Operating systems and concurrency (B12)

David Kendall

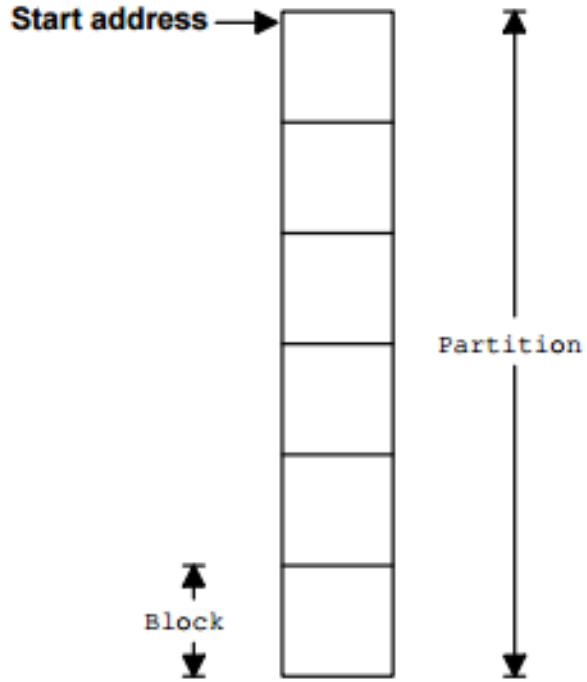
Northumbria University

Memory management

- uCOS-II provides some simple support for deterministic memory management
- It is possible to use `malloc` and `free` to manage the allocation and deallocation of memory on the heap but this is not always desirable.
- The use of `malloc` and `free` can lead to *memory fragmentation*, ie. free memory becomes fragmented into numerous, small-sized, *non-contiguous* blocks of memory.
- As memory becomes fragmented, it becomes difficult to obtain a large, contiguous block of memory, which may be necessary for many applications.
- *Garbage collectors* can help in reorganising the free space but this is often computationally expensive, ie. takes a long time to run.
- uCOS-II provides a simpler alternative that is less flexible but reliable and low-cost computationally.

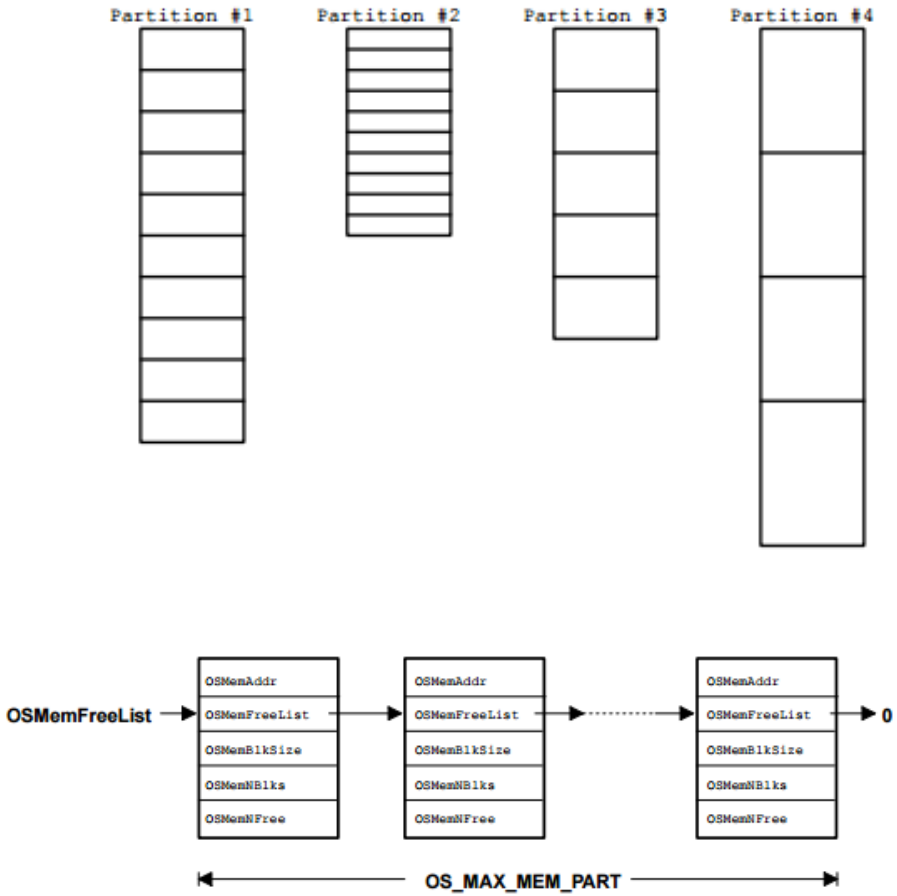
uCOS-II memory partition

- a uCOS-II partition is an area of *contiguous memory* consisting of a number of *fixed size* memory blocks.
- Memory blocks can be allocated and deallocated. When a memory block is deallocated, it is returned to the memory partition from which it was allocated.
- Memory fragmentation is eliminated by this scheme



Multiple uCOS-II memory partitions

- uCOS-II permits the use of multiple memory partitions
- Each partition can contain blocks of different sizes from the others
- This makes it possible for an application to acquire memory blocks of an appropriate size for their use
- Memory partitons are linked together in a list
- `OS_MAX_MEM_PART` determines the maximum number of memory partitions available



Creating a partition

```
OS_MEM *OSMemCreate(void *addr, INT32U nblks,
                    INT32U blksize, INT8U *err);
```

- `OSMemCreate()` returns a pointer to a memory control block for the partition (or `NULL` if the operation fails)
- `addr` is the start address of the memory partition
- `nblks` is the number of blocks that comprises the partition
- `blksize` is the size in bytes of each block
- `err` is the address of a status variable which will be used to provide an error code

```
OS_MEM *mcb;              // pointer to memory control block
INT8U buffer[100][32]; // 100 blocks of 32 bytes each
INT8U osStatus;

mcb = OSMemCreate(buffer, 100, 32, &osStatus);
```

Obtaining a memory block

```
void *OSMemGet(OS_MEM *mcb, INT8U *err);
```

- `OSMemGet()` returns a pointer to the memory block that is allocated by the call (or `NULL` if no block can be allocated)
- `mcb` is a pointer to the memory control block for the partition from which a block is requested
- `err` is the address of a status variable which will be used to provide an error code

```
INT8U *block;
INT8U osStatus;

block = (INT8U *)OSMemGet(mcb, &osStatus);
```

Returning a memory block

```
INT8U OSMemPut(OS_MEM *mcb, void *block);
```

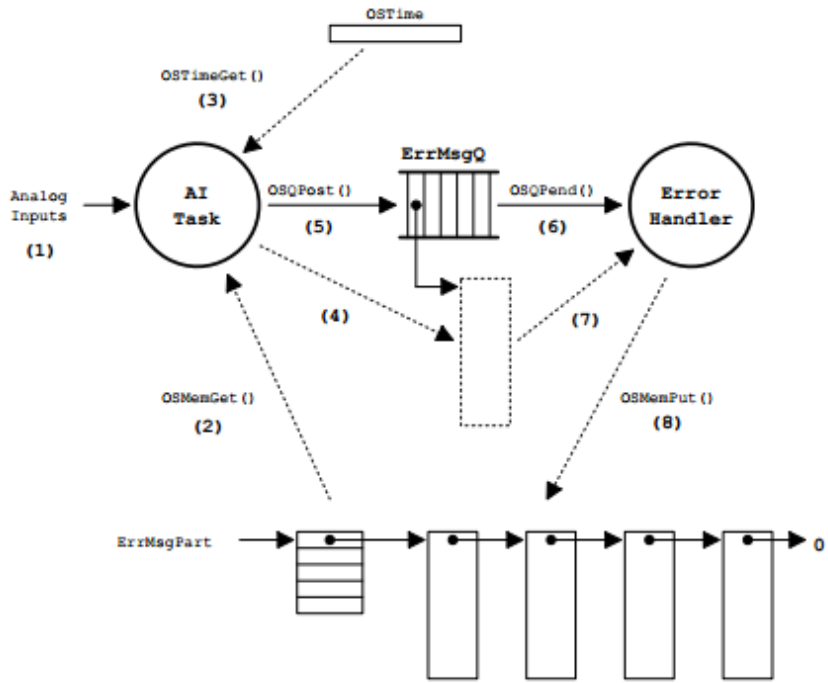
- `OSMemPut()` returns `OS_NO_ERR` if the memory block is returned successfully
- `mcb` is a pointer to the memory control block for the partition to which the block is returned
- `block` is the address of the block that should be returned
- Note that uCOS-II does not check that the partition that the block is returned to is the same one from which it was allocated. The application programmer is responsible for getting this right.

```
INT8U *block;
INT8U osStatus;

osStatus = OSMemPut(mcb, block);
```

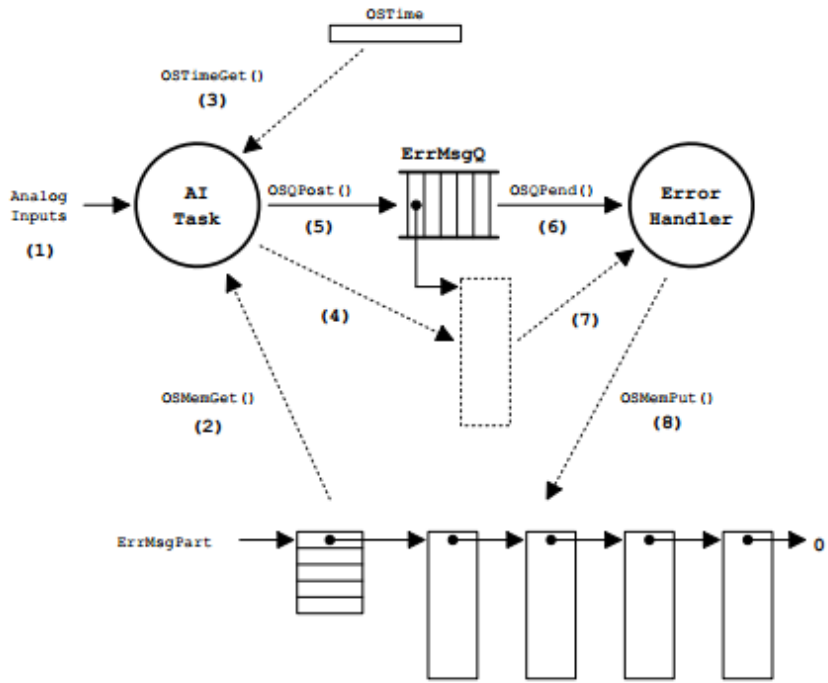
Using memory partitions with a message queue

- AI task is responsible for sampling analog inputs
- Error Handler tasks is responsible for reporting error conditions in the inputs, eg. temperature too low, pressure too high etc.
- AI task communicates with the Error Handler task using a message queue in which each message points to a memory block allocated from a memory partition
- Clean, efficient message-passing mechanism



Using memory partitions with a message queue

- Care needed to ensure that AI task does not overwrite data that is still needed by Error Handler task – use a ‘write-once’ mechanism.
- AI task gets fresh memory block, writes data into block, posts pointer to data block to queue, (does not use data block again)
- Error Handler task gets pointer to data block from queue, uses data, returns data block to memory partition



Acknowledgements

- Labrosse, J., MicroC/OS-II: The Real-time Kernel, CMP, 2002