_A_KEY
                                 | EVENT_B_KEY | EVENT_C_KEY);
   else if (c == '1') YKEventSet(numEvent, EVENT_1_KEY);
   else if (c == '2') YKEventSet(numEvent, EVENT_2_KEY);
   else if (c == '3') YKEventSet(numEvent, EVENT_3_KEY);
   else {
      print("\nKEYPRESS (",11);
      printChar(c);
      print(") IGNORED\n",10);
   }
}
```


Lab 7 application code

```
void CharTask(void) /* waits for any events triggered by letter keys */
{
   unsigned events;
   printString("Started CharTask (2)\n");
   while (1) {
      events = YKEventPend(charEvent, EVENT_A_KEY |
                           EVENT_B_KEY | EVENT_C_KEY, EVENT_WAIT_ANY);
      if (events == 0) {
         printString("Oops! At least one event should be set in return value!\n");
      }
      if (events & EVENT_A_KEY) {
         printString("CharTask (A)\n");
         YKEventReset(charEvent, EVENT_A_KEY);
      }
      if (events & EVENT_B_KEY) {
         printString("CharTask (B)\n");
         YKEventReset(charEvent, EVENT_B_KEY);
      }
      if (events & EVENT_C_KEY) {
         printString("CharTask (C)\n");
         YKEventReset(charEvent, EVENT_C_KEY);
      }
   }
}
```


Lab 7 application code

```
void AllCharsTask(void)    /* waits for all events triggered by letter keys */
{
   unsigned events;
   printString("Started AllCharsTask (3)\n");
   while(1) {
      events = YKEventPend(charEvent, EVENT_A_KEY | EVENT_B_KEY | EVENT_C_KEY,
                           EVENT_WAIT_ALL);
      // events will be reset by other task

      if (events != 0) {
         printString("Oops! Char events weren't reset by CharTask!\n");
      }
      printString("AllCharsTask (D)\n");
   }
}

void AllNumsTask(void)    /* waits for events triggered by number keys */
{
   unsigned events;
   printString("Started AllNumsTask  (1)\n");
   while(1) {
      events = YKEventPend(numEvent, EVENT_1_KEY | EVENT_2_KEY | EVENT_3_KEY,
                           EVENT_WAIT_ALL);

      if (events != (EVENT_1_KEY | EVENT_2_KEY | EVENT_3_KEY)) {
         printString("Oops! All events should be set in return value!\n");
      }

      printString("AllNumsTask  (123)\n");
      YKEventReset(numEvent, EVENT_1_KEY | EVENT_2_KEY | EVENT_3_KEY);
   }
}
```


Lab 7 application code

```
void STask(void)        /* tracks statistics */
{
   unsigned max, switchCount, idleCount;
   int tmp;

   YKDelayTask(1);
   printString("Welcome to the YAK kernel!\n");
   printString("Determining CPU capacity\n");
   YKDelayTask(1);
   YKIdleCount = 0;
   YKDelayTask(5);
   max = YKIdleCount / 25;
   YKIdleCount = 0;

   YKNewTask(CharTask, (void *) &CharTaskStk[TASK_STACK_SIZE], 2);
   YKNewTask(AllNumsTask, (void *) &AllNumsTaskStk[TASK_STACK_SIZE], 1);
   YKNewTask(AllCharsTask, (void *) &AllCharsTaskStk[TASK_STACK_SIZE], 3);
   while (1)  {
      YKDelayTask(20);
      YKEnterMutex();
      switchCount = YKCtxSwCount;
      idleCount = YKIdleCount;
      YKExitMutex();

      printString("<<<<< Context switches: ");
      printInt((int)switchCount);
      printString(", CPU usage: ");
      tmp = (int) (idleCount/max);
      printInt(100-tmp);
      printString("% >>>>>\r\n");

      YKEnterMutex();
      YKCtxSwCount = 0;
      YKIdleCount = 0;
      YKExitMutex();
   }
}
```


Example output