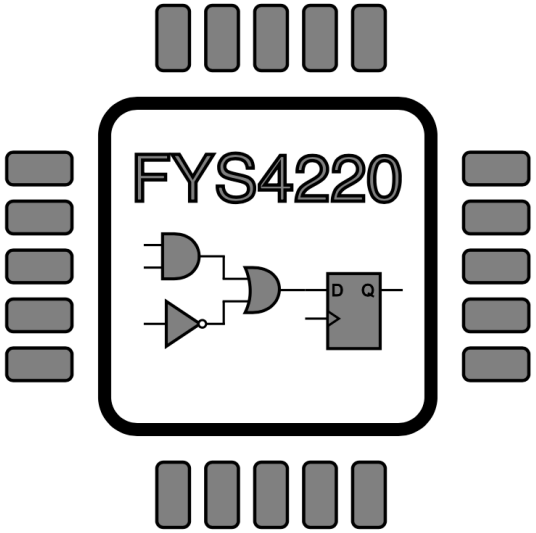
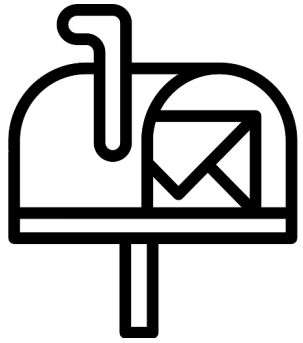
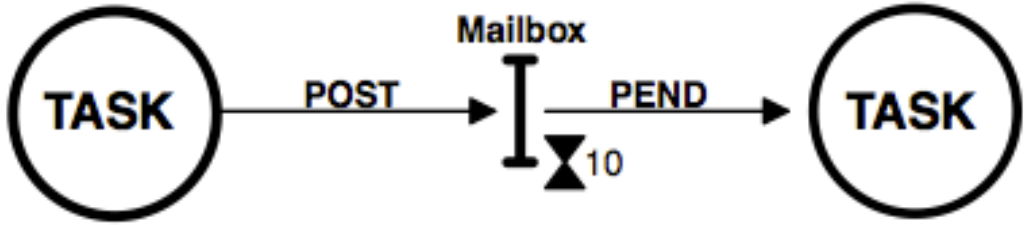


RTOS: Mailbox



Message mailbox

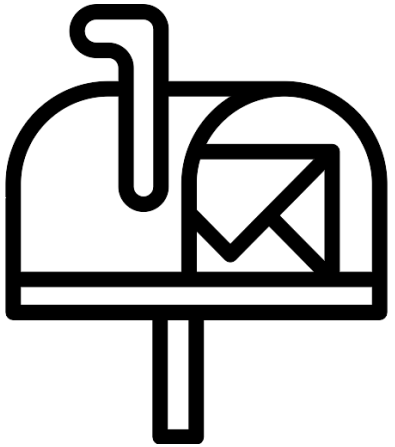
- Tasks can also communicate by sending messages via mailboxes



- Mutual exclusion of the mailbox is handled by the operating system

Message mailboxes

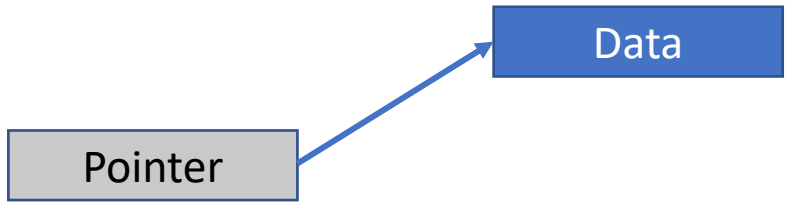
- A mailbox is a special memory location that one or more tasks can use to transfer data, or more generally for synchronization.
- The tasks rely on the kernel to allow them to
  - write to the mailbox via a post operation
  - Or read from it via a pend operation
- Direct access to any mailbox is not allowed
- A mailbox can only contain one message



Mailboxes

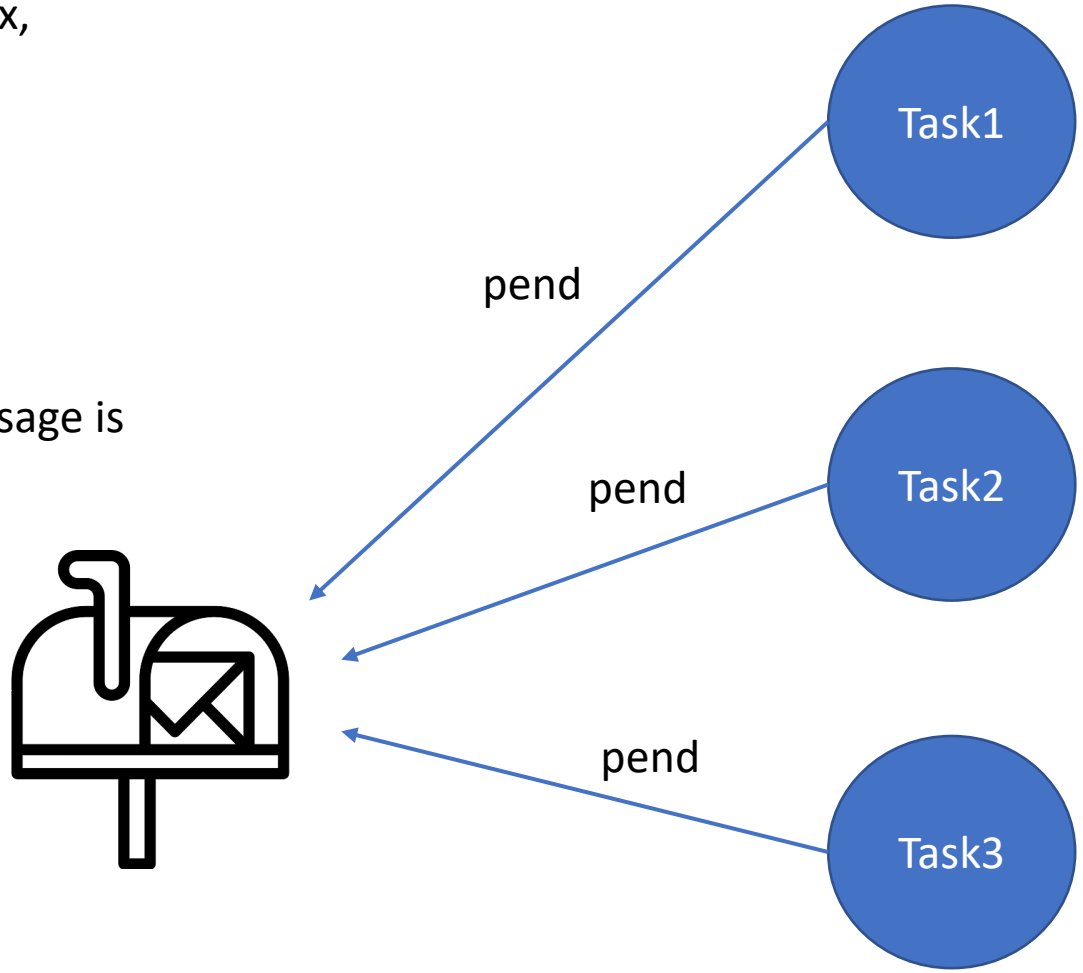
- The important difference between the pend operation and simply polling the mailbox location is that the pending task is suspended while waiting for the data to appear. (no CPU time is wasted for polling the mailbox)

- The mail that is passed via the mailbox can be
  - a single piece of data,
  - or a pointer to a data structure



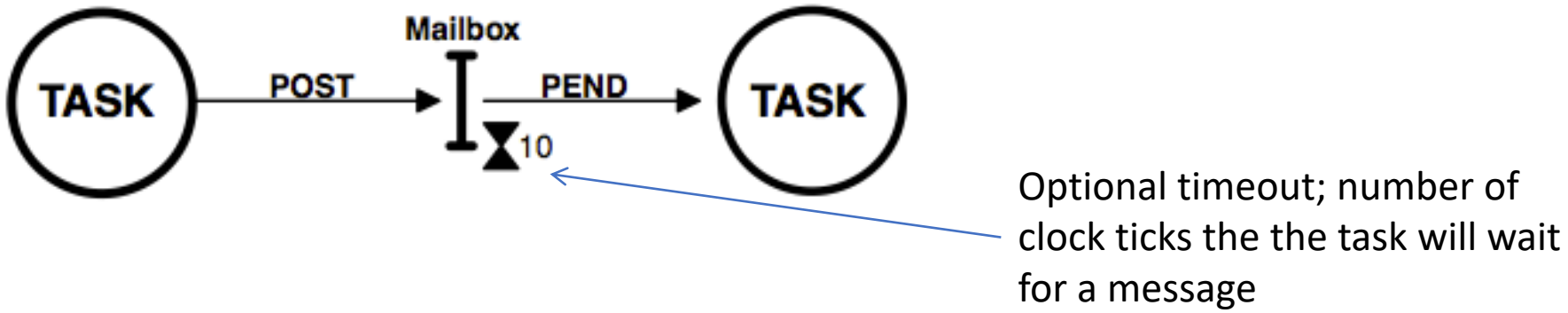
- Although several tasks can pend on the same mailbox,
- only one task can receive the message
- except for broadcast mode offered by uC/OS-II

- A waiting list is associated with each mailbox
- A task desiring a message from an empty mailbox is suspended and placed on the waiting list until a message is received.

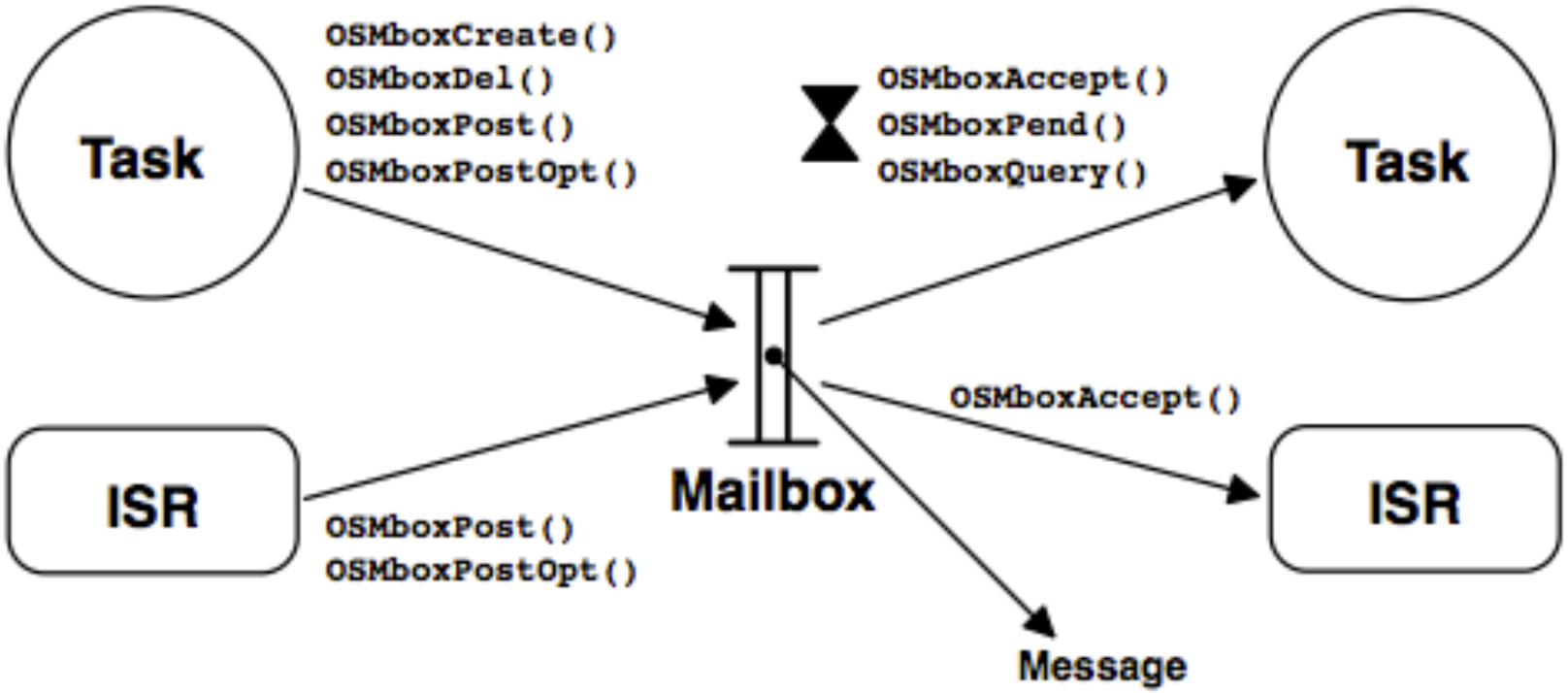


Mailboxes

- Generally, three types of operations can be performed on a mailbox
  - Initialize (with or without a message)
  - Deposit a message (POST)
  - Wait for a message (PEND)



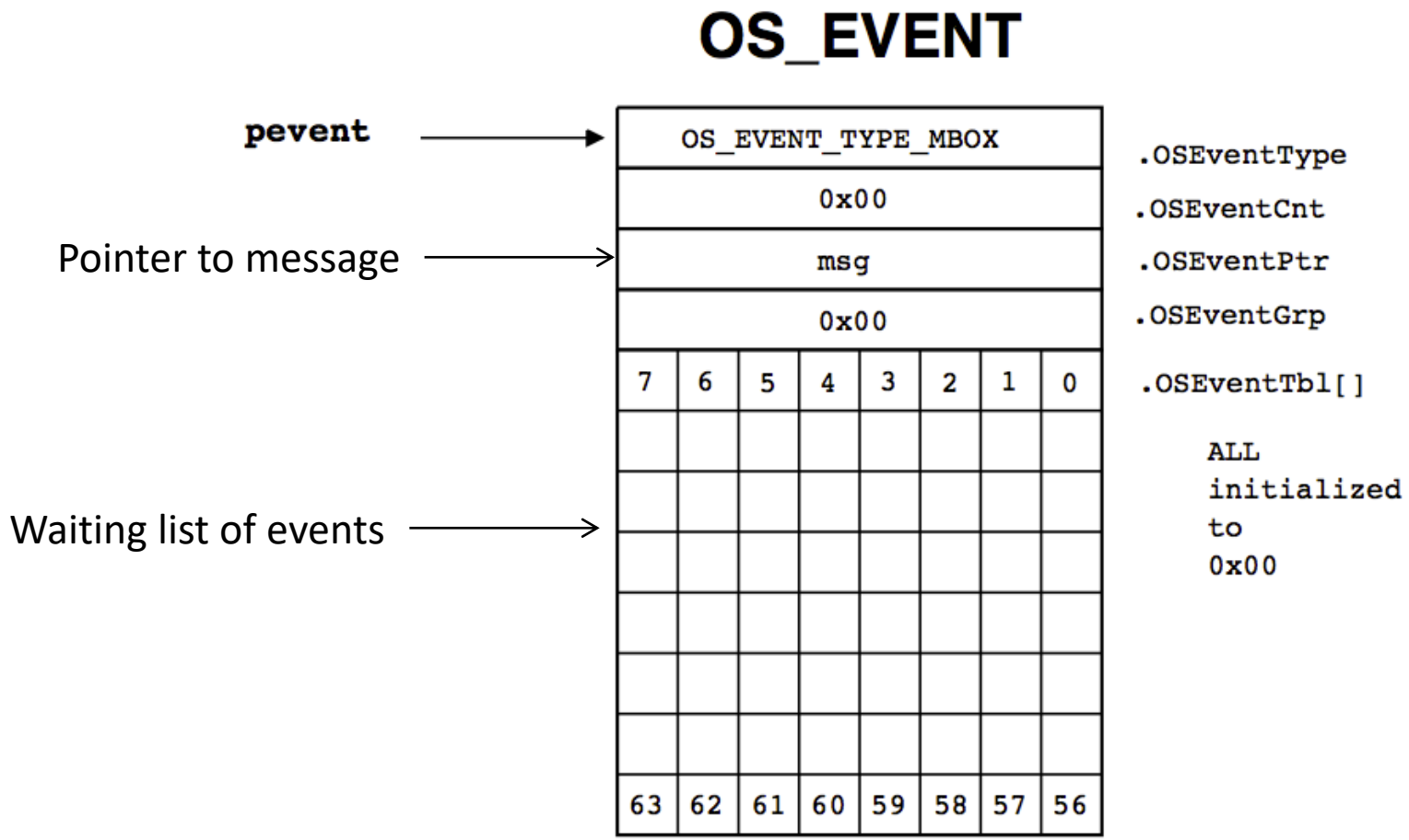
Relationship between task, ISR and mailbox



Mailbox functions in uc/OS-II

- OS_EVENT *OSMboxCreate(void *msg)
- void *OSMboxPend(OS_EVENT *pevent, INT16U timeout, INT8U *err)
  - Timeout: integral #ticks (0: wait forever)
- INT8U OSMboxPost(OS_EVENT *pevent, void *msg)
- INT8U OSMboxPostOpt(OS_EVENT *pevent, void *msg, INT8U opt)
  - Allows posting of a message to all tasks (i.e. OS_POST_OPT_BROADCAST) waiting on the mailbox

Data structure used for mailboxes



Mailbox example for uC/OS-II

```
#include <stdio.h>
#include "includes.h"

/* Definition of Task Stacks */
#define   TASK_STACKSIZE       2048
OS_STK    task1_stk[TASK_STACKSIZE];
OS_STK    task2_stk[TASK_STACKSIZE];
OS_STK    task3_stk[TASK_STACKSIZE];

/* Definition of Task Priorities */
#define TASK1_PRIORITY      6
#define TASK2_PRIORITY      7
#define TASK3_PRIORITY      8

//Semaphore to protect jtag uart
OS_EVENT *shared_jtag_sem;

//Message mailbox OS_EVENT structure
OS_EVENT *MSG_box;
```

```c
int main(void)
{
  //Initialize uc/OS-II
  OSInit();

  printf("========================================\n");
  printf("  Starting mailbox example\n");
  printf("========================================\n");

  //Create semaphore to protect jtag uart
  shared_jtag_sem = OSSemCreate(1);
  //Create an empty mailbox
  msg_box = OSMboxCreate((void*)NULL);

  //Create the various tasks
  OSTaskCreateExt(task1,
                  NULL,
                  &task1_stk[TASK_STACKSIZE-1],
                  TASK1_PRIORITY,
                  TASK1_PRIORITY,
                  &task1_stk[0],
                  TASK_STACKSIZE,
                  NULL,
                  OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR
                  );

  OSTaskCreateExt(task2,
                  NULL,
                  &task2_stk[TASK_STACKSIZE-1],
                  TASK2_PRIORITY,
                  TASK2_PRIORITY,
                  &task2_stk[0],
                  TASK_STACKSIZE,
                  NULL,
                  OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR
                  );

  OSTaskCreateExt(task3,
                  NULL,
                  &task3_stk[TASK_STACKSIZE-1],
                  TASK3_PRIORITY,
                  TASK3_PRIORITY,
                  &task3_stk[0],
                  TASK_STACKSIZE,
                  NULL,
                  OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR
                  );

  //Start multitasking under ucosii
  OSStart();

  return 0;
}
```

Mailbox example for uC/OS-II

Sending task

```
void task1(void* pdata)
{
  INT8U error_code = OS_NO_ERR;
  int t1;

  while (1)
  {
    t1 = OSTimeGet();
    OSSemPend(shared_jtag_sem,0,&error_code);
    printf("Task1 sending message: %d ms\n",t1);
    OSSemPost(shared_jtag_sem);
    //Post the message with broadcast to all pending tasks
    error_code = OSMboxPostOpt(MSG_box,(void *)&t1,OS_POST_OPT_BROADCAST);
    OSTimeDlyHMSM(0, 0, 1, 0);
  }
}
```

Receiving task

(Similar for task3)

```
void task2(void* pdata)
{
  INT8U error_code = OS_NO_ERR;
  int t1;
  int *msg_rx;

  while (1)
  {
    //Pend messages sent from task1
    msg_rx = (int*)OSMboxPend(MSG_box,0,&error_code);
    t1 = OSTimeGet();

    OSSemPend(shared_jtag_sem,0,&error_code);
    printf("Task2 received message: %d ms (at %d ms)\n",*msg_rx ,t1);
    OSSemPost(shared_jtag_sem);
  }
}
```

Running application with broadcast

```
Task1 sending message: 1071 ms
Task2 received message: 1071 ms (at 1076 ms)
Task3 received message: 1071 ms (at 1079 ms)
Task1 sending message: 2076 ms
Task2 received message: 2076 ms (at 2081 ms)
Task3 received message: 2076 ms (at 2083 ms)
```

Running application w/o broadcast

```
Task1 sending message: 1097 ms
Task2 received message: 1097 ms (at 1100 ms)
Task1 sending message: 2100 ms
Task2 received message: 2100 ms (at 2104 ms)
```

```
error_code = OSMboxPost(MSG_box,(void *)&t1);
```

Mailbox example for uC/OS-II

- OSMboxPend returns a pointer to the message sent through the mailbox
- If that messages is updated before the receiving task has processed the message, the data will be overwritten
- Solution: create a local copy of the received message

```
Task1 sending message: 171 ms
Task2 received message: 171 ms (at 174 ms)
Task1 sending message: 274 ms
Task2 received message: 274 ms (at 277 ms)
Task3 received message: 274 ms (at 176 ms)
```

```
Task1 sending message: 170 ms
Task2 received message: 170 ms (at 173 ms)
Task1 sending message: 273 ms
Task2 received message: 273 ms (at 276 ms)
Task3 received message: 170 ms (at 175 ms)
```

```c
void task3(void* pdata)
{
  INT8U error_code = OS_NO_ERR;
  int t1;
  int *msg_rx;
  int msg_local;
  while (1)
  {
    msg_rx = (int*)OSMboxPend(MSG_box,0,&error_code);
    msg_local =  *msg_rx;
    t1 = OSTimeGet();
    usleep(110000);
    OSSemPend(shared_jtag_sem,0,&error_code);
    printf("Task3 received message: %d ms (at %d ms)\n",msg_local ,t1);
    OSSemPost(shared_jtag_sem);
  }
}
```