```c
/*
 * @brief FreeRTOS Blinky example
 *
 * @note
 * Copyright(C) NXP Semiconductors, 2014
 * All rights reserved.
 *
 * @par
 * Software that is described herein is for illustrative purposes only
 * which provides customers with programming information regarding the
 * LPC products.  This software is supplied "AS IS" without any warranties of
 * any kind, and NXP Semiconductors and its licensor disclaim any and
 * all warranties, express or implied, including all implied warranties of
 * merchantability, fitness for a particular purpose and non-infringement of
 * intellectual property rights.  NXP Semiconductors assumes no responsibility
 * or liability for the use of the software, conveys no license or rights under any
 * patent, copyright, mask work right, or any other intellectual property rights in
 * or to any products. NXP Semiconductors reserves the right to make changes
 * in the software without notification. NXP Semiconductors also makes no
 * representation or warranty that such application will be suitable for the
 * specified use without further testing or modification.
 *
 * @par
 * Permission to use, copy, modify, and distribute this software and its
 * documentation is hereby granted, under NXP Semiconductors' and its
 * licensor's relevant copyrights in the software, without fee, provided that it
 * is used in conjunction with NXP Semiconductors microcontrollers.  This
 * copyright, permission, and disclaimer notice must appear in all copies of
 * this code.
 */

#include "board.h"
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"


#include "FreeRTOSConfig.h"
xQueueHandle Q;
void tasksender(void *pvParameters)
{
    char M = '0';
    portBASE_TYPE status = '0';
    while(1)
    {

        M= (int) pvParameters;
        status = xQueueSendToBack(Q, &M, 0);
        if(status == errQUEUE_FULL)
        {
        vPrintString("Could not send to Queue.\r\n");
        }
        taskYIELD();
    }
}

void taskreceiver(void *pvParameters)
{
    char RM= '0';
    portTickType timeToWait = 150 / portTICK_RATE_MS;
    portBASE_TYPE status = '0';
    while(1)
    {
        if(uxQueueMessagesWaiting(Q) != 0)
        {
            vPrintString("Could not send to Queue.\r\n");
        }
        status = xQueueReceive(Q, &RM, timeToWait);
        if(status == pdPASS)
        {
            if(RM == 0x02) {

            } else if(RM== 0x03) {

            }
        } else if(status == errQUEUE_EMPTY){


    }
}

}
int main(void)
{
    Q = xQueueCreate(1, sizeof(char));
    if(Q!= NULL) {

        char S1 = 0x02;
        char S2 = 0x03;

    xTaskCreate(tasksender, NULL, 100, &S1, 1, NULL);
    xTaskCreate(tasksender, NULL, 100, &S2, 1, NULL);
    xTaskCreate(taskreceiver, NULL, 100, NULL, 2, NULL);

    vTaskStartScheduler();
}
return 0;

}
```