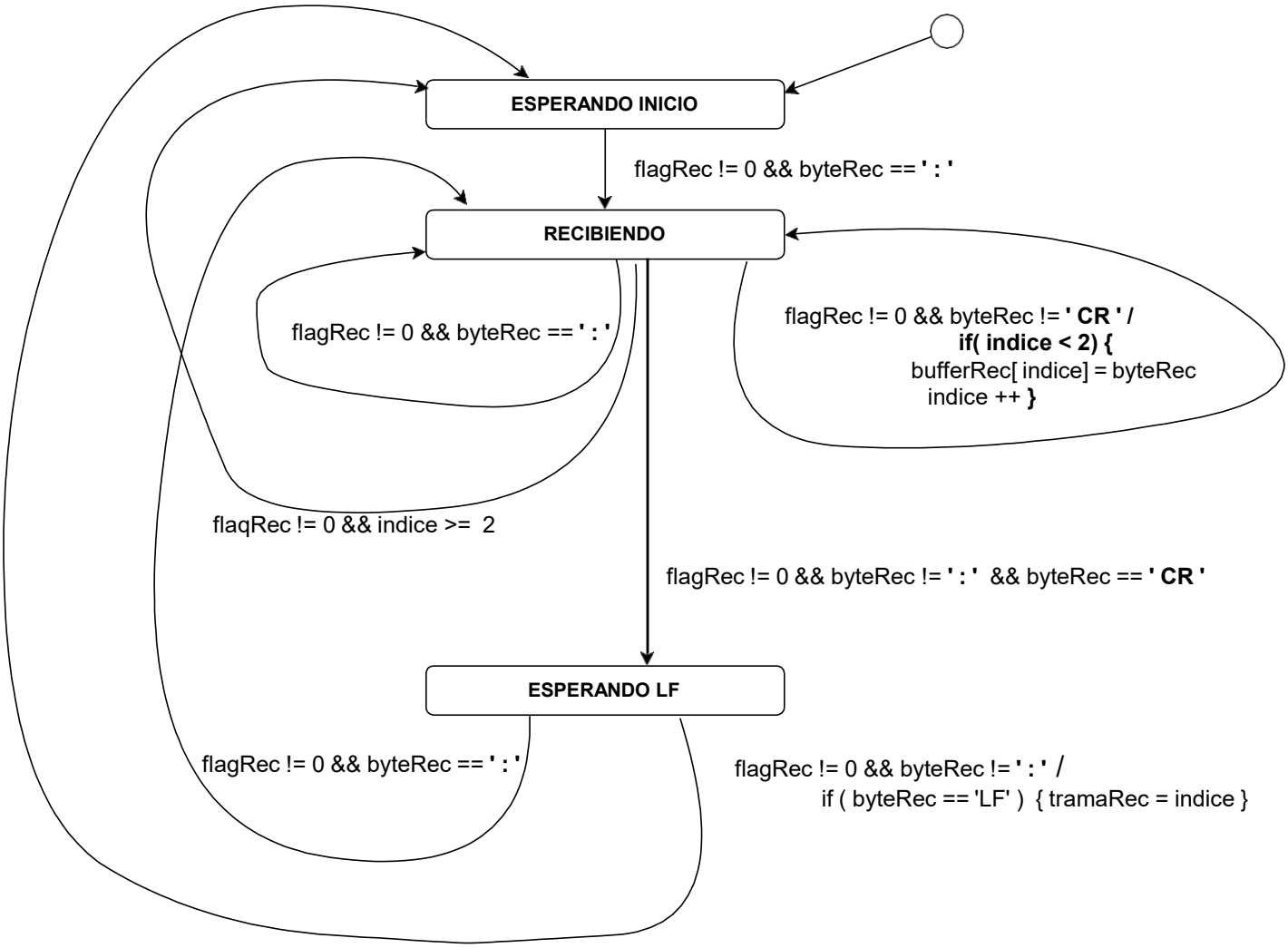
Practica 3 ejercicio 6

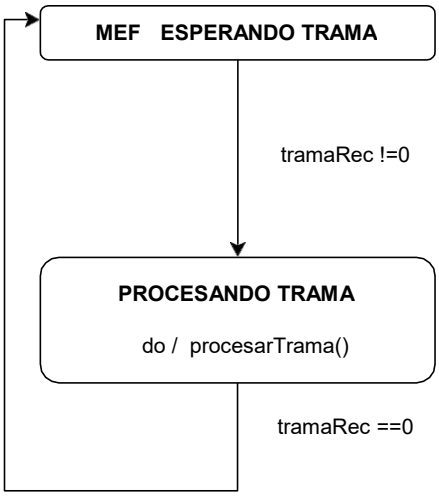
se ejecutará esta función al comienzo de la función MEF1

flagRec = uart_ringBuffer_recDatos (&byteRec, 1));

MEF1



MEF2



FUNCIÓN procesarTrama():

```
static void procesarTrama(void)  {
     switch (bufferRec[0])  {
          case 'A':
               accionLed(bufferRec[1],  BOARD_LED_MSG_OFF);
               break;
          case 'P':
               accionLed(bufferRec[1],  BOARD_LED_MSG_ON);
               break;
          case 'T':
               accionLed(bufferRec[1],  BOARD_LED_MSG_TOGGLE);
               break;
     }

     tramaRec=0;
}
```

FUNCIÓN accionLed():

```
static void accionLed(uint8_t charIdLed, board_ledMsg_enum ledMsg)
{
     switch (charIdLed)    {
          case '1':
               board_setLed(BOARD_LED_ID_ROJO,  ledMsg);
               break;

          case '2':
               board_setLed(BOARD_LED_ID_VERDE,  ledMsg);
               break;
     }
}
```

la solución de este problema está pensada utilizando la solución "uart.ringBuffer" brindada por la cátedra. Donde cada vez que se recibe un dato por UART el mismo es almacenado en una cola circular dentro de la rutina de interrupción

recordemos la función para extraer datos de una cola circular (ejecutada en la MEF cada vez que se inicia):

```
int32_t uart_ringBuffer_recDatos(uint8_t *pBuf, int32_t size){

    int32_t ret = 0;

    /* entra sección de código crítico */
    NVIC_DisableIRQ(UART0_IRQn);

    while (!ringBuffer_isEmpty(pRingBufferRx) && ret < size)   {

        uint8_t dato;

        ringBuffer_getData(pRingBufferRx, &dato);

        pBuf[ret] = dato;

        ret++;
     }
```

Se usarán las varaibles globales:

```
uint8_t bufferRec[2];

uint8_t tramaRec = 0;

uint8_t byteRec, flagRec;
```