ARM PROGRAMMING

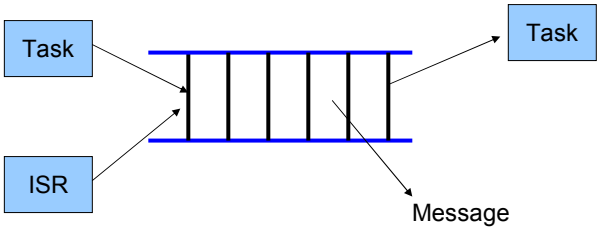
Bùi Quốc Bảo

QUEUE (hàng đợi)

- Để truyền thông tin giữa các tác vụ (intertask communication), các phương pháp sau có thể dùng:
  - Biến toàn cục
  - Queue

Queue

- Queue là một mảng các phần tử có cùng chiều dài, thường là 1 bộ đệm FIFO.
- Queue được dùng chung cho các task. Thông thường, nhiều task ghi vào queue và 1 task đọc queue.
- Data được copy vào queue



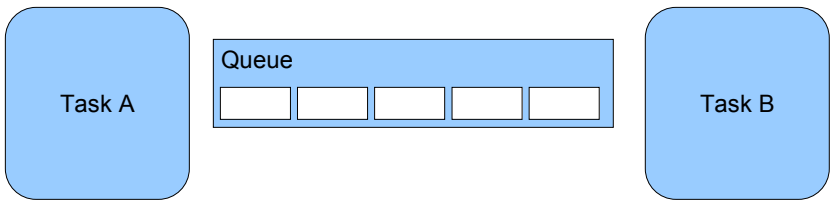
Blocking on queue reads

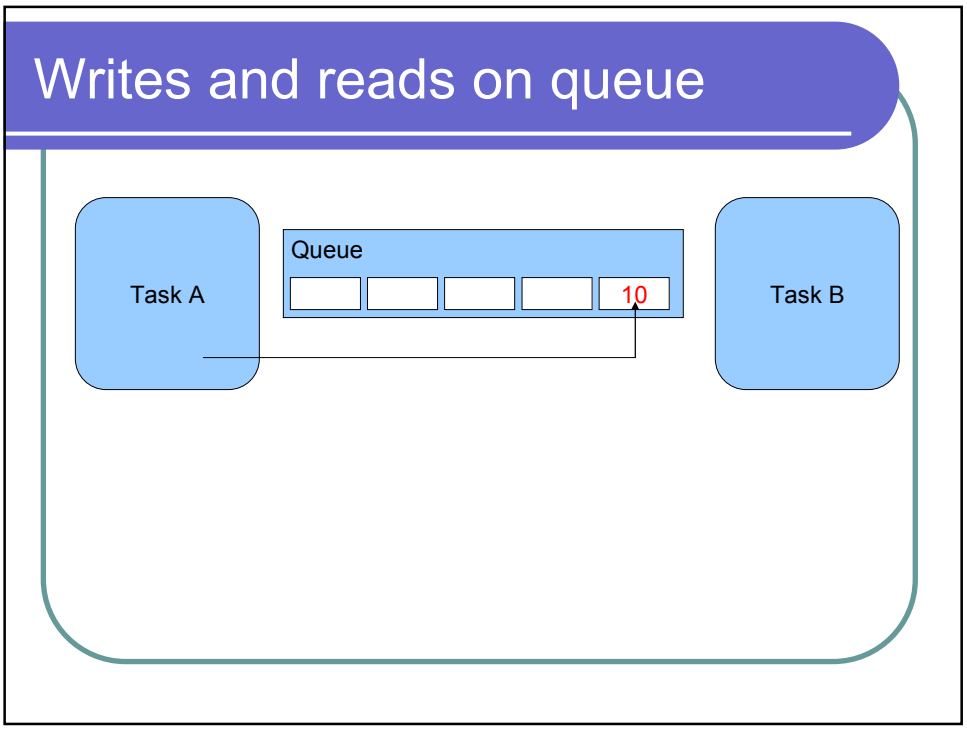
- Khi 1 task đọc queue, nó có thể chỉ định 1 thời gian “block time”.
- Nếu queue rỗng, task sẽ được đưa vào trạng thái “block” trong “block time”.
- Khi queue có dữ liệu, task sẽ được đưa vào trạng thái “ready”.

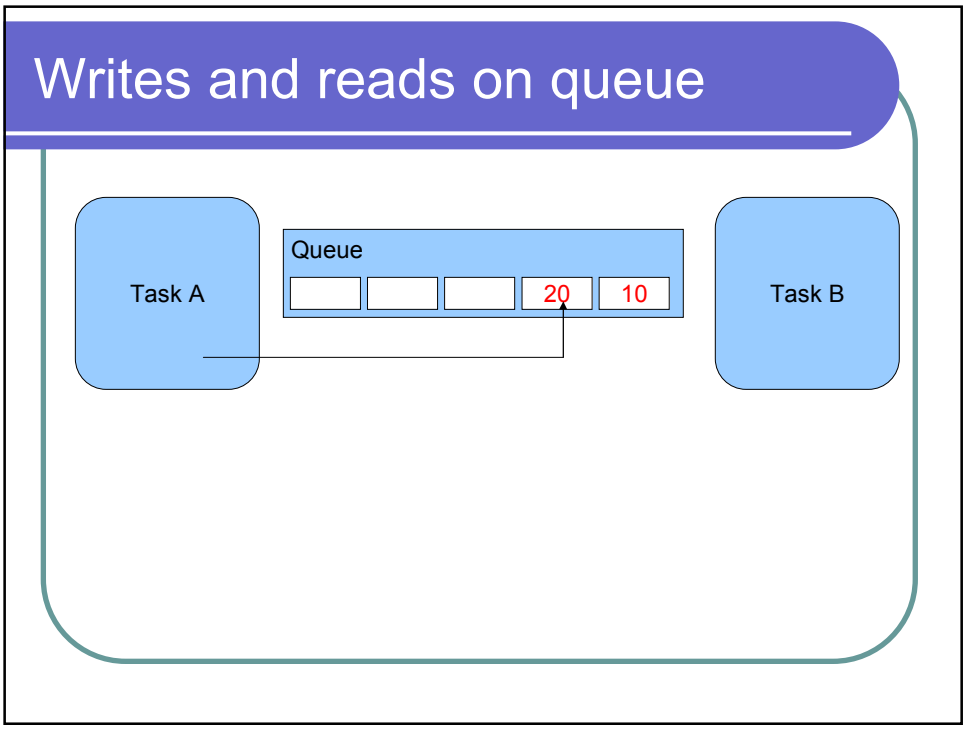
Blocking on queue writes

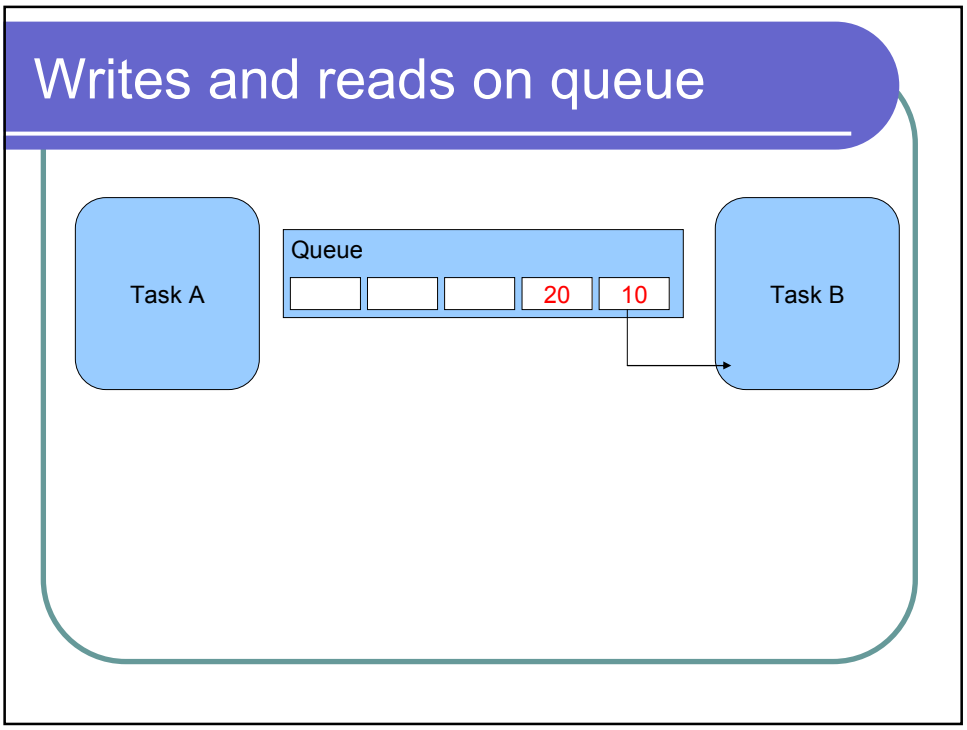
- Khi 1 task ghi vào queue, nó có thể chỉ định 1 thời gian “block time”.
- Nếu queue đầy, task sẽ được đưa vào trạng thái “block” trong “block time”.
- Khi queue có chỗ trống, task sẽ được đưa vào trạng thái “ready”.
- Nếu có nhiều task cùng đợi queue, task có mức ưu tiên cao sẽ vào trạng thái ready
- Nếu có nhiều task cùng mức ưu tiên, task nào đã đợi lâu nhất sẽ vào trang thái ready trước

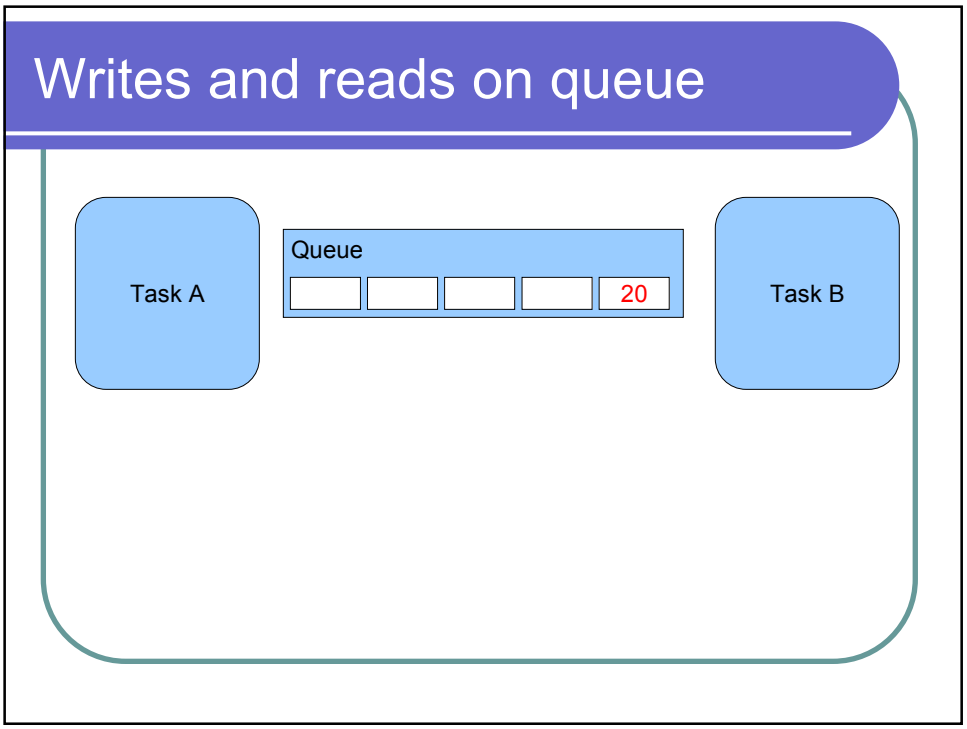
Writes and reads on queue




Writes and reads on queue
Task A
Queue
10
Task B


Writes and reads on queue
Task A
Queue
20
10
Task B


Writes and reads on queue
Task A
Queue
20
10
Task B


Writes and reads on queue
Task A
Queue
20
Task B

Example using queue

```
static void vSenderTask( void *pvParameters )
{
long lValueToSend;
portBASE_TYPE xStatus;
lValueToSend = ( long ) pvParameters;
for( ;; )
   {
         xStatus = xQueueSendToBack( xQueue, &lValueToSend, 0 );

      if( xStatus != pdPASS )
      {
         vPrintString( "Could not send to the queue.\r\n" );
      }
         taskYIELD();
   }
}
```

Example using queue

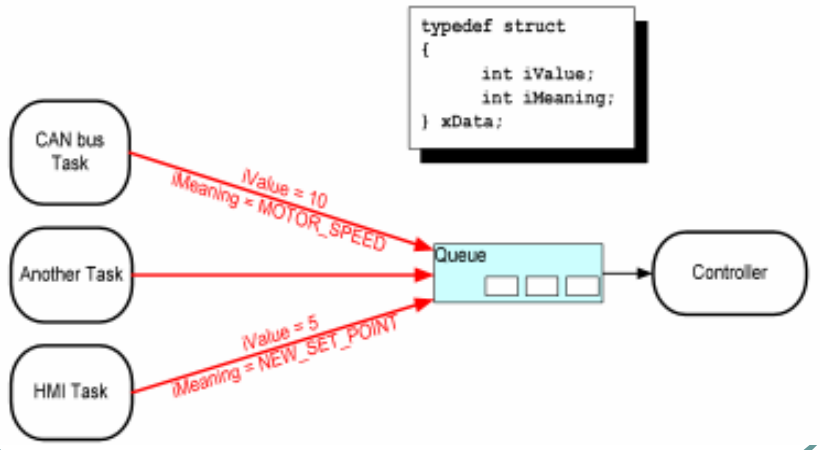
```
static void vReceiverTask( void *pvParameters )
{
long lReceivedValue;
portBASE_TYPE xStatus;
const portTickType xTicksToWait = 100 / portTICK_RATE_MS;
for( ;; ) {
if( uxQueueMessagesWaiting( xQueue ) != 0 )
      {
         vPrintString( "Queue should have been empty!\r\n" );
      }
xStatus = xQueueReceive( xQueue, &lReceivedValue, xTicksToWait );
      if( xStatus == pdPASS )
         vPrintStringAndNumber( "Received = ", lReceivedValue );
   else
         vPrintString( "Could not receive from the queue.\r\n" );
 } }
```

Example using queue

```
xQueueHandle xQueue;
int main( void )
{
xQueue = xQueueCreate( 5, sizeof( long ) );

    if( xQueue != NULL )
    {
   xTaskCreate( vSenderTask, "Sender1", 1000, ( void * ) 100, 1, NULL );
   xTaskCreate( vSenderTask, "Sender2", 1000, ( void * ) 200, 1, NULL );
   xTaskCreate( vReceiverTask, "Receiver", 1000, NULL, 2, NULL );
   vTaskStartScheduler();
    }
for( ;; );
}
```

Passing compound data



Queue Management APIs

xQueueCreate
vQueueDelete

xQueueSend
xQueueSendToBack
xQueueSendToFront
xQueueReceive
xQueueMessageWaiting
xQueuePeek

xQueueSendFromISR
xQueueSendToBackFromISR
xQueueSendToFrontFromISR
xQueueReceiveFromISR
xQueueMessageWaitingFromISR

xQueueAltSendToBack
xQueueAltSendToFront
xQueueAltReceive

Create a queue: xQueueCreate()

- xQueueHandle xQueueCreate(
  unsigned portBASE_TYPE uxQueueLength,
  unsigned portBASE_TYPE uxItemSize);
  - uxQueueLength: The maximum number of items time.
  - uxItemSize: The size in bytes of each data item
  - Return Value:
    - NULL: can not create queue.
    - A non-NULL value: handle of created queue

Delete a queue vQueueDelete

- void vQueueDelete(
  xQueueHandle xQueue)
  - xQueue: handle to the queue tobe deleted

Pass data to queue xQueueSend

- portBASE_TYPE xQueueSend(
  xQueueHandle xQueue,
  const void * pvItemToQueue,
  portTickType xTicksToWait).
  - xQueue: The handle of the queue
  - pvItemToQueue: A pointer to the data
  - xTicksToWait: The “block time”
  - Return value:
    - pdPass
    - errQUEUE_FULL

Pass data to tail of queue xQueueSendToBack

- portBASE_TYPE xQueueSendToBack(
  xQueueHandle xQueue,
  const void * pvItemToQueue,
  portTickType xTicksToWait).
  - xQueue: The handle of the queue
  - pvItemToQueue: A pointer to the data
  - xTicksToWait: The “block time”
  - Return value:
    - pdPass
    - errQUEUE_FULL

Pass data to head of queue xQueueSendToFront

- portBASE_TYPE xQueueSendToFront(
  xQueueHandle xQueue,
  const void * pvItemToQueue,
  portTickType xTicksToWait).
  - =xQueueSend with LIFO order

Destructive receiving of a message xQueueReceive

- portBASE_TYPE xQueueReceive(
xQueueHandle xQueue,
const void * pvBuffer,
portTickType xTicksToWait);

- pvBuffer: pointer to the buffer that the message will be copied
- Return value:
  - pdTrue: success
  - Otherwise: fail
- Note: the item will be removed from the queue

Non-Destructive receiving of a message xQueuePeek

- portBASE_TYPE xQueueReceive(
xQueueHandle xQueue,
const void * pvBuffer,
portTickType xTicksToWait);

- pvBuffer: pointer to the buffer that the message will be copied
- Return value:
  - pdTrue: success
  - Otherwise: fail
- Note: the item will **not** be removed from the queue

Get the number of messages in queue uxQueueMessagesWaiting

- unsigned portBASE_TYPE uxQueueMessagesWaiting(xQueueHandle xQueue);
  - xQueue The handle of the queue being queried
  - Return value: number of messages in the queue

Note

- Không được sử dụng
  - xQueueSend()
  - xQueueSendToFront()
  - xQueueSendToBack()
  - xQueueReceive()
  - xQueuePeek()
  - uxQueueMessagesWaiting()

  trong ISR