

Système embarqué & Programmation RTOS

■ ■ ■ Programmation RTOS

1 – Que fait le programme suivant ?

```
#define LED_BUILTIN_PIN 4

TimerHandle_t timerLed, timerTrace;

int ledState = 1;

void toggleLed( TimerHandle_t xTimer )
{
    ledState = ledState ^ 1;
    digitalWrite( LED_BUILTIN_PIN, ledState );
}

void traceCount( TimerHandle_t xTimer )
{
    int timerCount = ( int ) pvTimerGetTimerID( xTimer );
    Serial.printf("timerCount = %08d \n\r", timerCount++ );
    vTimerSetTimerID( xTimer, ( void * ) timerCount );
}

void setup( void )
{
    pinMode(LED_BUILTIN_PIN, OUTPUT);
    Serial.begin(115200);
    sleep( 2 );
    Serial.printf( "[ Timers ] \n\n\r" );

    timerTrace = xTimerCreate("TimerTrace",          // Timer name
                              pdMS_TO_TICKS(500),   // Timer period
                              pdTRUE,                // auto-reload
                              ( void * ) 0,          // Timer ID used as counter (initialized
to 0)
                              traceCount );          // Call back function

    timerLed = xTimerCreate( "TimerLed",             // Timer name
                              pdMS_TO_TICKS(200),   // Timer period
                              pdTRUE,                // auto-reload
                              ( void * ) 0,          // Timer ID used as counter (initialized
to 0)
                              toggleLed );            // Call back function

    xTimerStart( timerTrace, 0 );
    xTimerStart( timerLed, 0 );
}

void loop()
{
}
```

2 – Que fait le programme suivant :

```
#define LED_EXT_PIN      4    // External red LED

// Array of characters
#define DATA_LENGTH    32
char data[ DATA_LENGTH ];

// Handles
QueueHandle_t queue;

// Data structure for the queue message
struct message
{
    char count;
    char *pData;
};

void taskDataGen( void *pvParameters )
{
    char cnt = 0;
    struct message dataToSend;

    while( 1 )
    {
        vTaskDelay( 10 );
        dataToSend.count = cnt;
        dataToSend.pData = &data[ cnt++ % DATA_LENGTH ];
        xQueueSend( queue, ( void * )&dataToSend, ( TickType_t ) 100 );

        dataToSend.count = cnt;
        dataToSend.pData = &data[ cnt++ % DATA_LENGTH ];
        xQueueSend( queue, ( void * )&dataToSend, ( TickType_t ) 100 );

        dataToSend.count = cnt;
        dataToSend.pData = &data[ cnt++ % DATA_LENGTH ];
        xQueueSend( queue, ( void * )&dataToSend, ( TickType_t ) 100 );
    }
}

void taskDataConsum( void *pvParameters )
{
    int ledState = 1;
    char message = 0xFF;

    struct message dataReceived;

    while( 1 )
    {
        xQueueReceive( queue, &dataReceived, 1000 );
        printf( "Data received [ Count: %03d ] [Data: %c ]\n\r", dataReceived.count,
        *( char* )dataReceived.pData );
        for ( int x = 0; x < 2; x++ )
        {
            ledState = ledState ^ 1;
            digitalWrite( LED_EXT_PIN, ledState );
            vTaskDelay(2);
        }
    }
}

void setup( )
{
    pinMode(LED_EXT_PIN, OUTPUT);
    Serial.begin(115200);
    sleep( 2 );
    Serial.printf( "[ Queues ] \n\n\r" );

    for ( int x = 0; x < DATA_LENGTH; x++ )
    {
        data[ x ] = 'A' + x;
    }

    queue = xQueueCreate(10, sizeof( struct message ) );
    xTaskCreate( taskDataConsum, "DataConsumer", 4096, NULL, 1, NULL );
    xTaskCreate( taskDataGen, "DataGenerator", 4096, NULL, 1, NULL );
}

void loop()
{
}
```