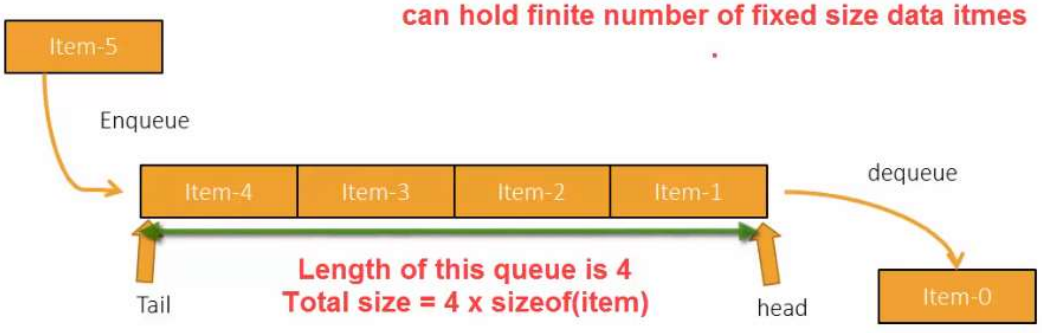
Free RTOS Part II

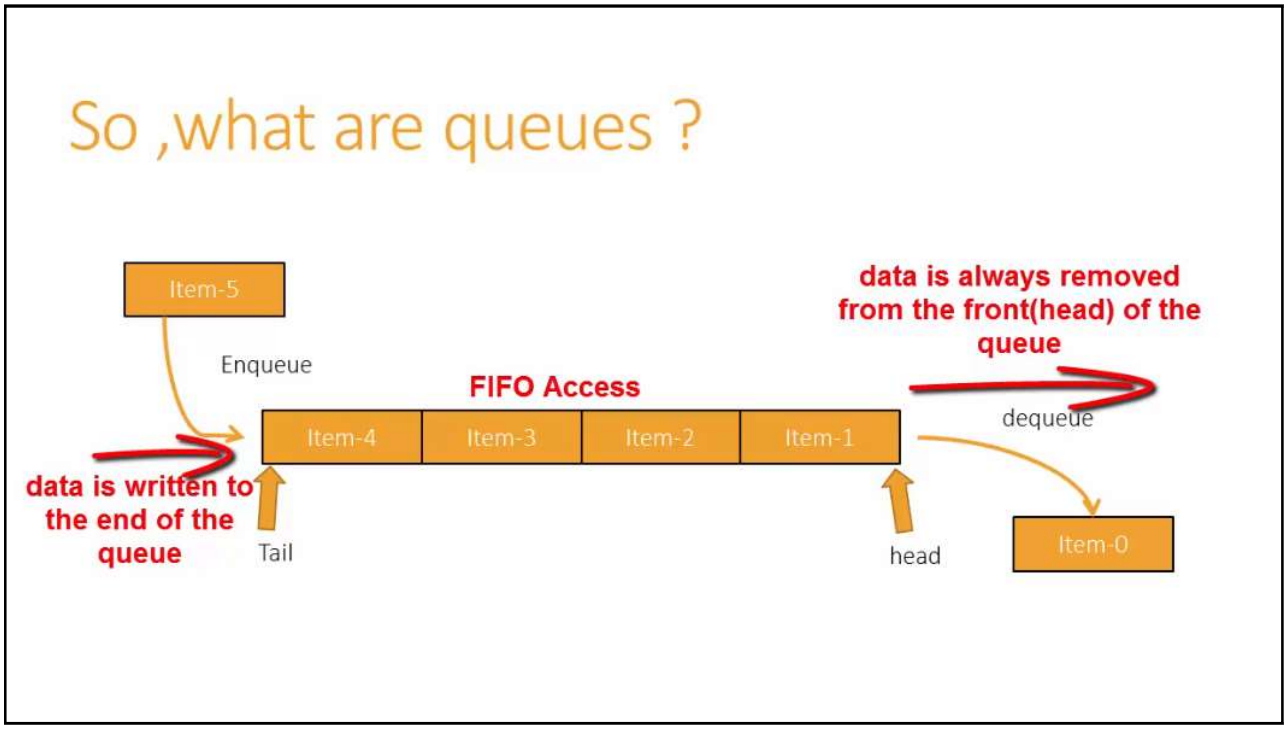
1

So ,what are queues ?

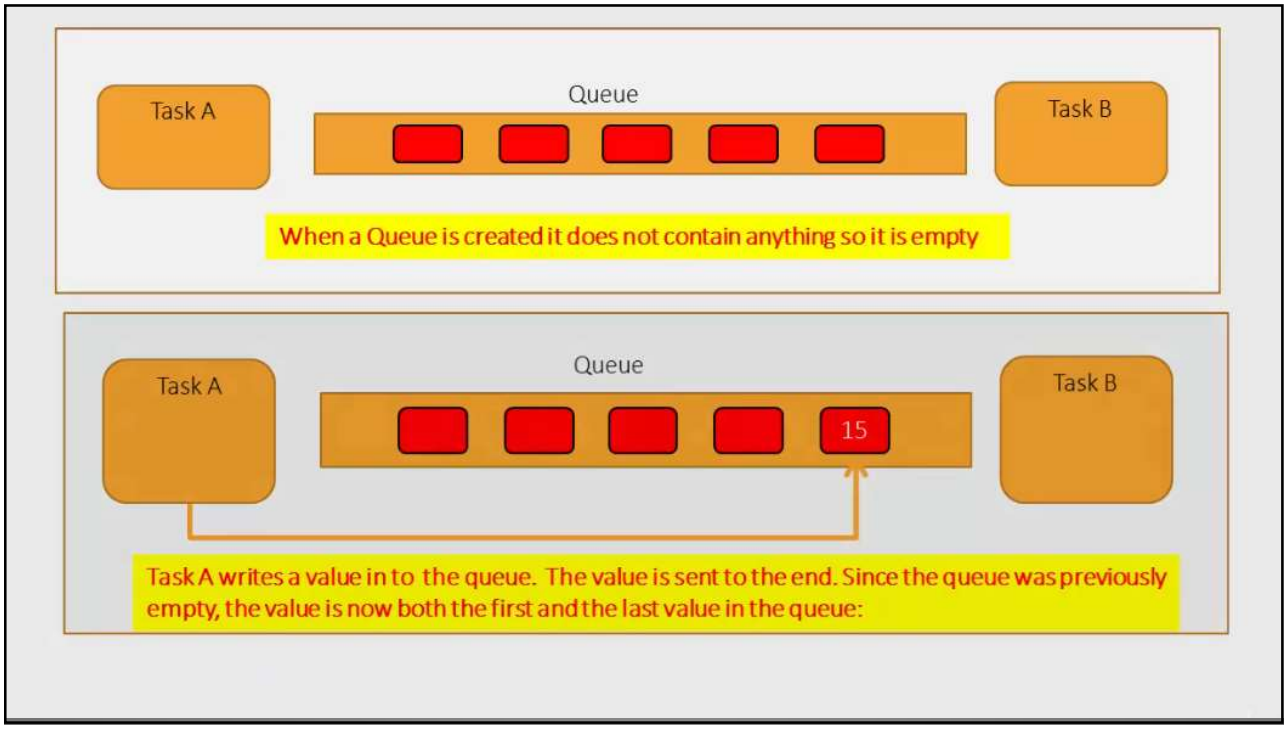
A Queue is nothing but a data structure which can hold finite number of fixed size data itmes .



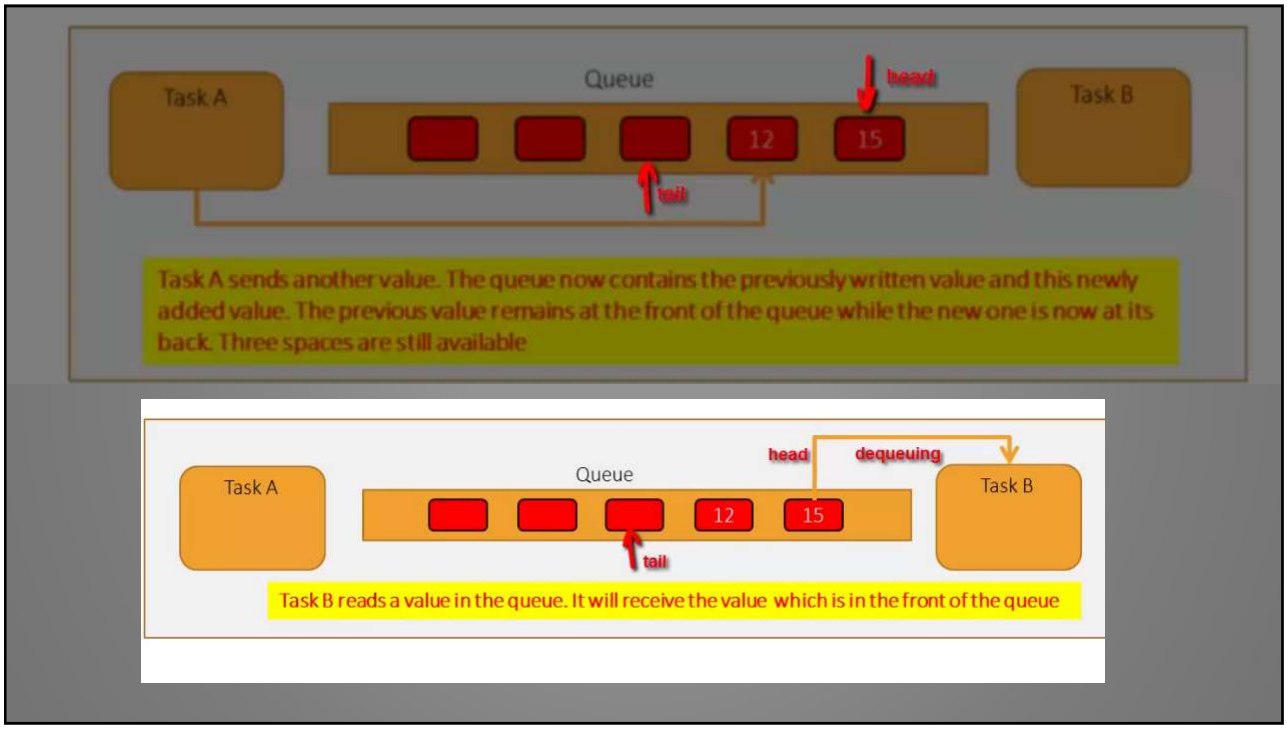
2


So ,what are queues ?
Item-5
Enqueue
FIFO Access
data is always removed from the front(head) of the queue
dequeue
Item-4
Item-3
Item-2
Item-1
data is written to the end of the queue
Tail
head
Item-0

3


Task A
Queue
Task B
When a Queue is created it does not contain anything so it is empty
Task A
Queue
Task B
15
Task A writes a value in to the queue. The value is sent to the end. Since the queue was previously empty, the value is now both the first and the last value in the queue:

4



5

FreeRTOS API to Create a Queue

This handle is then used to do lots of queue operations like writing , reading, seeking,deleting, etc

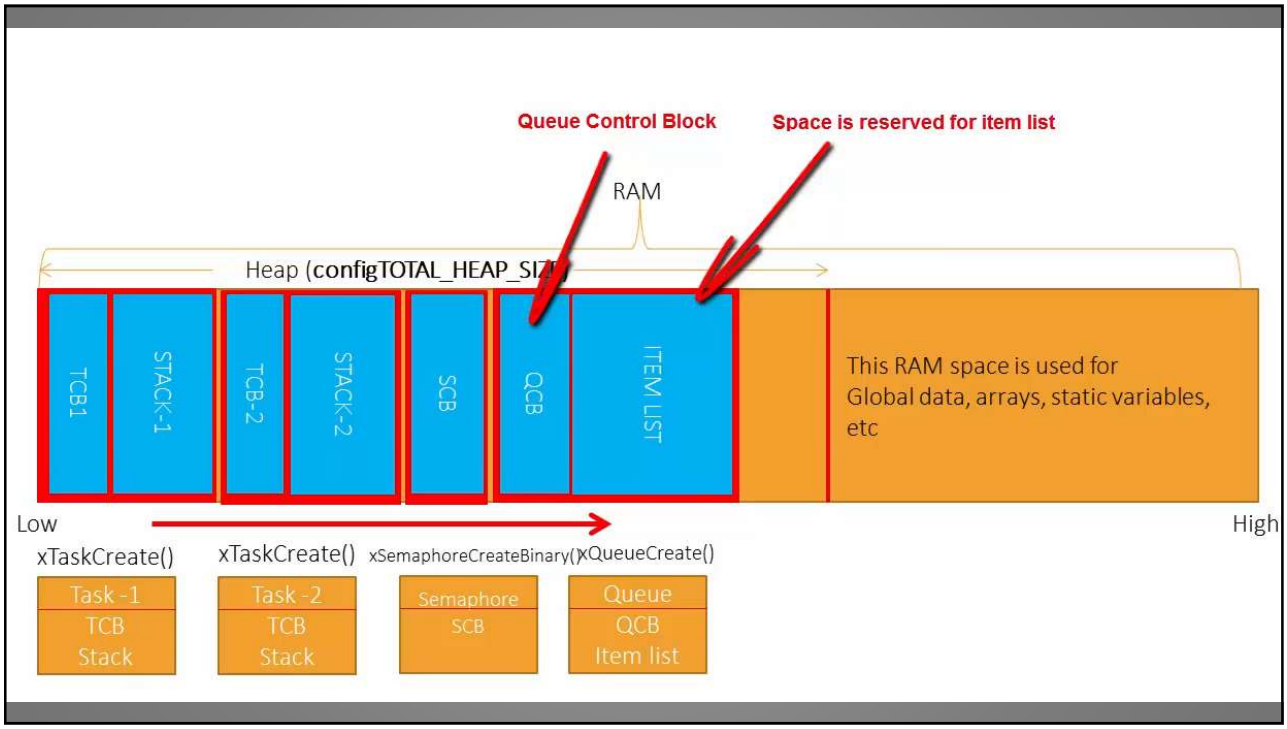
mention here, How many items this queue should hold (i.e. length)

```
xQueueHandle xQueueCreate( unsigned portBASE_TYPE uxQueueLength,
                           unsigned portBASE_TYPE uxItemSize );
```

if the creation is successful the API returns the reference(a Pointer) to the created queue otherwise NULL.

here you should mention, what's the size of single item in bytes.

6



7

```
struct AMessage
{
    char ucMessageID;
    char ucData[ 20 ];
};

void vATask( void *pvParameters )
{
QueueHandle_t xQueue1, xQueue2;

    /* Create a queue capable of containing 10 unsigned long values.
    xQueue1 = xQueueCreate( 10, sizeof( unsigned long ) );

    if( xQueue1 == NULL )
    {
        /* Queue was not created and must not be used. */
    }

    /* Create a queue capable of containing 10 pointers to AMessage
    structures.  These are to be queued by pointers as they are
    relatively large structures. */
    xQueue2 = xQueueCreate( 10, sizeof( struct AMessage * ) );

    if( xQueue2 == NULL )
    {
        /* Queue was not created and must not be used. */
    }

    /* ... Rest of task code. */
 }
```

10 chunks

4 bytes

Total Size = 40 bytes

- xQueue1 က QCB နဲ့ ITEM LISTS များကို pointer အနေနဲ့ လှမ်းသုံး။
- memory မလုံလောက်ပါက NULL return ပြန်မှာပါ။

8

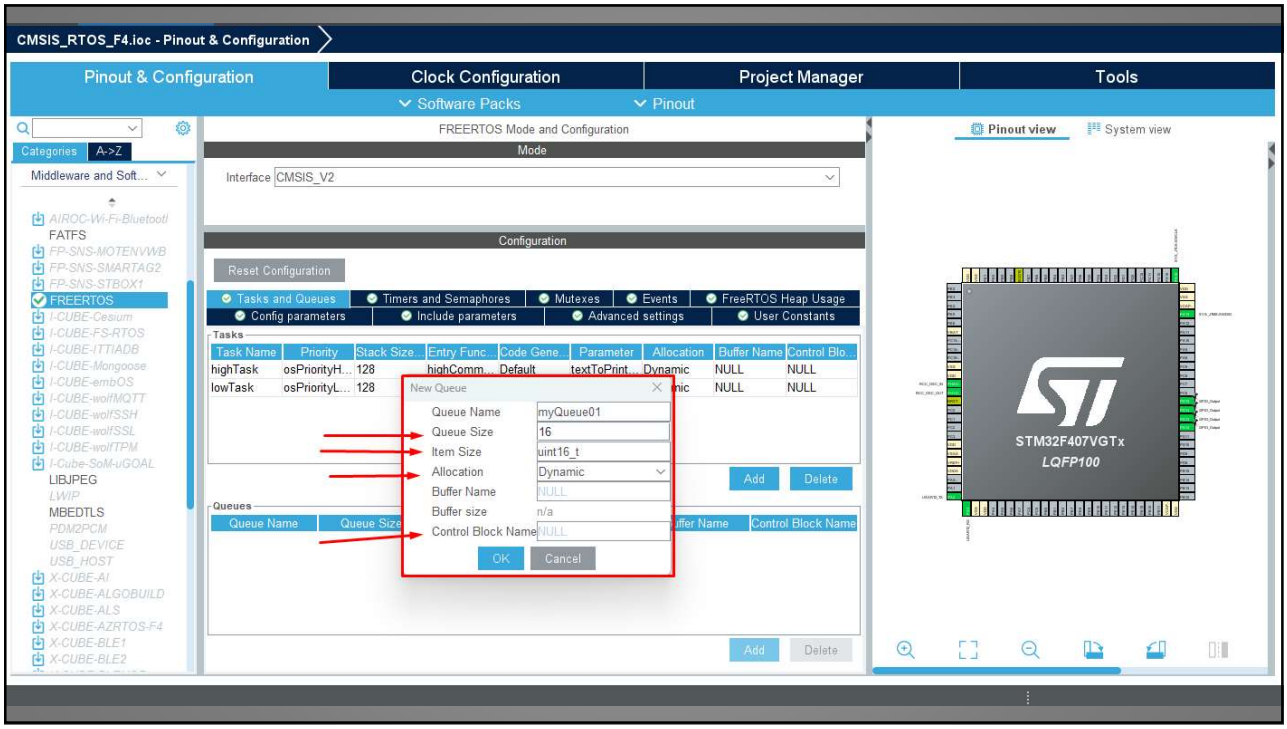
```
struct AMessage
{
    char ucMessageID;
    char ucData[ 20 ];
};
```

```
/* Create a queue capable of containing 10 pointers to AMessage
structures.  These are to be queued by pointers as they are
relatively large structures. */
xQueue2 = xQueueCreate( 10, sizeof( struct AMessage * ) );

if( xQueue2 == NULL )
{
    /* Queue was not created and must not be used. */
}

/* ... Rest of task code. */
}
```

Here, this queue will hold 10 different struct pointers instead of struct itself.

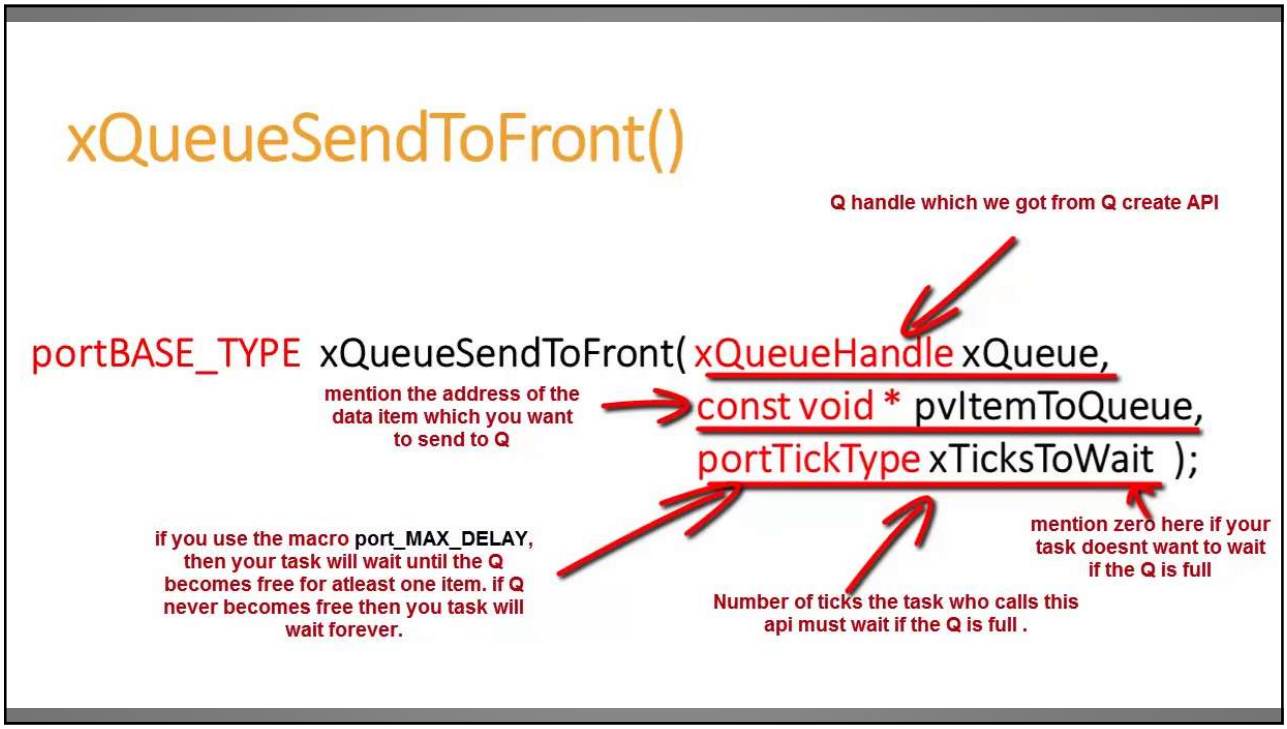
- QCB အတွက်လည်း heap ထဲက size အနဲငယ်ယူသုံးသေးတာကိုသတိပြုရန်
- AMessage Structure အသုံးပြုပြီး Queue တစ်ခုဆောက်မှာပါ။
- ပုံမှန် Structure တစ်ခုလုံးရဲ့ size က 21 bytes ဖြစ်မယ်။ Queue ဆောက်ထားတာက 10 chunks ဆိုရင် = 210 bytes ဖြစ်ပါတယ်။
- Space ကများလွန်းတဲ့အတွက်၊ Amessage Structure ကို pointer အနေနဲ့ ပြန်သုံးလိုက်ခြင်းဖြစ် Struct ရဲ့ size က 4 bytes နဲ့ 10 chunks ဆိုရင် 40 bytes ပဲပြန်ရမှာပါ။

9



10

xQueueSendToFront()



11

```
unsigned long ulVar = 10UL;

void vATask( void *pvParameters )
{
QueueHandle_t xQueue1, xQueue2;
struct AMessage *pxMessage;

    /* Create a queue capable of containing 10 unsigned long values. */
    xQueue1 = xQueueCreate( 10, sizeof( unsigned long ) );

    /* Create a queue capable of containing 10 pointers to AMessage
    structures.  These should be passed by pointer as they contain a lot of
    data. */
    xQueue2 = xQueueCreate( 10, sizeof( struct AMessage * ) );

    /* ... */

    if( xQueue1 != 0 )
    {
        /* Send an unsigned long.  Wait for 10 ticks for space to become
        available if necessary. */
        if( xQueueSendToFront( xQueue1,
                               ( void * ) &ulVar,
                               ( TickType_t ) 10 ) != pdPASS )
        {
            /* Failed to post the message, even after 10 ticks. */
        }
    }
```

12

xQueueSendToBack()

```
portBASE_TYPE xQueueSendToBack( xQueueHandle xQueue,
                                const void * pvItemToQueue,
                                portTickType xTicksToWait );
```

This API will post the data item to the back(tail) of the Q

13

xQueueReceive()

This API will Read a data item from the Q. The data item which is being read will be removed from the Q.

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

the pointer which you supply here will hold the data item which is being read.

returns TRUE if read is successfull
returns QUEUE_EMPTY if Q is Empty even after xTicksToWait

if the Q is empty, then mention here for how many number of ticks your task wishes to block or wait !

14

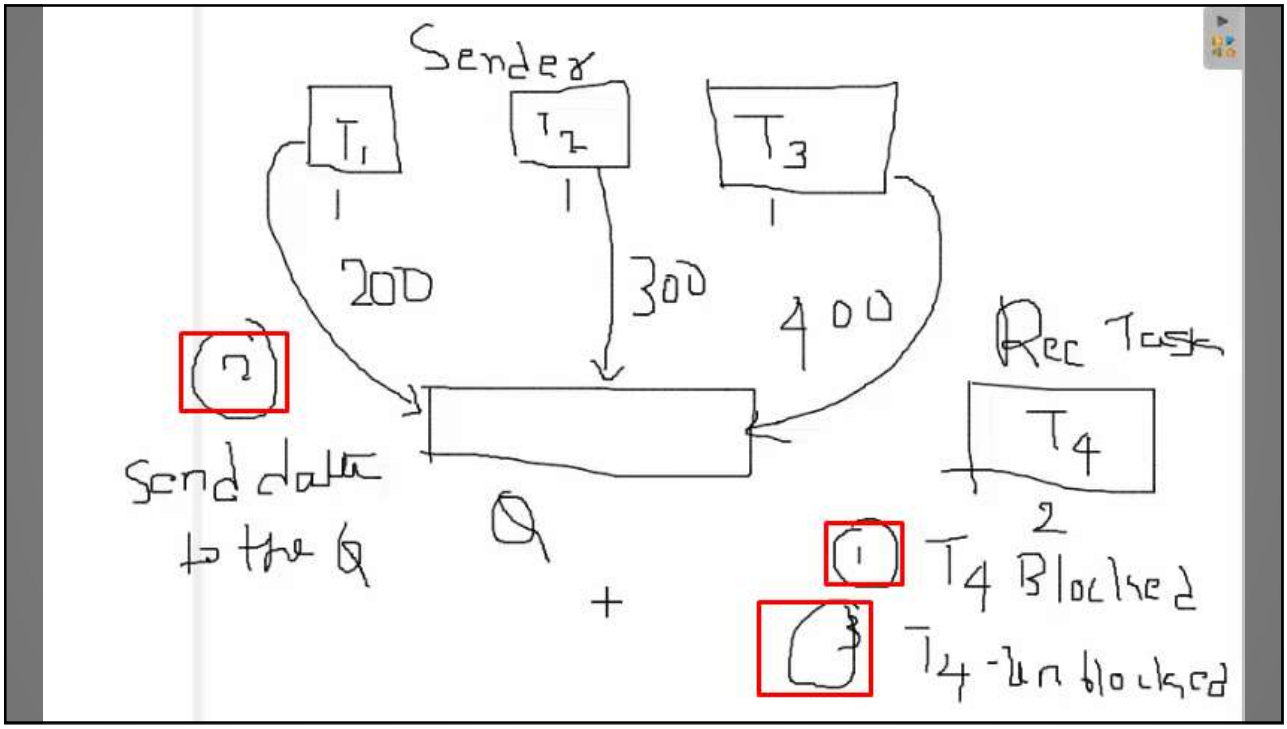
xQueuePeek()

In this case, the data item being read is not removed from the Queue

```
portBASE_TYPE xQueuePeek(xQueueHandle xQueue,
                         const void * pvBuffer,
                         portTickType xTicksToWait );
```

Receive က Queue ထဲ data ကို ထုတ်ယူသွားတာဖြစ်တယ်။
Peek က တော့ Queue ထဲက data ကို ထုတ်မယူသွားပဲ copy ကူးသွားမှာဖြစ်တယ်။

15



16

```c
/* USER CODE END Header_StartTask01 */
void SenderStartFunc(void *argument){
    long sendVal;
    osStatus_t  xStatus;

    sendVal = (long) argument;
    printf("Started put to Queue..\n");
    for(;;){
        xStatus = osMessageQueuePut(myQueue01Handle, &sendVal, 0,0);
        if (xStatus != osOK){
            printf(" Couldn't send to queue01.\r\n");
        }
        taskYIELD(); //same as portYIELD() in CMSIS-API
        BLUE_LED_TOGGLE();
        //osDelay(1);
    }
}
```

osMessageQueuePut()

osMessageQueueGet()

```c
void readQueueInit(void *argument)
{
  /* USER CODE BEGIN readQueueInit */
    long recVal;
    osStatus_t xStatus;
    const portTickType xTicksToWait = 100/portTICK_RATE_MS; //100 ms
  /* Infinite loop */
  for(;;)
  {
      printf("About to read value from Queue 01 \n");
      xStatus = osMessageQueueGet(myQueue01Handle, &recVal, NULL, xTicksToWait);
      if (xStatus == osOK){
          printf ("Received Value : %ld \n", recVal);
          GREEN_LED_TOGGLE();

      }
      else {
          printf ("Not Received ! \n");
          RED_LED_ON();
      }
    //osDelay(1);
  }
  /* USER CODE END readQueueInit */
}
```

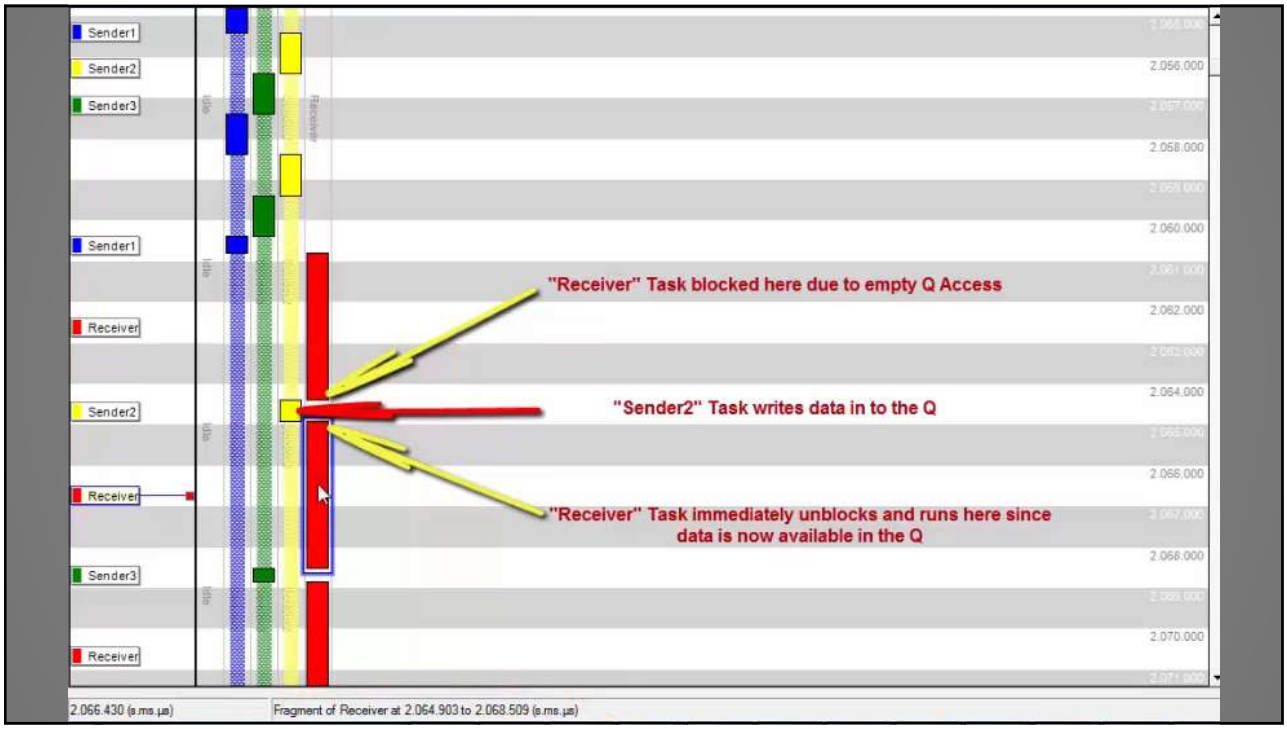
17

```c
  .stack_size = 128 * 4,
  .priority = (osPriority_t) osPriorityNormal,
};
/* Definitions for myTask02 */
osThreadId_t myTask02Handle;
const osThreadAttr_t myTask02_attributes = {
  .name = "myTask02",
  .stack_size = 128 * 4,
  .priority = (osPriority_t) osPriorityNormal,
};
/* Definitions for myTask03 */
osThreadId_t myTask03Handle;
const osThreadAttr_t myTask03_attributes = {
  .name = "myTask03",
  .stack_size = 128 * 4,
  .priority = (osPriority_t) osPriorityNormal,
};
/* Definitions for readQueue */
osThreadId_t readQueueHandle;
const osThreadAttr_t readQueue_attributes = {
  .name = "readQueue",
  .stack_size = 128 * 4,
  .priority = (osPriority_t) osPriorityHigh,
};
/* Definitions for myQueue01 */
osMessageQueueId_t myQueue01Handle;
```

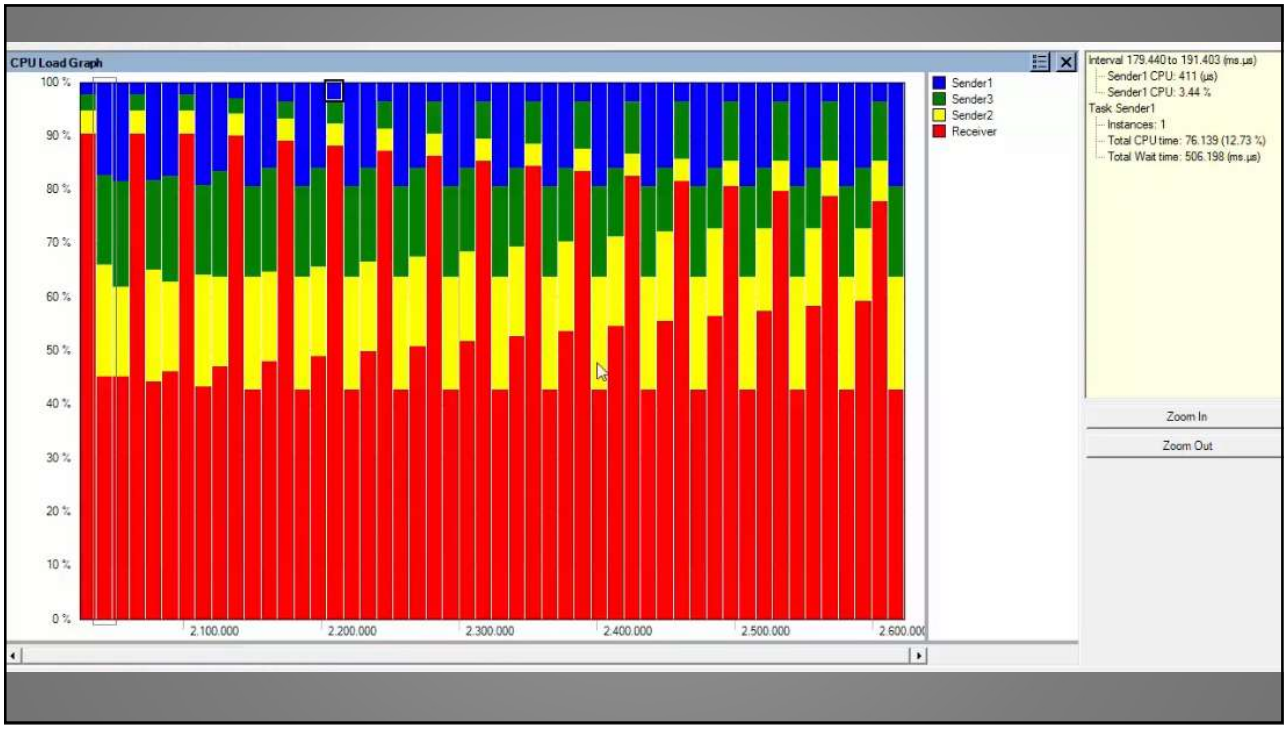
```
SWV ITM Data Console - ITM Port: 0
osKernel Initialized....
About to read value from Queue 01
Started put to Queue..
Received Value : 200
About to read value from Queue 01
Started put to Queue..
Received Value : 300
About to read value from Queue 01
Started put to Queue..
Received Value : 400
About to read value from Queue 01
Received Value : 200
About to read value from Queue 01
Received Value : 300
About to read value from Queue 01
Received Value : 400
About to read value from Queue 01
Received Value : 200
About to read value from Queue 01
Received Value : 300
About to read value from Queue 01
```

```
File download complete
Time elapsed during download operation: 00:00:00.815
```

18

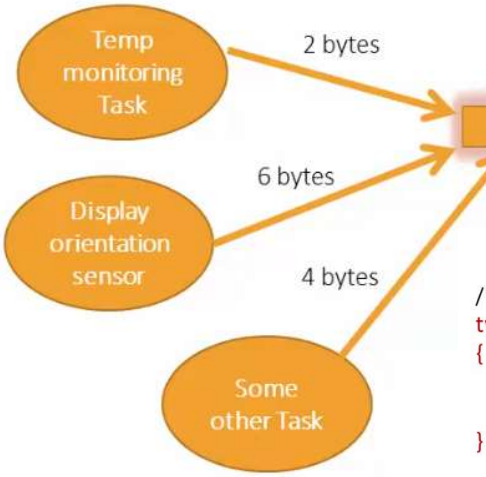

Sender1
Sender2
Sender3
Sender1
Receiver
Sender2
Receiver
Sender3
Receiver
"Receiver" Task blocked here due to empty Q Access
"Sender2" Task writes data in to the Q
"Receiver" Task immediately unblocks and runs here since data is now available in the Q
2.056.000
2.058.000
2.060.000
2.062.000
2.064.000
2.066.000
2.068.000
2.070.000
2.066.430 (s.ms.µs)
Fragment of Receiver at 2.064.903 to 2.068.509 (s.ms.µs)

19


CPU Load Graph
100 %
90 %
80 %
70 %
60 %
50 %
40 %
30 %
20 %
10 %
0 %
2.100.000
2.200.000
2.300.000
2.400.000
2.500.000
2.600.000
Sender1
Sender3
Sender2
Receiver
Interval 179.440 to 191.403 (ms.µs)
Sender1 CPU: 411 (µs)
Sender1 CPU: 3.44 %
Task: Sender1
Instances: 1
Total CPU time: 76.139 (12.73 %)
Total Wait time: 506.198 (ms.µs)
Zoom In
Zoom Out

20

Scenario of Task receiving data from multiple sources on a single queue




```
/* Define the generic structure used by different tasks to send their data to queue. */
typedef struct
{
    unsigned char code;
    unsigned char data[MAX_LEN];
} xData;
```

21

Structure နှင့် Pointer သုံးပြီး စမ်းသပ်ရန် ကျန်ရှိ

22

23