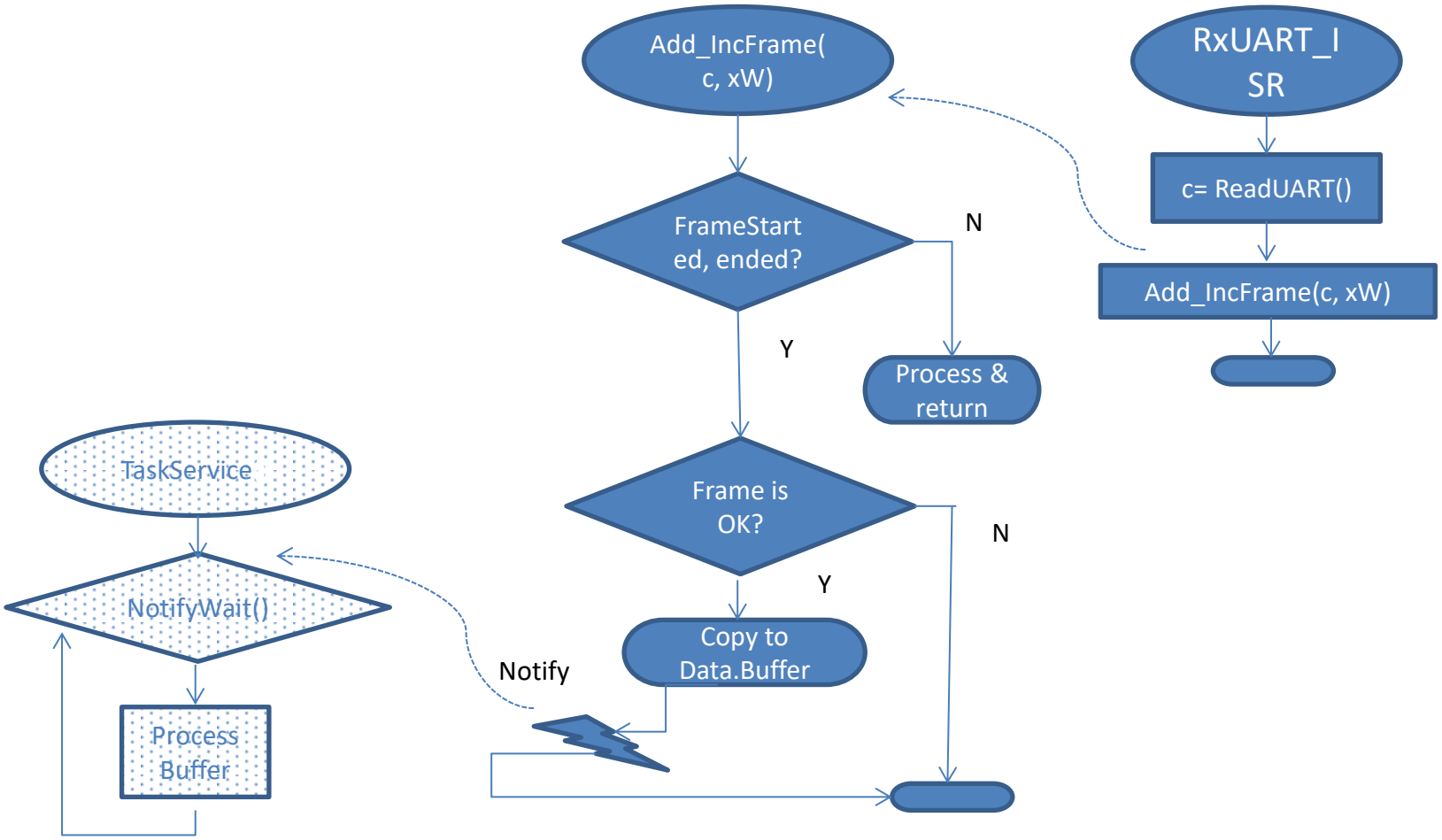
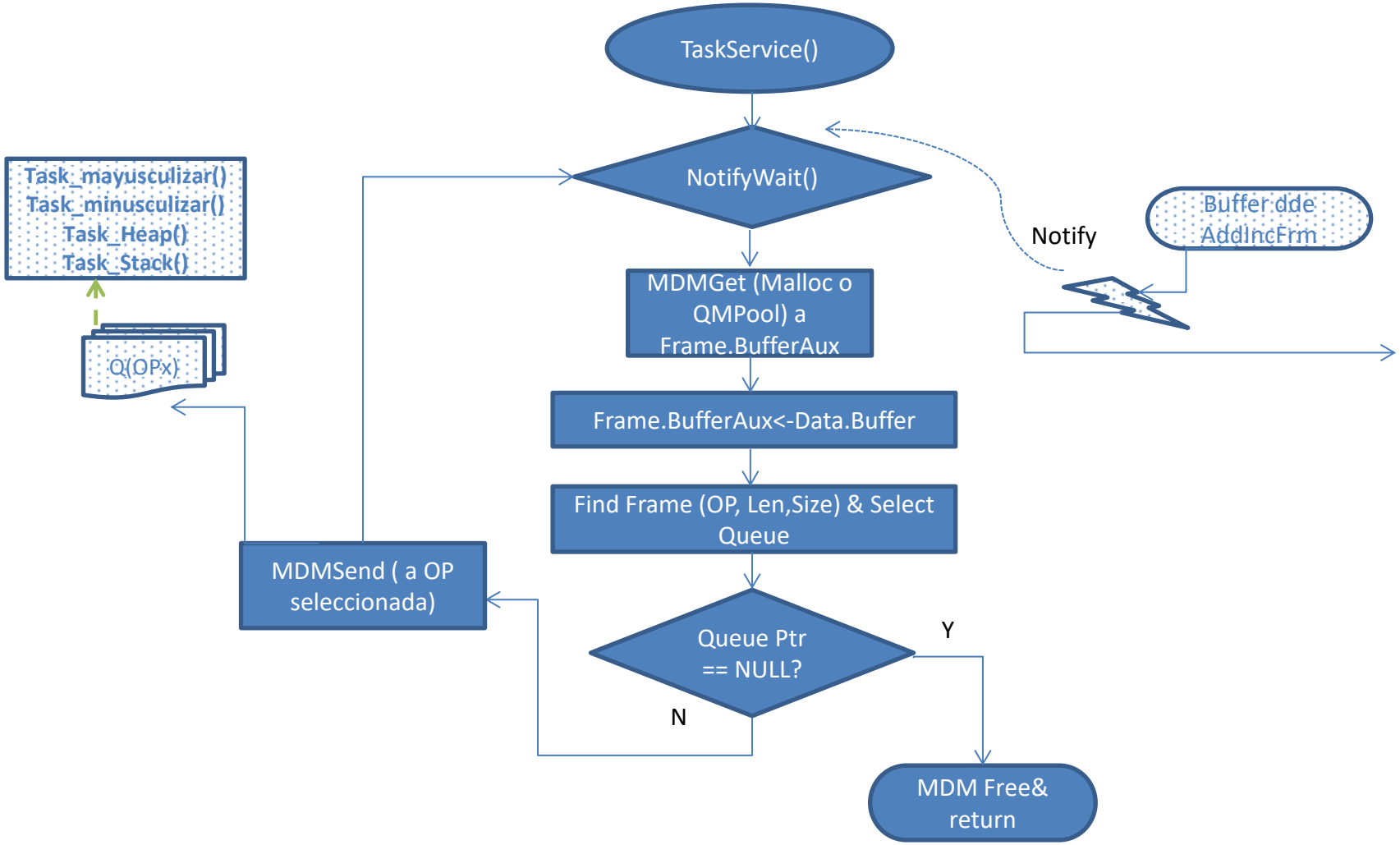
Estructura actual TP1 / Grupo 1 RTOS2 – JB&todos

1.a) Ingreso caracteres dde UART v4.6.19



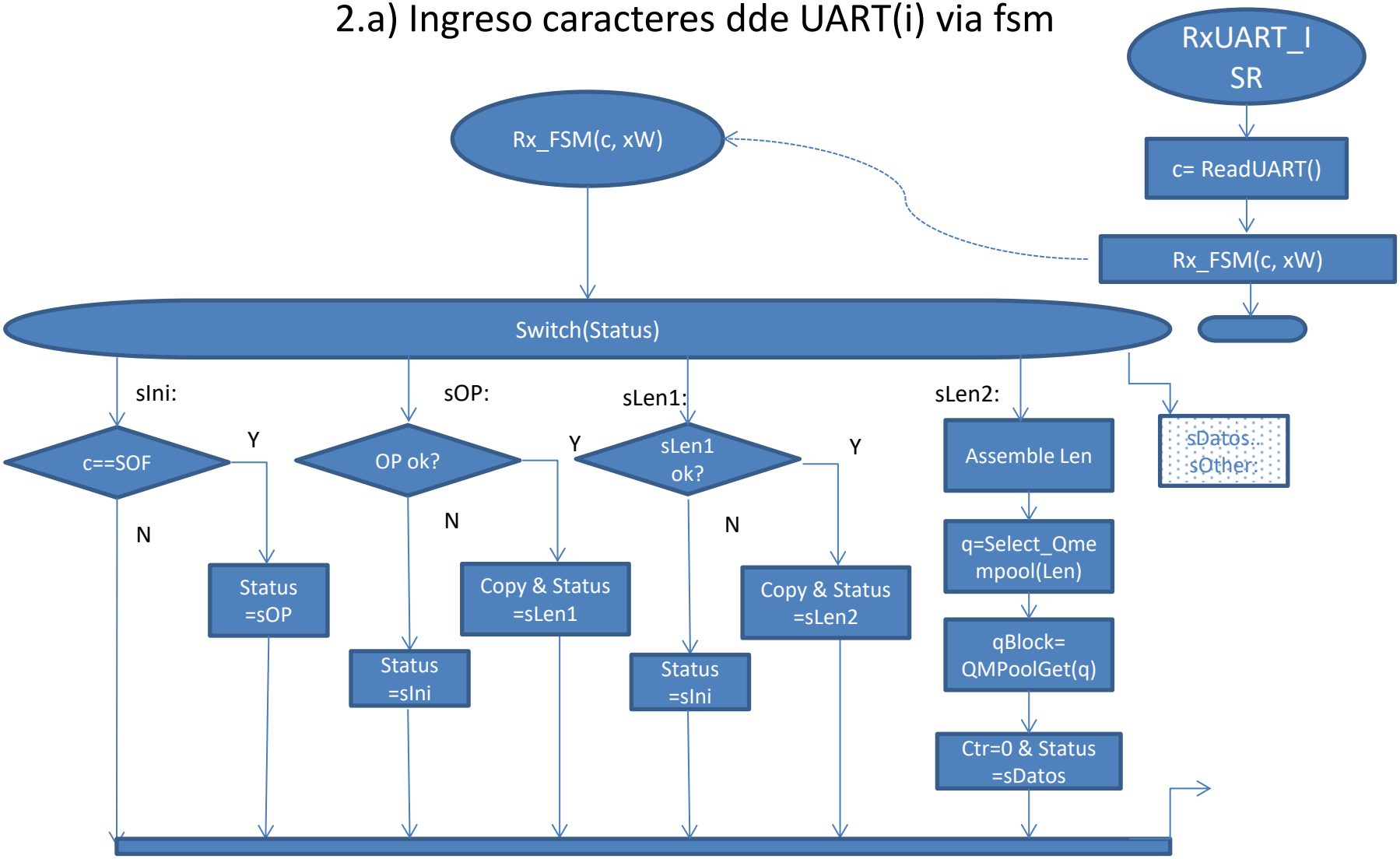
Estructura actual TP1 / Grupo 1 RTOS2 – JB&todos

1.b) Proceso en TaskService v4.6.19



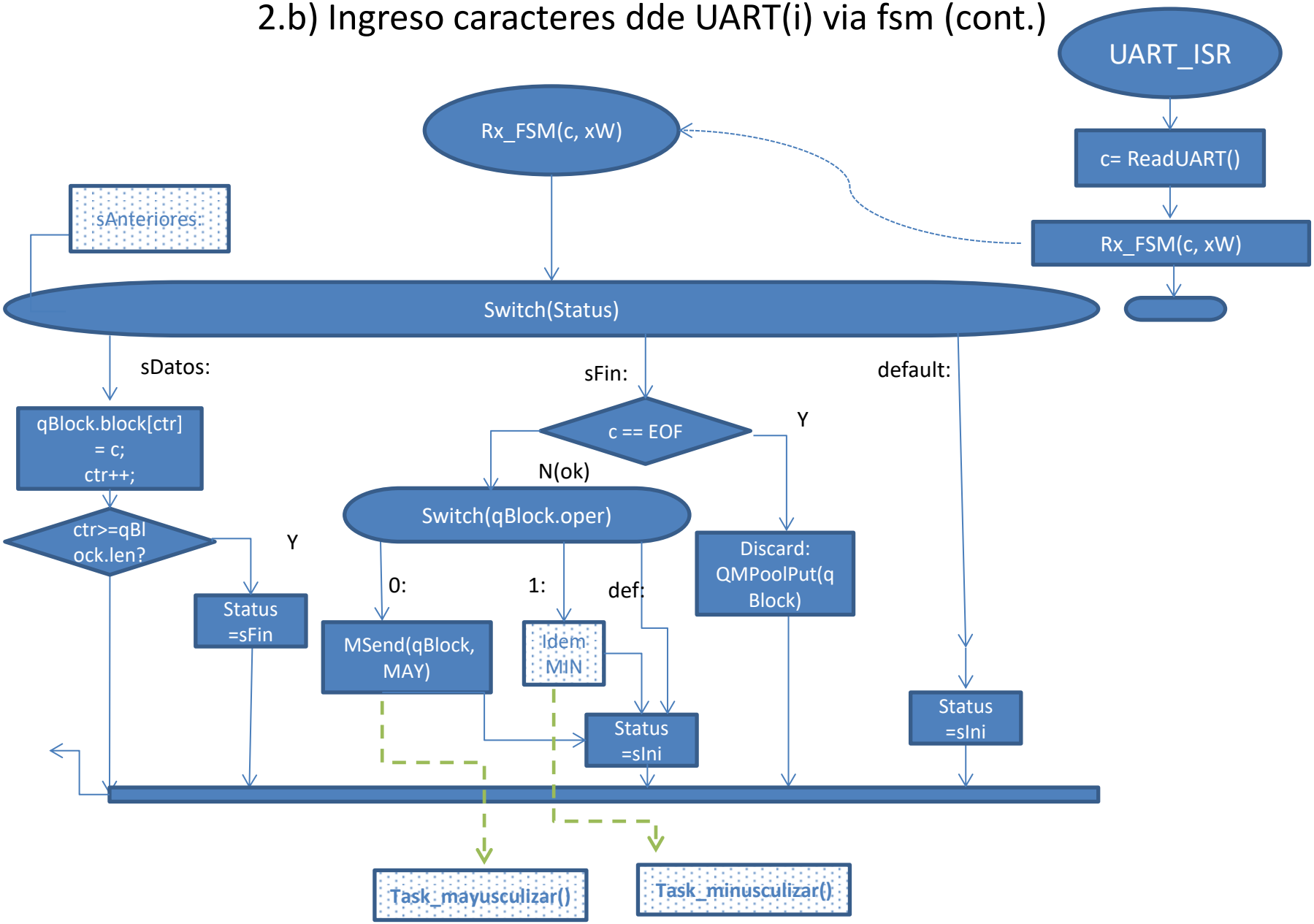
Estructura Propuesta TP1 / Grupo 1 RTOS2 – Celery

2.a) Ingreso caracteres dde UART(i) via fsm



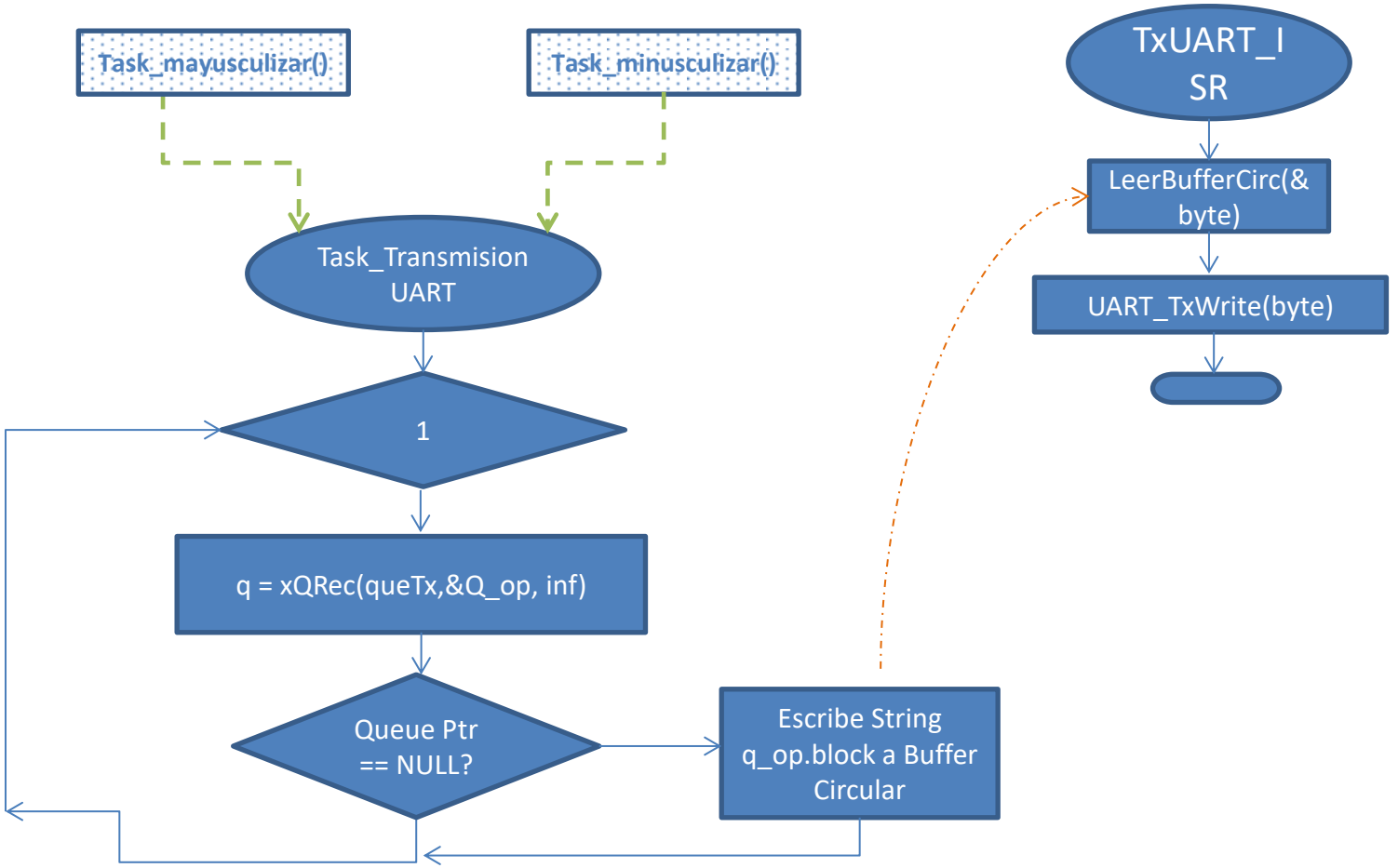
Estructura Propuesta TP1 / Grupo 1 RTOS2 – Celery

2.b) Ingreso caracteres dde UART(i) via fsm (cont.)



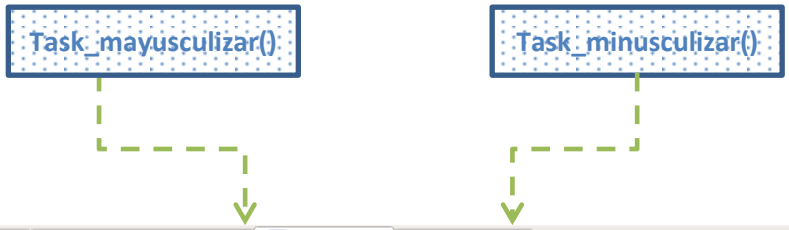
Estructura Propuesta TP1 / Grupo 1 RTOS2 – Celery

2.C) Salida de Tasks Mayusculizar, Minusc. Via Task_Txmit por interrupción TXEmpty (asincrónico, no bloqueante)



Estructura Actual TP1 / Grupo 1 RTOS2 – JBytodos

2.d) Salida de Tasks Mayusculizar, Minusc. Via Task_Txmit bloqueante



task_operaciones.c | task_operaciones.c | Task1.c | Task1.h

```
113
114 /*=====================================================================================
115                                     | Tarea tx |
116  ====================================================================================*/
117 void TaskTxUart( void* taskParmPtr ){
118     char * BSend;
119     char Txbuffer[100];
120     while(true){
121
122         /*Recibe por la cola*/
123         BSend = ModuleDinamicMemory_receive(&ModuleData, xPointerQueue_3, portMAX_DELAY);
124         gpioToggle( LED3 );
125         if( uartTxReady( UART_USB ) ){
126             sprintf( Txbuffer, "%s",BSend);
127             //Transmit_UART( 0 );   // La primera vez – con esto arranca
128             uartWriteString(UART_USB,Txbuffer);
129         }
130         //ModuleDinamicMemory_Free(&ModuleData, BSend);
131         ModuleData.MemoryFreeFunction(&mem_pool_big,BSend,0,NULL,QmPoolOrMalloc);
132     }
133 }
134
```

Estructura de Datos original TP1 / Gr1 RTOS2 – Celery

3.a) Sin medición, uso QMPool

task_operaciones.c | task_operaciones.c

```
36  QMPool mem_pool_chico;
37
38  // Colas de operaciones
39  QueueHandle_t queMayusculizar;
40  QueueHandle_t queMinusculizar;
41  QueueHandle_t queTransmision;
42
43  #define tamanio_queue_paquetes_pendientes   10
44
45  enum protocolo                  { eProtocoloInicio_STX = 0x55, eProtocoloFi
46  enum paquete_estados_e          { eInicio, eOperacion, eLongDatos, eDatos,
47  enum paquete_operaciones_e      { oMayusculizar, oMinusculizar, oStackDispo
48
49  typedef struct {
50      uint8_t operacion;
51      uint8_t longDatos;
52      QMPool * mem_pool;
53      char *  block;
54  } queue_operaciones_t;
```

(a)

(b)

(c)

TP1

Estructura de datos TP1 / Grupo 1 rtos2 – JB&todos

3.b) En revisión v4.6.19

task_operaciones.c | task_operaciones.c | Task1.c | Task1.h

```
35 /** ======= Datos para llenar buffer local=====================================*/
36 typedef struct {
37     char Buffer[106];
38     uint8_t Ready;
39     uint8_t Index;
40     uint8_t StartFrame;
41 }DataFrame_t;
42 extern volatile DataFrame_t Data;
43
44 /** =======Operaciones a usar=================================================*/
45 typedef enum{
46     OP0 = 0,    /* Convertir los caracteres recibidos a mayúsculas. (CMD/RTA)*/
47     OP1,        /*Convertir los caracteres recibidos a minúsculas. (CMD/RTA)*/
48     OP2,        /*Reportar stack disponible (RTA)*/
49     OP3         /*Reportar heap disponible. (RTA)*/
50 }Enum_Op_t;
51
52 /** =======Parametros de la trama de  llegada ================================*/
53 typedef struct {
54     char _SOF;
55     Enum_Op_t Operation;
56     uint8_t T[2];
57     char* Datos;
58     char* BufferAux;
59     char _EOF;
60 }Frame_parameters_t;
61
62 extern volatile Frame_parameters_t Frame_parameters;
```

TP1 – trama usada por ISR

Similar a : (a)

Similar a : (b)

Similar a : (c)

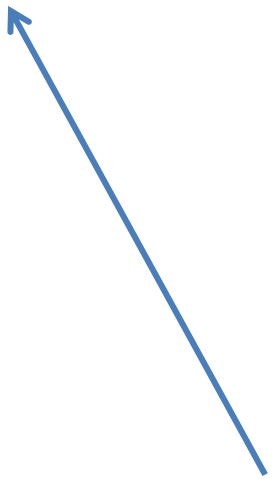
TP1 – trama del frame q llega de ISR

TP2 / Grupo 1 RTOS2 – Celery

4.a) Se insertan comandos de medición de performance y función para ACT

Comando ‘4’ - > medir performance del sistema (CMD/RTA)

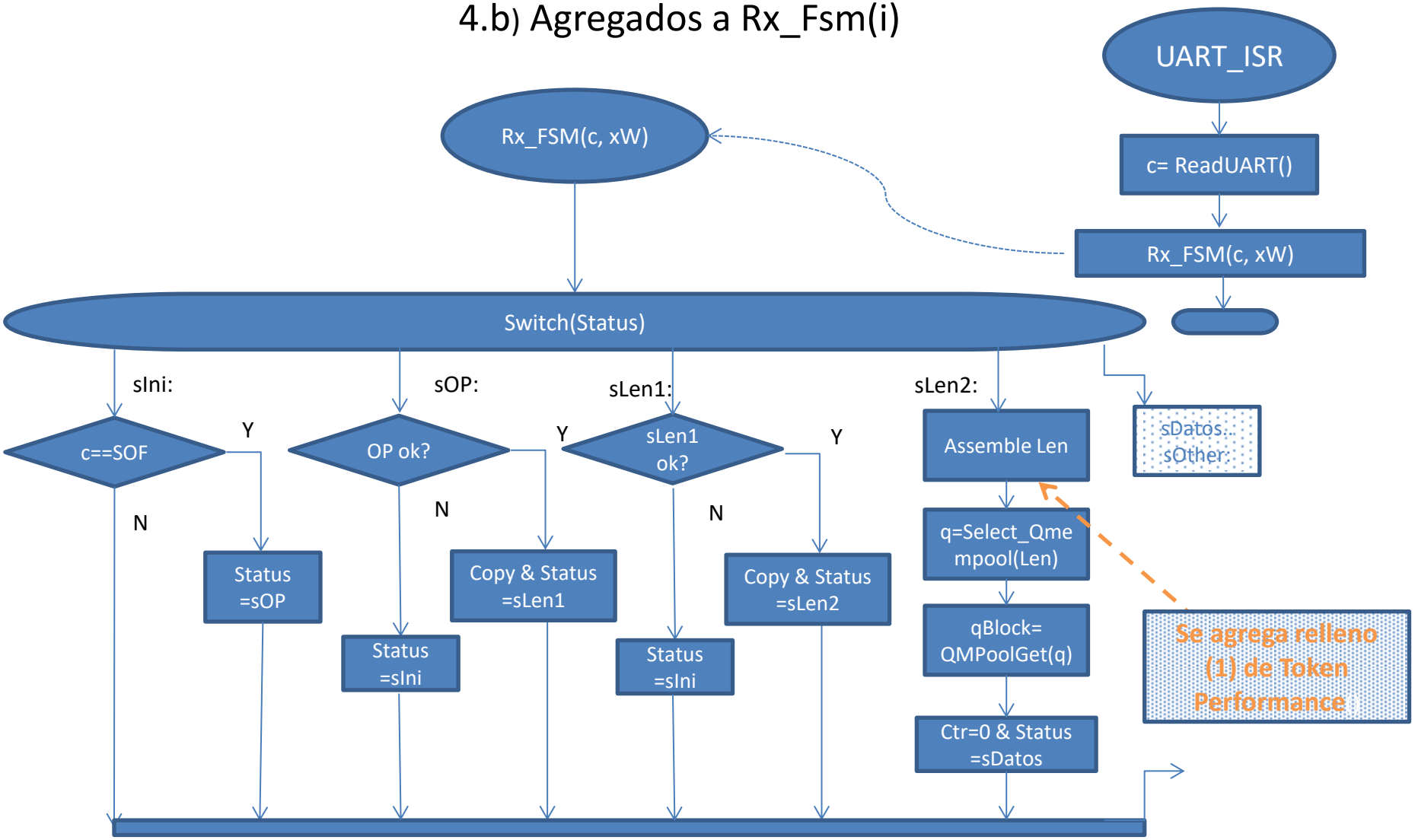
Comando ‘5’ -> Resultado de la medición de performance (RTA)



Agregados en TP2

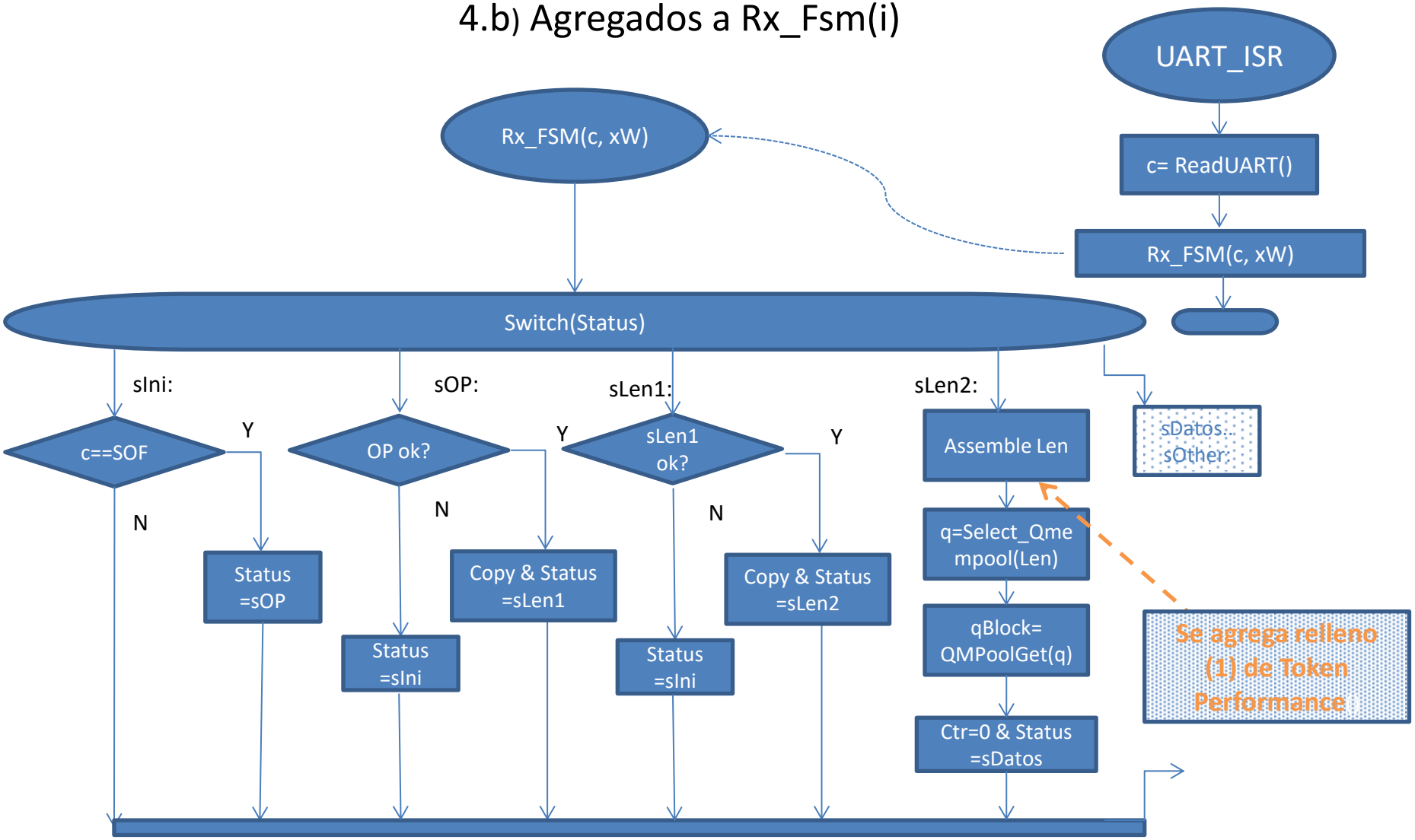
Estructura Propuesta TP2 / Grupo 1 RTOS2– Celery

4.b) Agregados a Rx_Fsm(i)



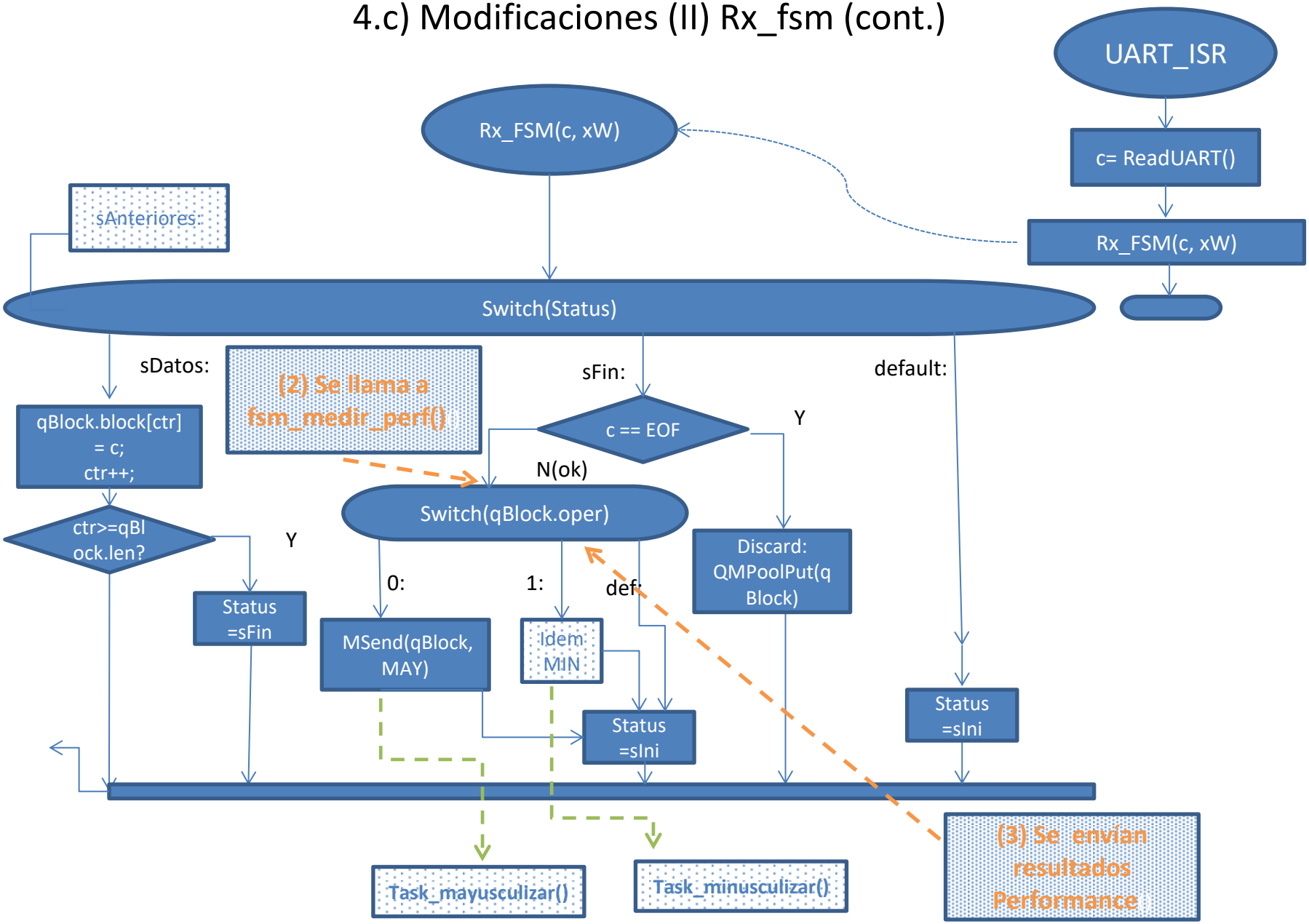
Estructura Propuesta TP2 / Grupo 1 RTOS2– Celery

4.b) Agregados a Rx_Fsm(i)



Estructura Propuesta TP2 / Grupo 1 RTOS2 – Celery

4.c) Modificaciones (II) Rx_fsm (cont.)



Estructura de Datos ampliada TP2 RTOS2 / Grupo 1 – Celery

3.b) Incluye Token con estructura de medicion y función para ACT

task_operaciones.c

```
56
57  // Medir performance
58  typedef enum estado_medicion    { eT_LL, eT_R, eT_I, eT_F, eT_S, eT_T } estado_mp;
59
60  static uint32_t  id_de_paquete = 0;
61
62  typedef struct {
63      estado_mp estado_Token;                          //Estado para la fsm_medir_performance();
64
65      uint32_t  id_de_paquete;                         //id_de_paquete=0 ---> id_de_paquete++
66      char * payload;                                  //apuntará al paquete de datos a procesar.
67      uint32_t  tiempo_de_llegada;                     //De la llegada del paquete.
68      uint32_t  tiempo_de_recepcion;                   //Del fin del paquete recibido.
69      uint32_t  tiempo_de_inicio;                      //Inicio de la operaciÃ³n (Mayusculizar).
70      uint32_t  tiempo_de_fin;                         //Fin de la operaciÃ³n (Mayusculizar):
71      uint32_t  tiempo_de_salida;                      //Tiempo de inicio de transmisiÃ³n [STX].
72      uint32_t  tiempo_de_transmision;                 //Tiempo de fin de transmisiÃ³n [ETX].
73      uint16_t  largo_del_paquete;                     //NÂº de bytes (Datos)+ Header=4 bytes.
74      uint16_t  memoria_alojada;                       //TamaÃ±o del pool.
75      void (*ptr_completion_handler)( void *T, BaseType_t * xHig );       //Puntero a funciÃ³n.
76  } Token_t;
77  ////////////////////
78
79  typedef struct {
80      uint8_t operacion;
81      uint8_t longDatos;
82      QMPool * mem_pool;
83      char *  block;
84      Token_t * Token;
85  } queue_operaciones_t;
```

Agregado en TP2