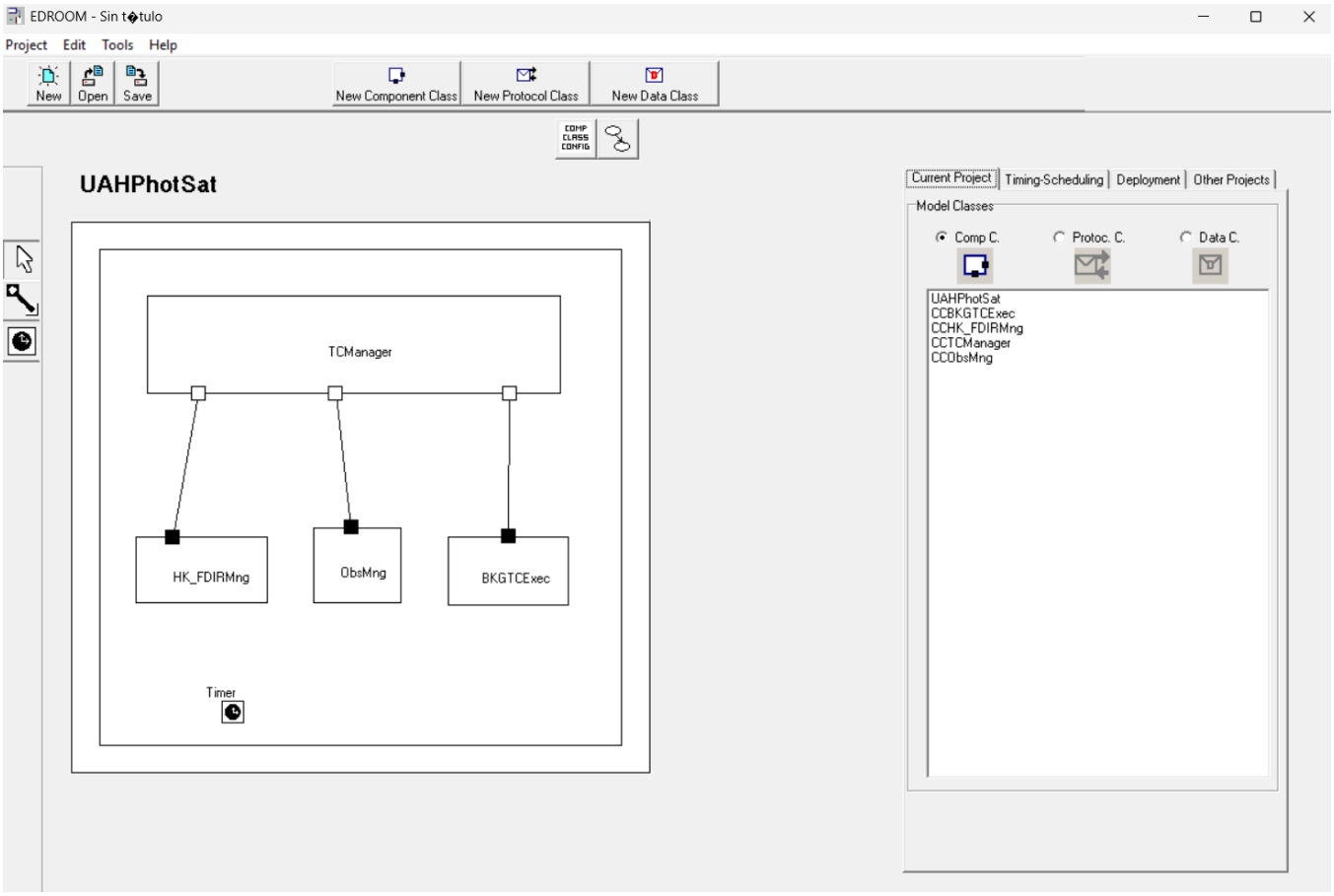
PROYECTO FINAL – IMPLEMENTACIÓN EN EDROOM

Elena Serrano

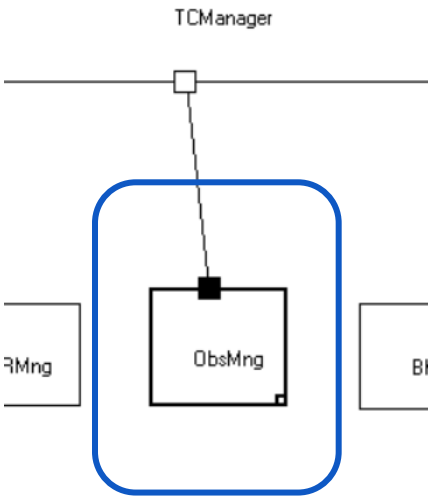
1. INTRODUCCIÓN

Este documento describe el proceso de implementación del componente ObsMng en EDROOM, siguiendo la arquitectura definida y su integración con el TcManager. Se detallan los accesos, prioridades, puertos, estados, transiciones, acciones y funciones utilizadas, apoyado con esquemas visuales.

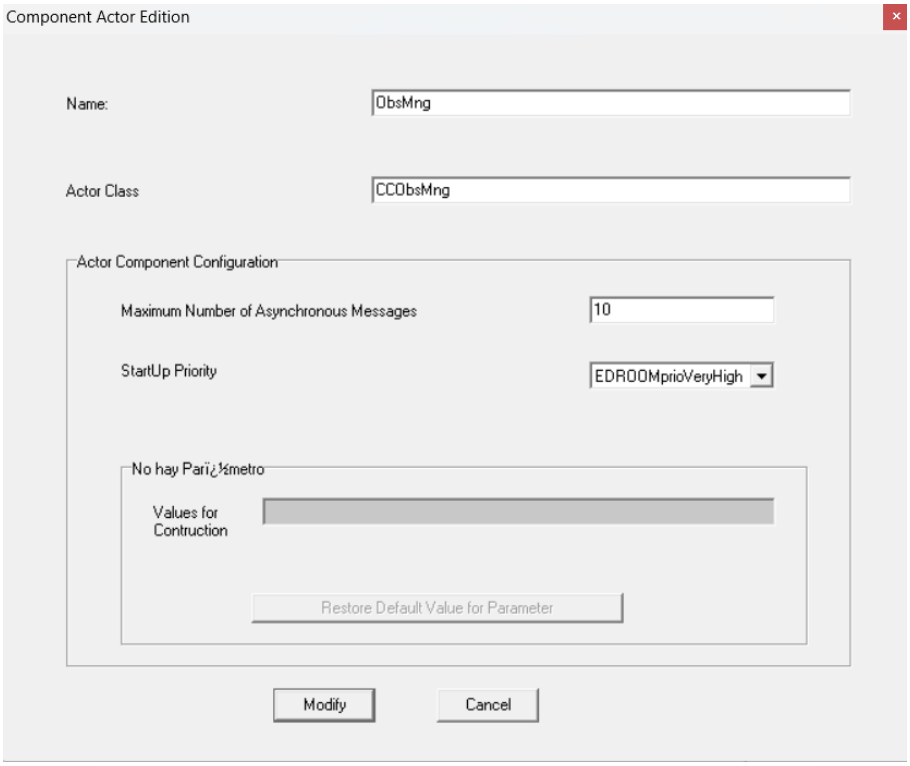


2. Configuración del Componente ObsMng

2.1 PRIORIDAD DE LA CLASE Y ACCESOS



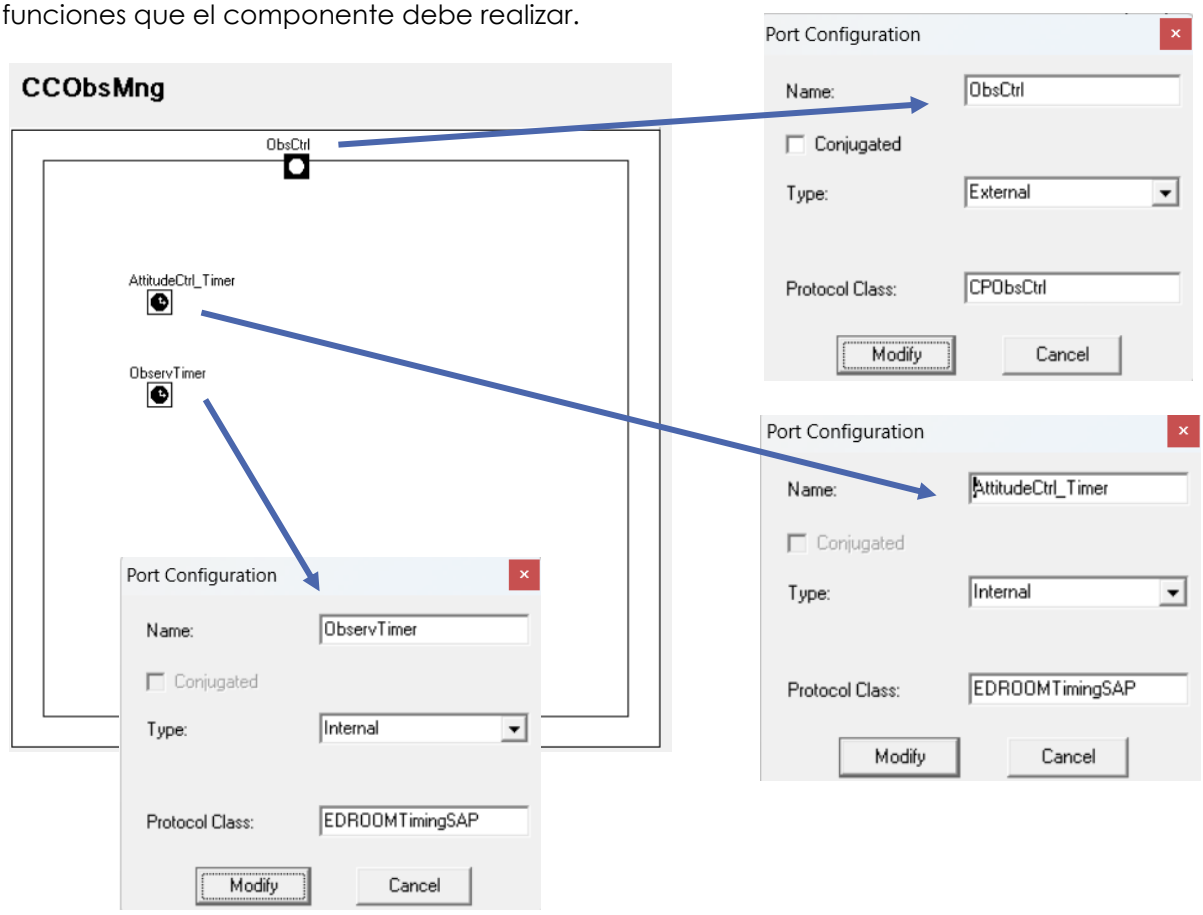
Se ha creado la clase componente ObsMng y se le ha asignado la prioridad más alta, para garantizar que sus acciones críticas se ejecuten con la máxima prioridad posible en el sistema. Además, se ha especificado a qué módulos tiene acceso, facilitando su integración dentro del modelo global.

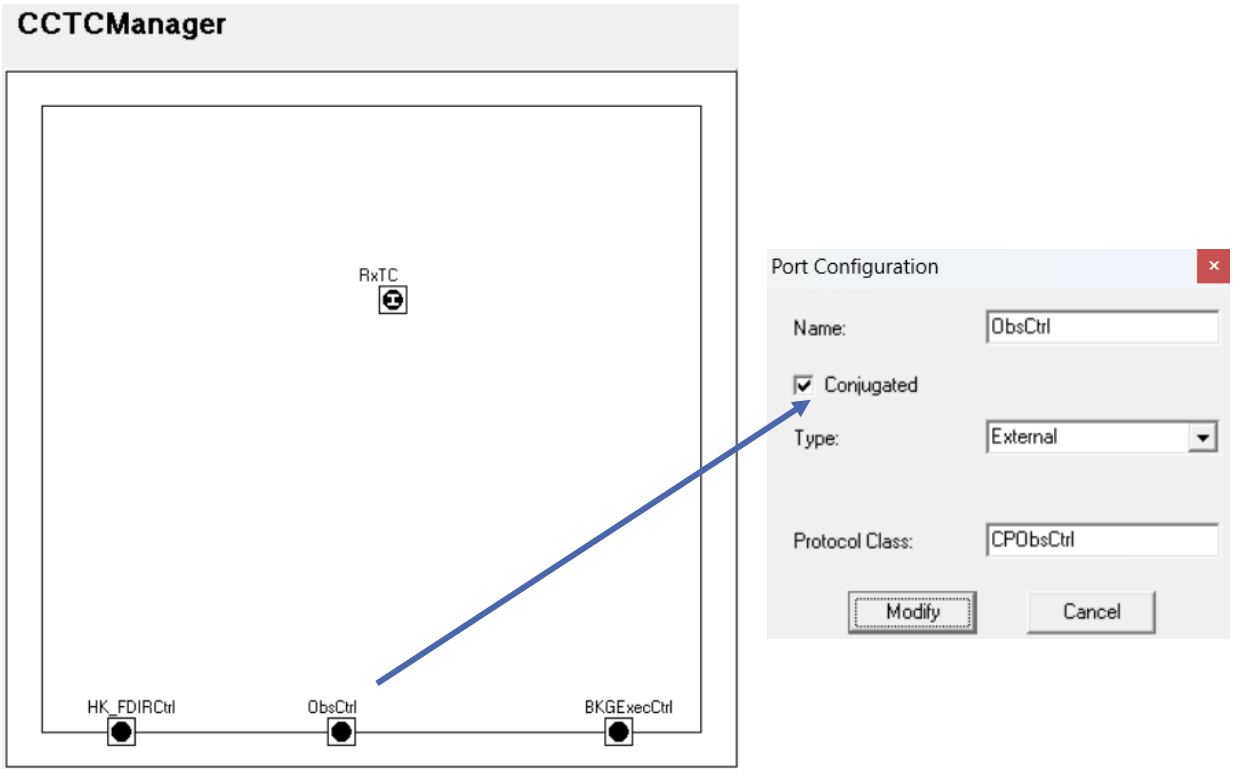


3. PREPARACIÓN PARA SEGUIR EL ESQUEMA DE OBSMNG

3.1 CREACIÓN DE PUERTOS Y ESTRUCTURA BÁSICA

Para que ObsMng pueda comunicarse correctamente, se han creado los puertos necesarios, además de su diagrama de flujo funcional. Esta estructura está alineada con el comportamiento esperado y las funciones que el componente debe realizar.





4. PASOS DE IMPLEMENTACIÓN

4.1 CREACIÓN DEL PUERTO EXTERNO

Se crea un puerto externo que actúa como interfaz entre ObsMng y los módulos con los que se comunica, siguiendo la especificación del sistema.

4.2 CREACIÓN DE TEMPORIZADORES

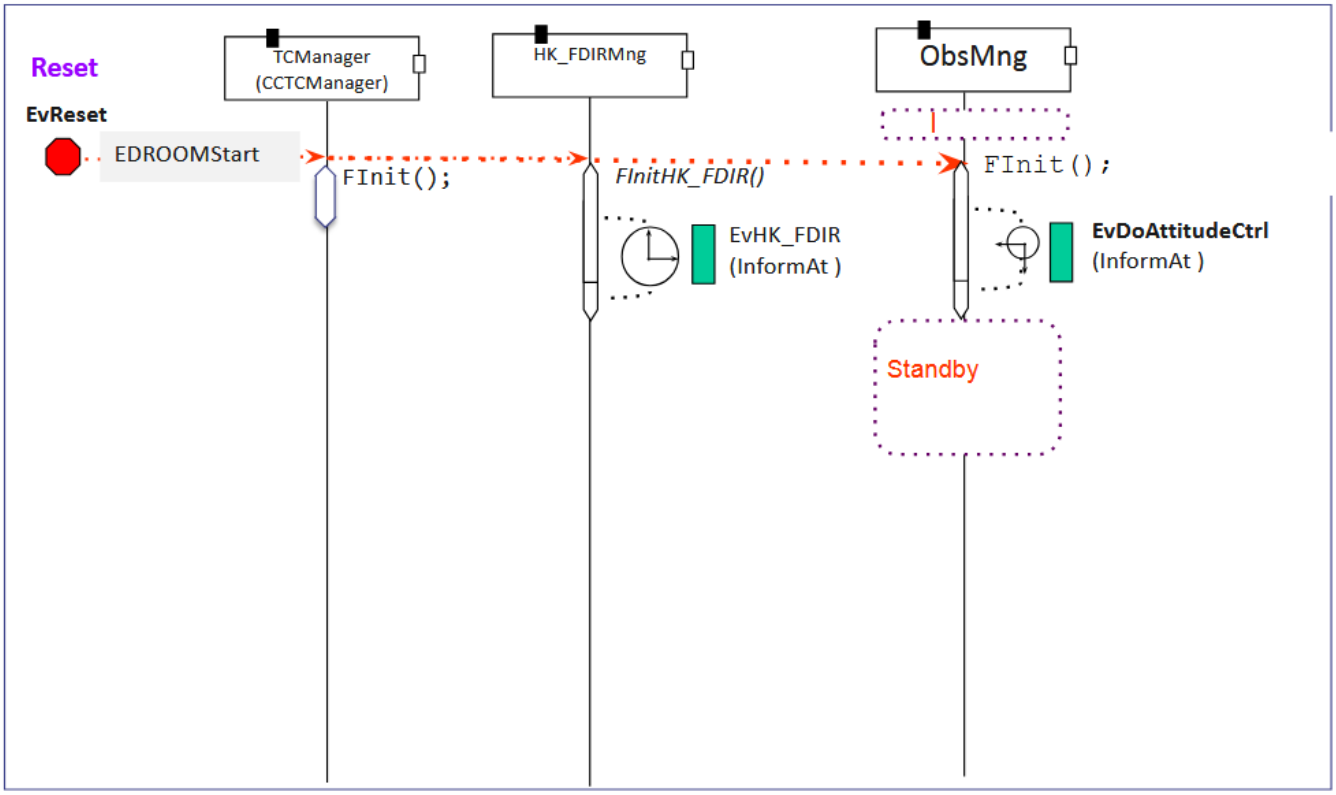
Se crean dos temporizadores necesarios para la ejecución de ciertas acciones de control, como la ejecución de una observación con retardo.

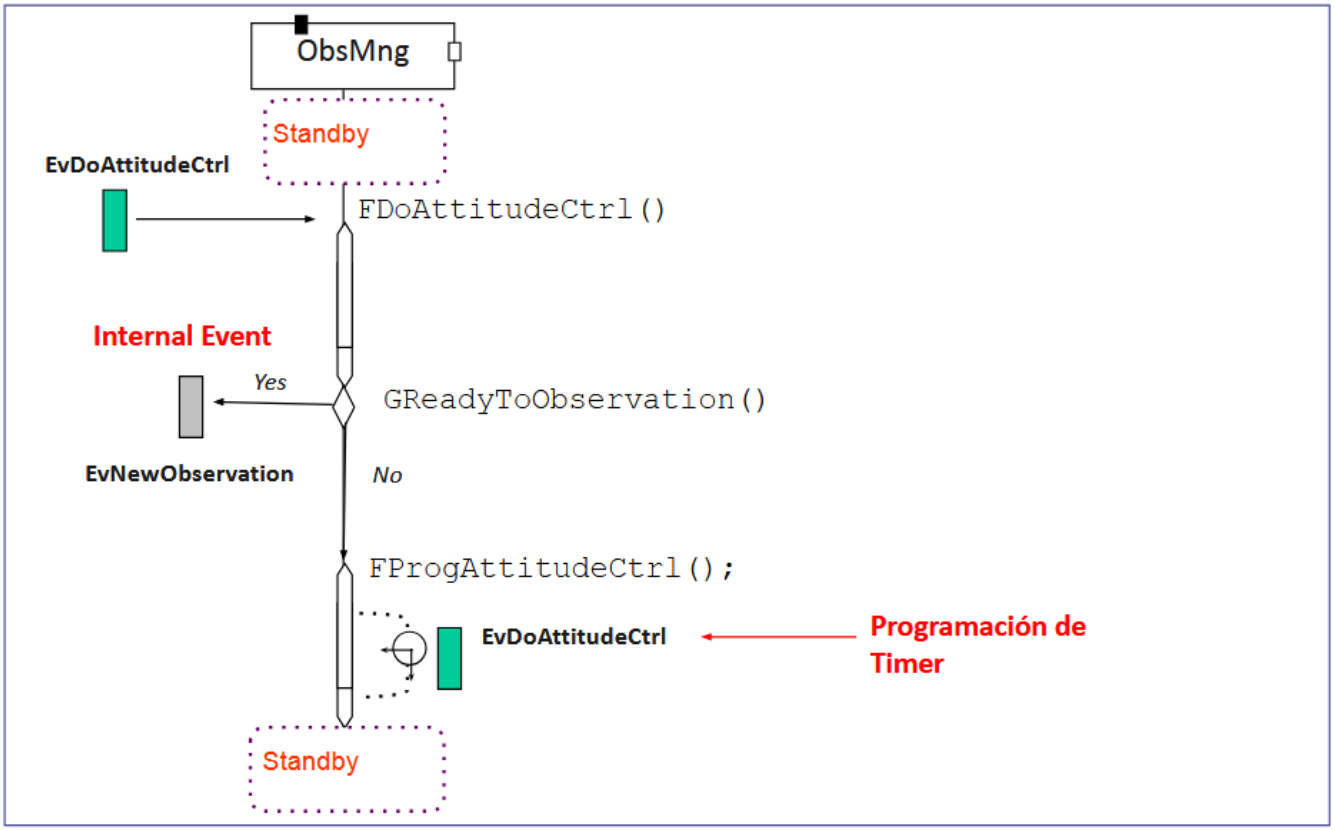
4.3 INTEGRACIÓN CON TCMANAGER

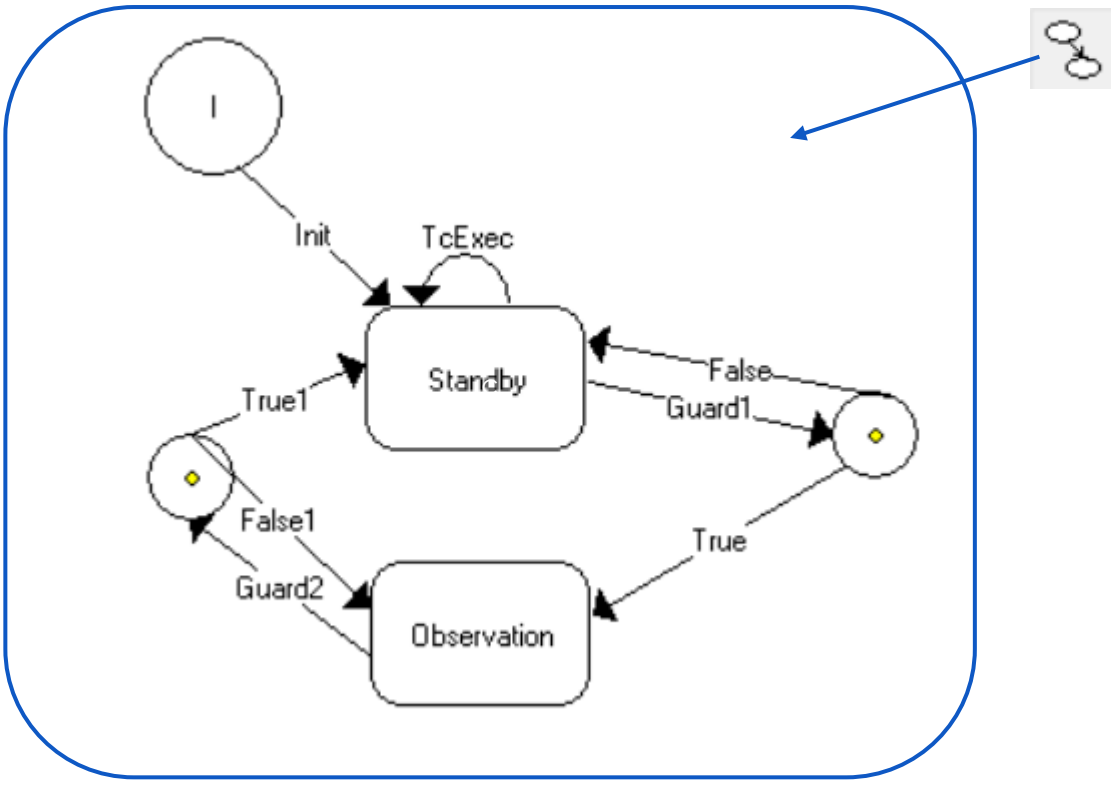
Se conjuga el puerto del TcManager con el componente ObsMng, permitiendo que este reciba eventos y comandos de telecontrol que desencadenan acciones en ObsMng.

5. DETALLE DEL DIAGRAMA DE FLUJO

Es importante tener presente el diagrama de flujo general para entender la lógica de transiciones y acciones implementadas.

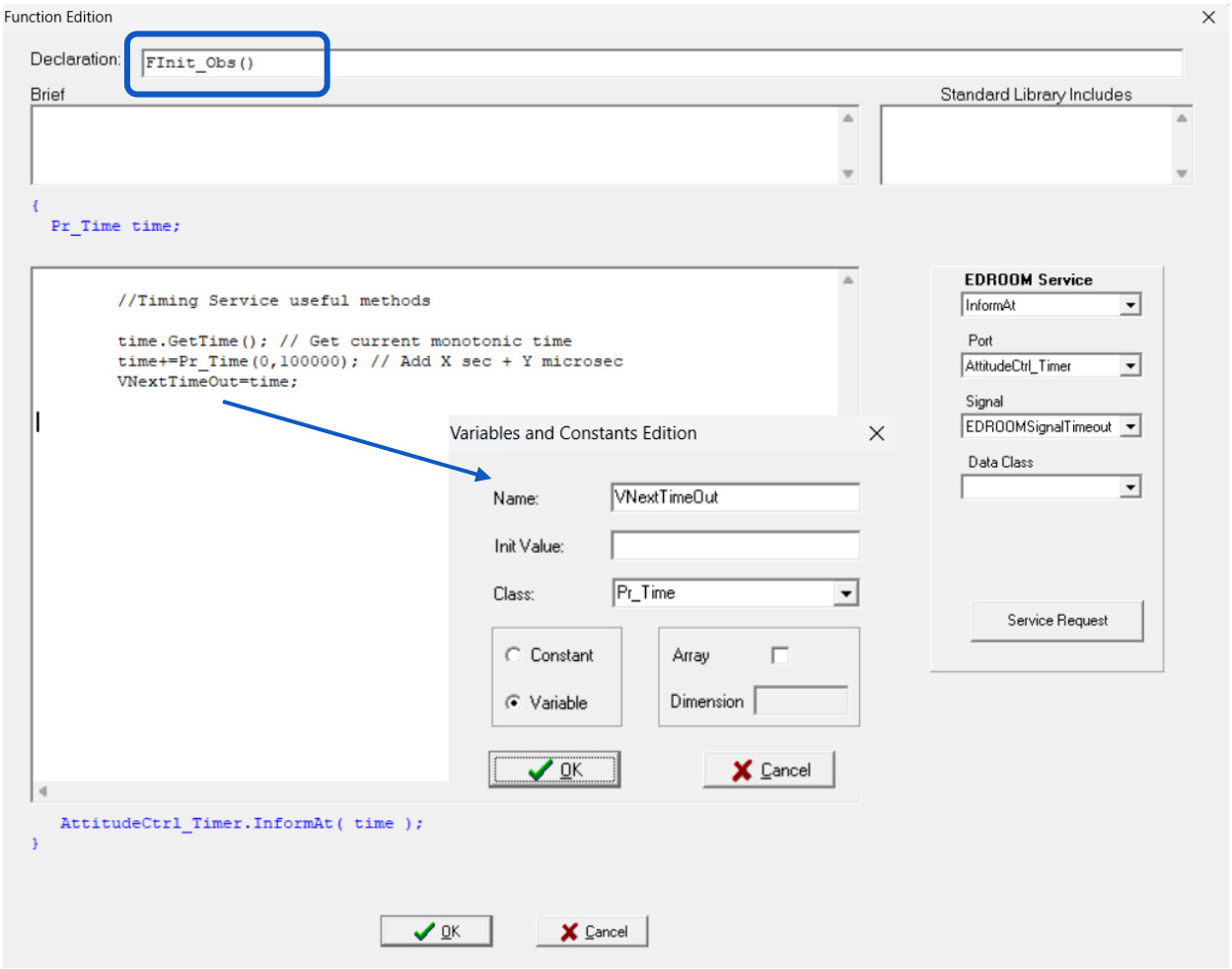




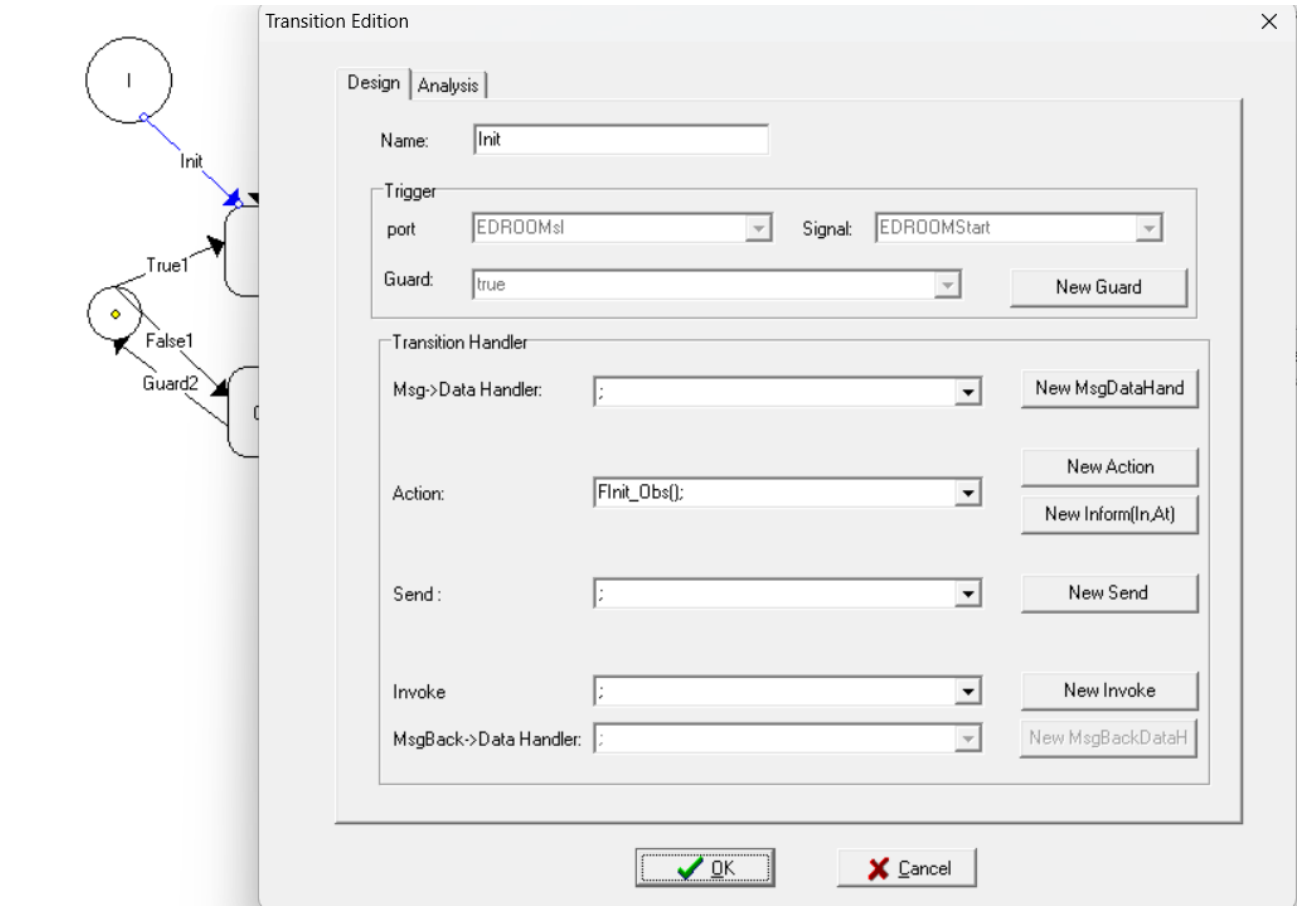


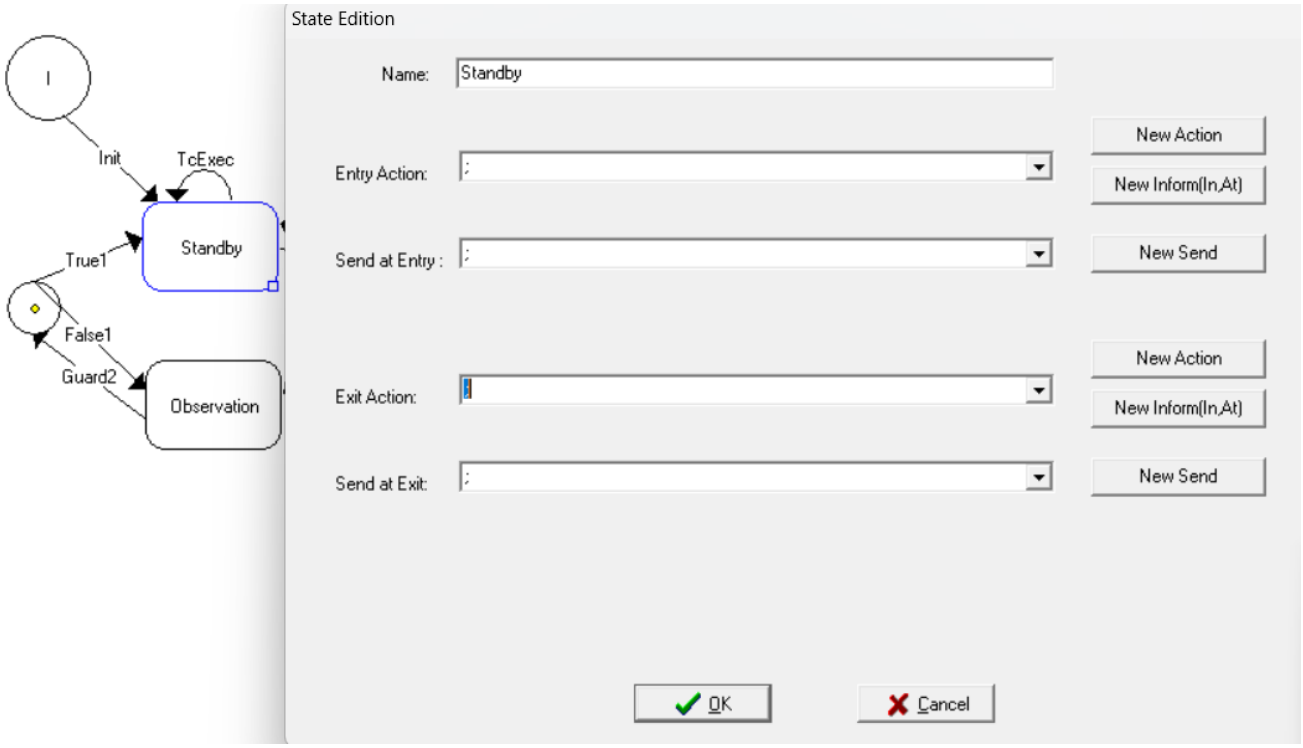
5.1 ESTADO INICIAL Y TRANSICIÓN A STANDBY

- Se define la función **FInit_Obs()** que inicializa el componente.



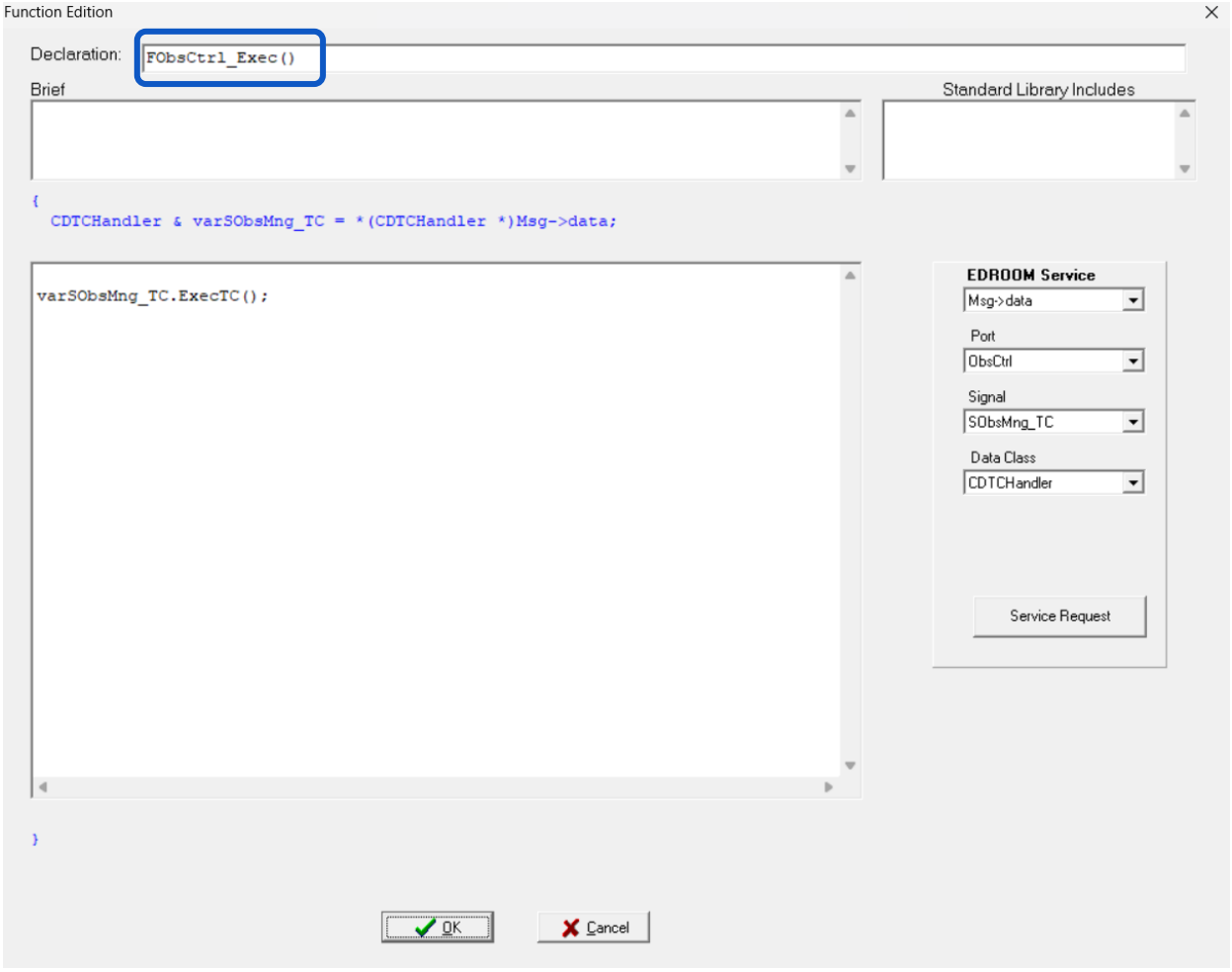
- Se crea un nuevo estado llamado **Standby**, que se alcanza mediante una transición desde el estado inicial **Init.**

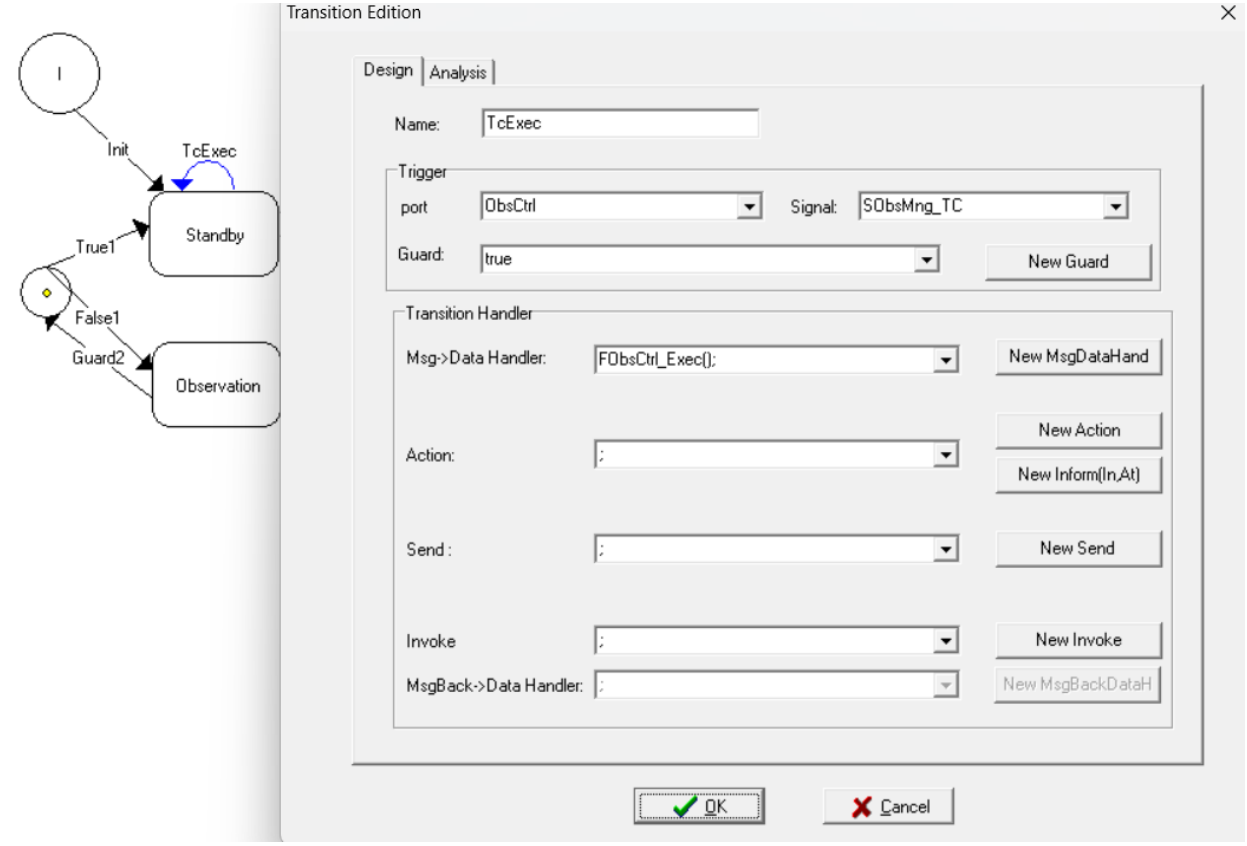




5.2 TRANSICIÓN TCEXEC

- Se crea la transición **TcExec**, asociada a la función **FObsCtrl_Exec()**, que gestiona la ejecución de un comando de observación.

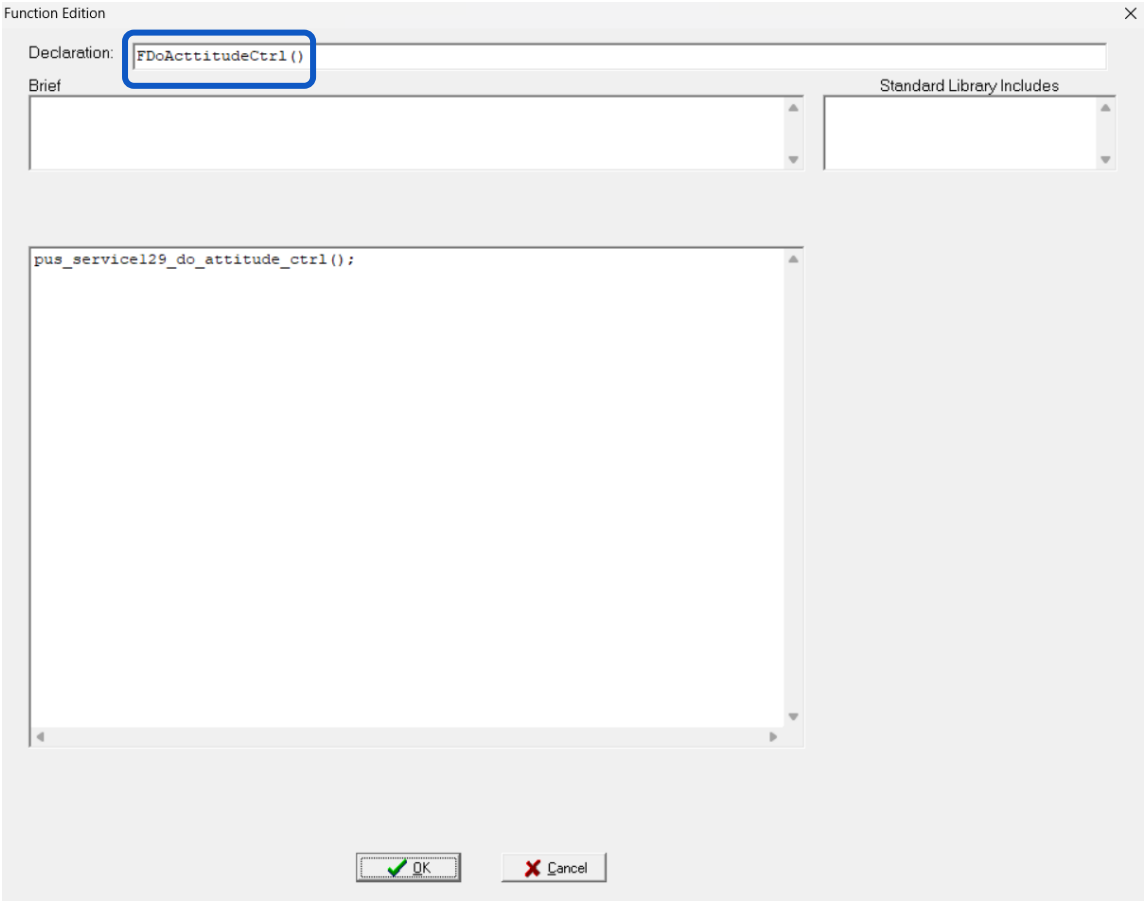


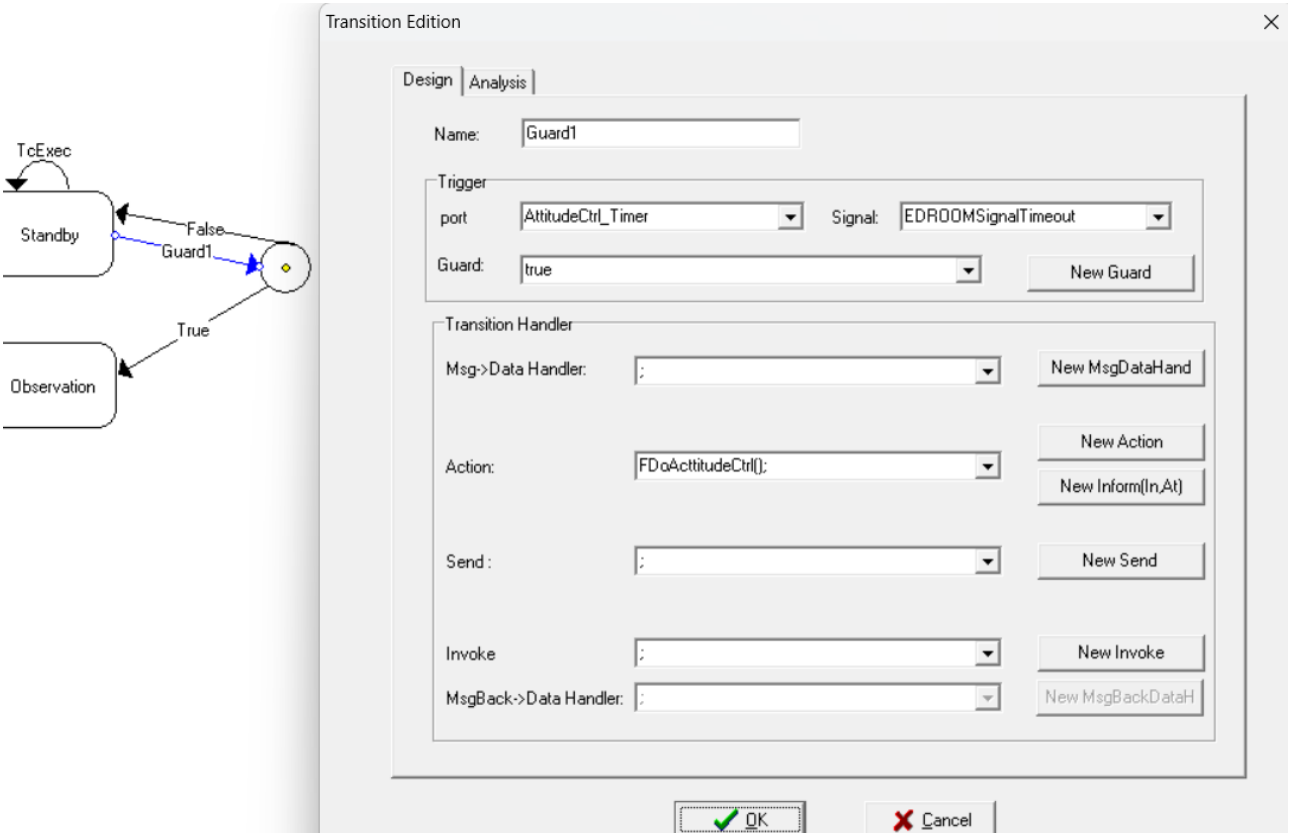


5.3 TRANSICIONES INICIALES TRAS TCEXEC

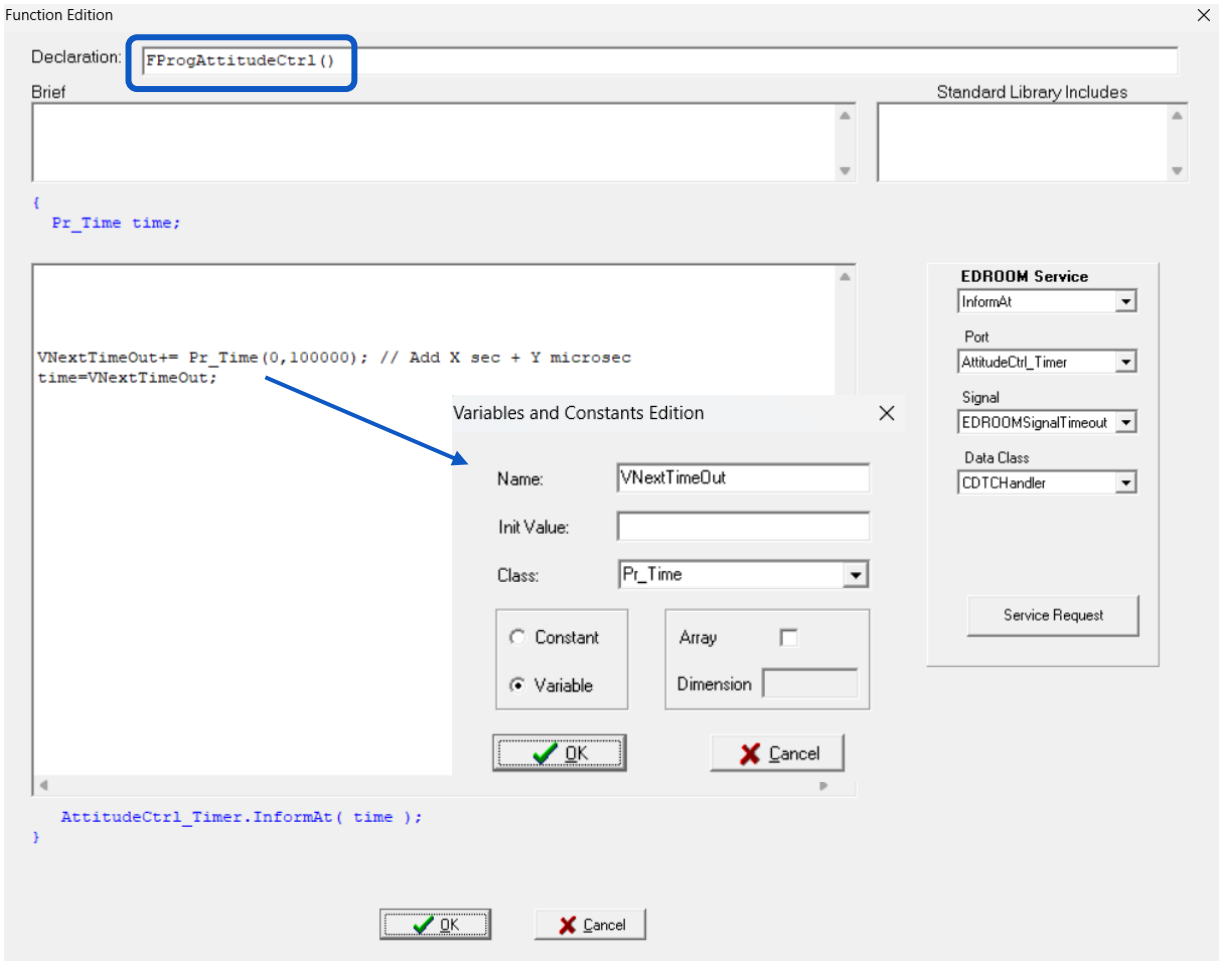
Antes de alcanzar el nodo de decisión **(ChoicePoint)**, se definen dos transiciones que preparan el sistema para la observación.

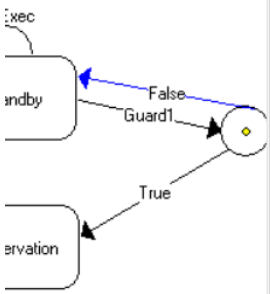
- **Transición Guard1:** Se activa al recibir la señal **EDROOMSignalTimeout** del temporizador **AttitudeCtrl_Timer**, ejecutando la acción **FDoAttitudeCtrl();** para iniciar el control de actitud.

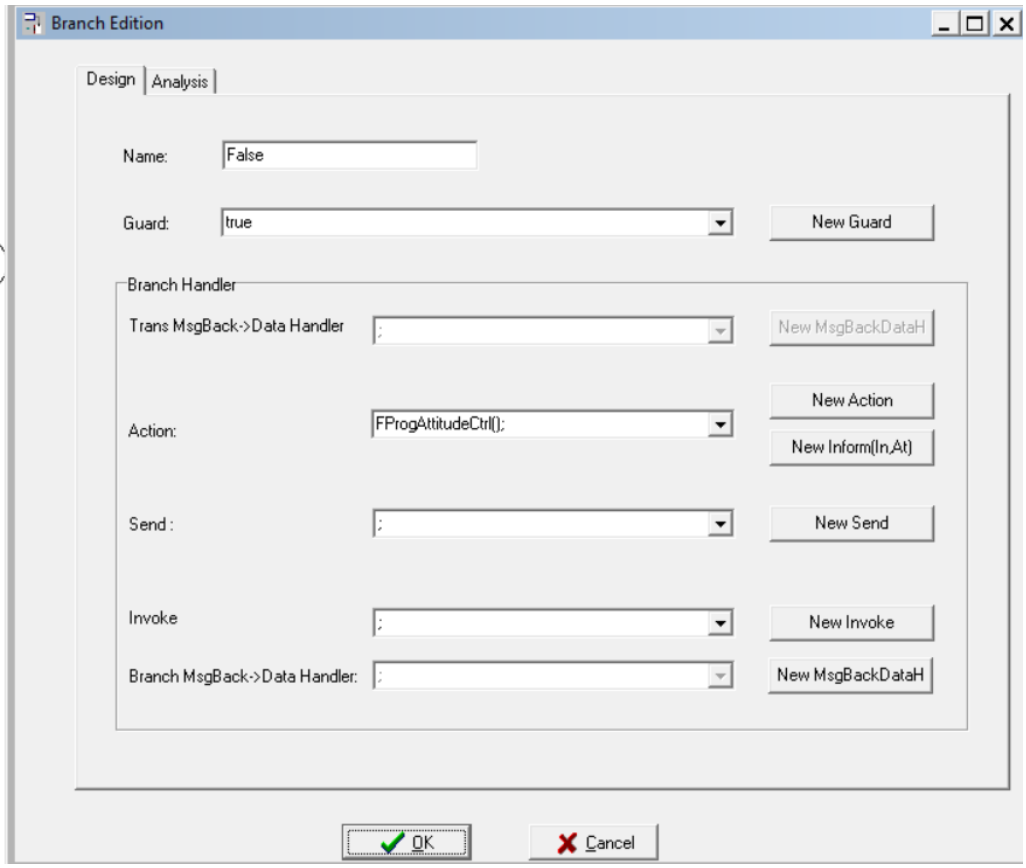




- **Rama False:** Representa una ruta alternativa con guarda true, que ejecuta la acción **FProgAttitudeCtrl();** para reprogramar el control de actitud si no procede continuar directamente



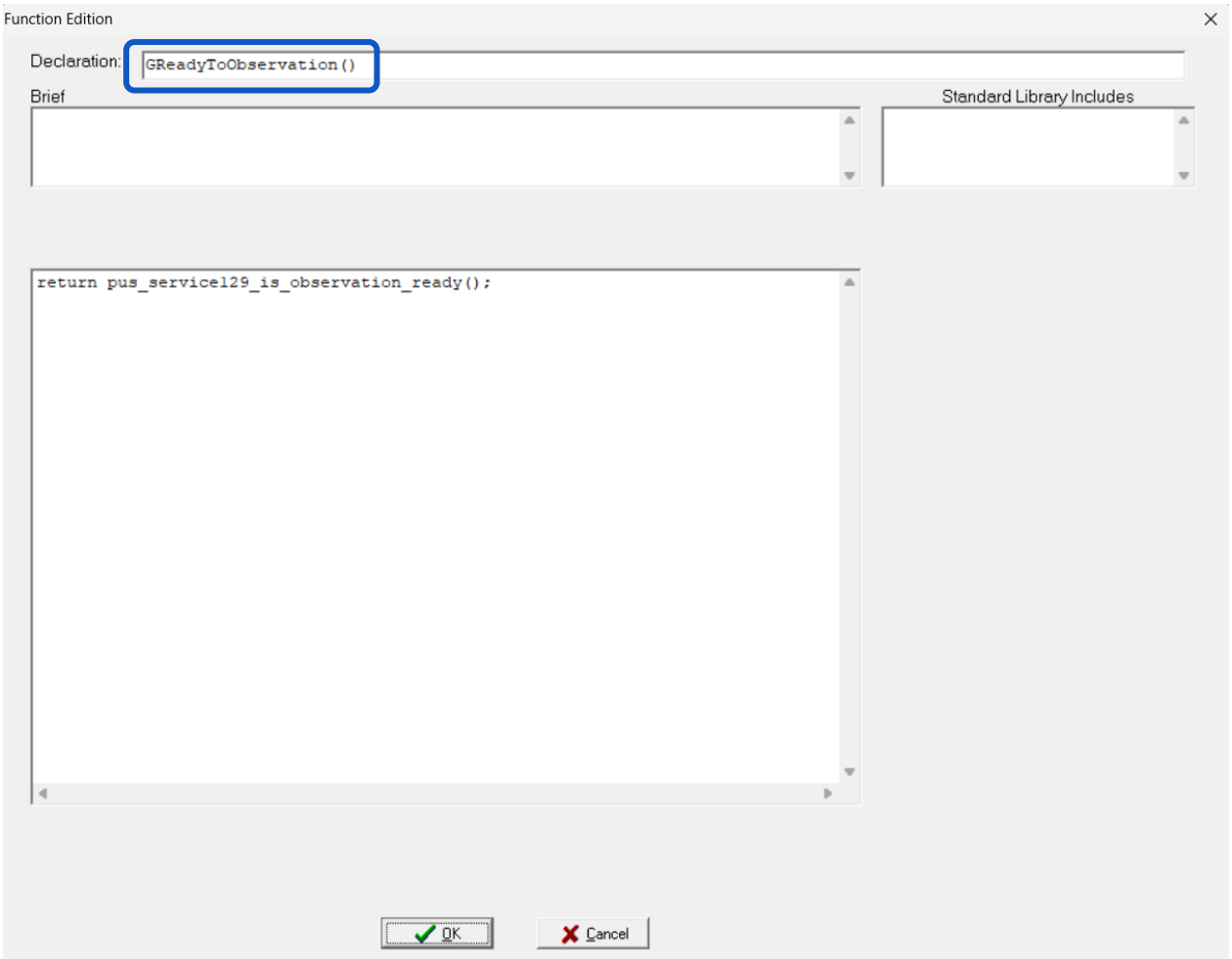




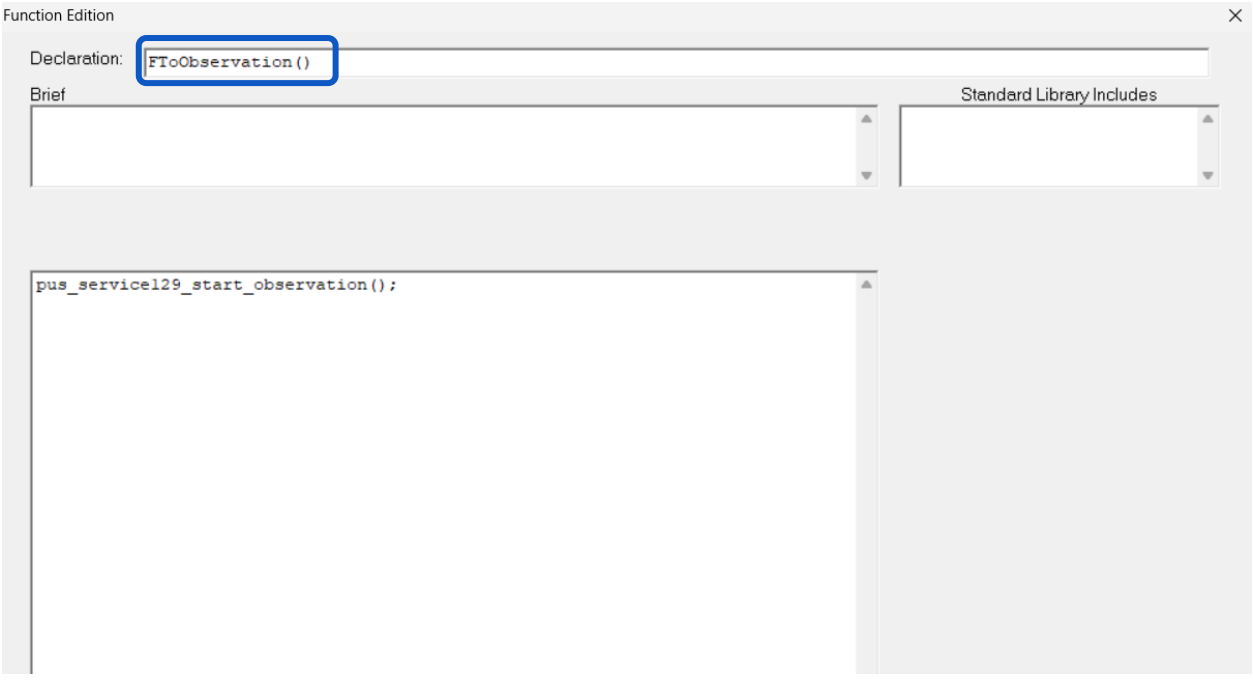
5.4 NODO CHOICEPOINT Y TRANSICIÓN A OBSERVATION

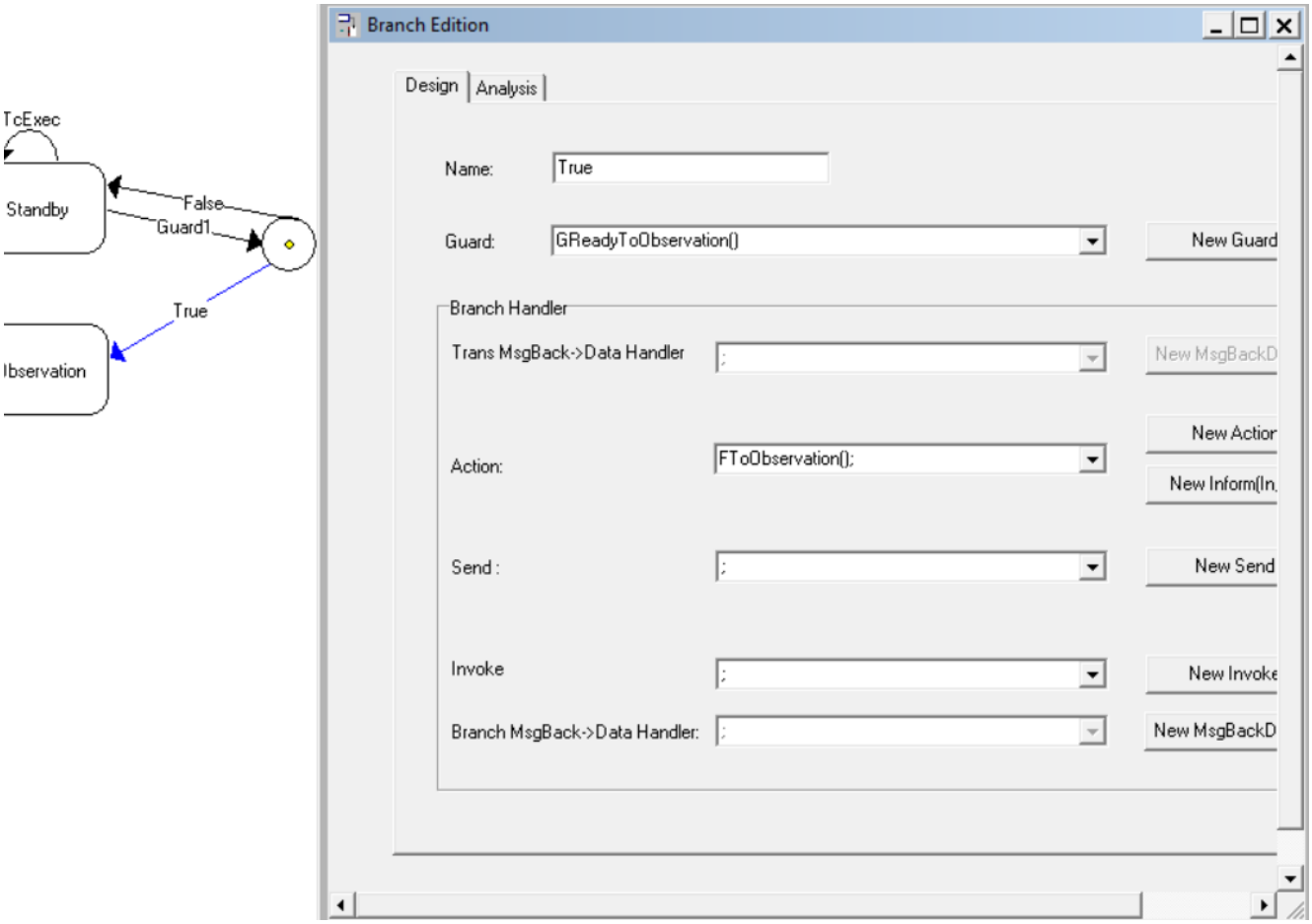
Desde el nodo de decisión **(ChoicePoint)**, si la condición se cumple **(true)**, se realiza una transición hacia el **nuevo estado Observation**.

- Se implementa una transición condicional con la guarda **GReadyToObservation().**

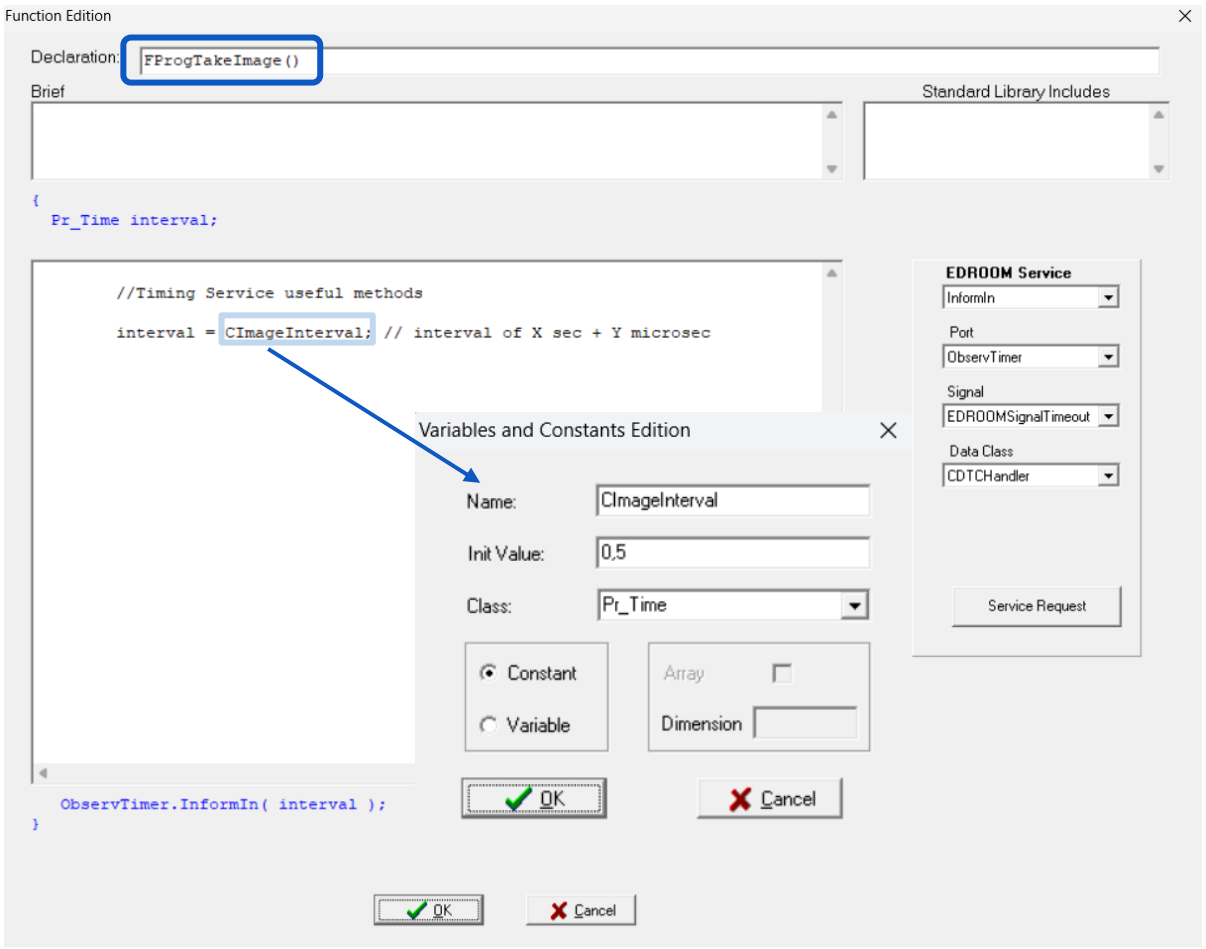


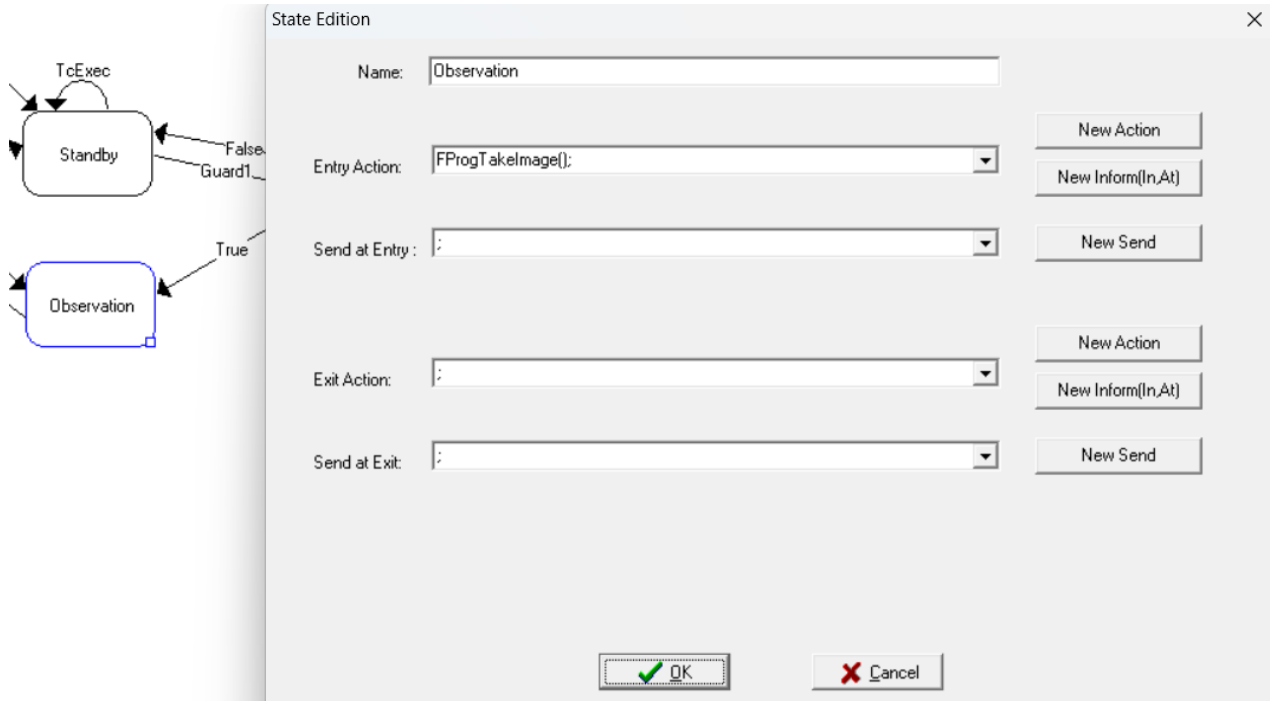
- Si la condición es verdadera, se ejecuta la función **FToObservation()** (que también se ha creado).





- En este estado se define la acción **FProgTakeImage()**, asociada a la constante **CImageInterval.**

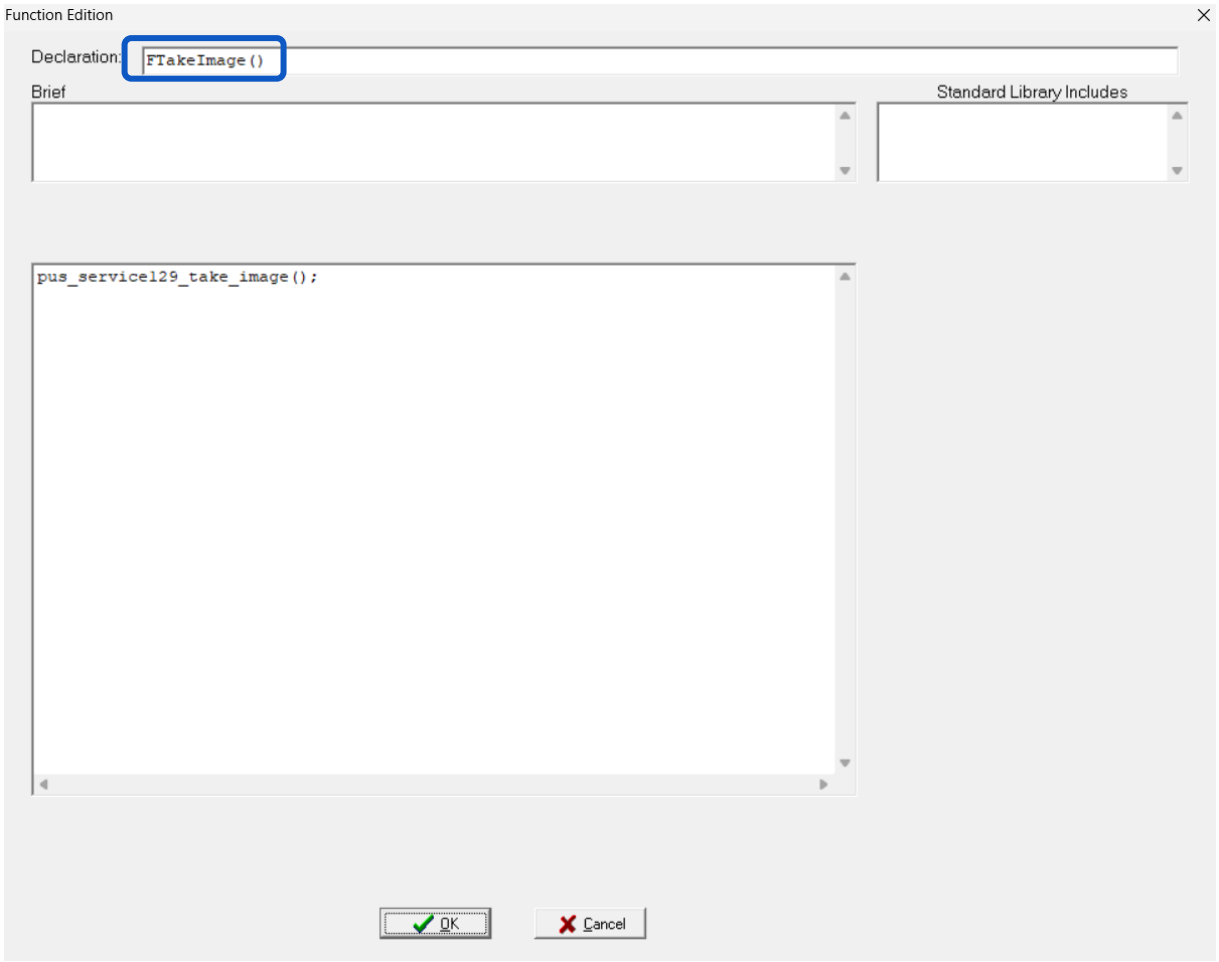


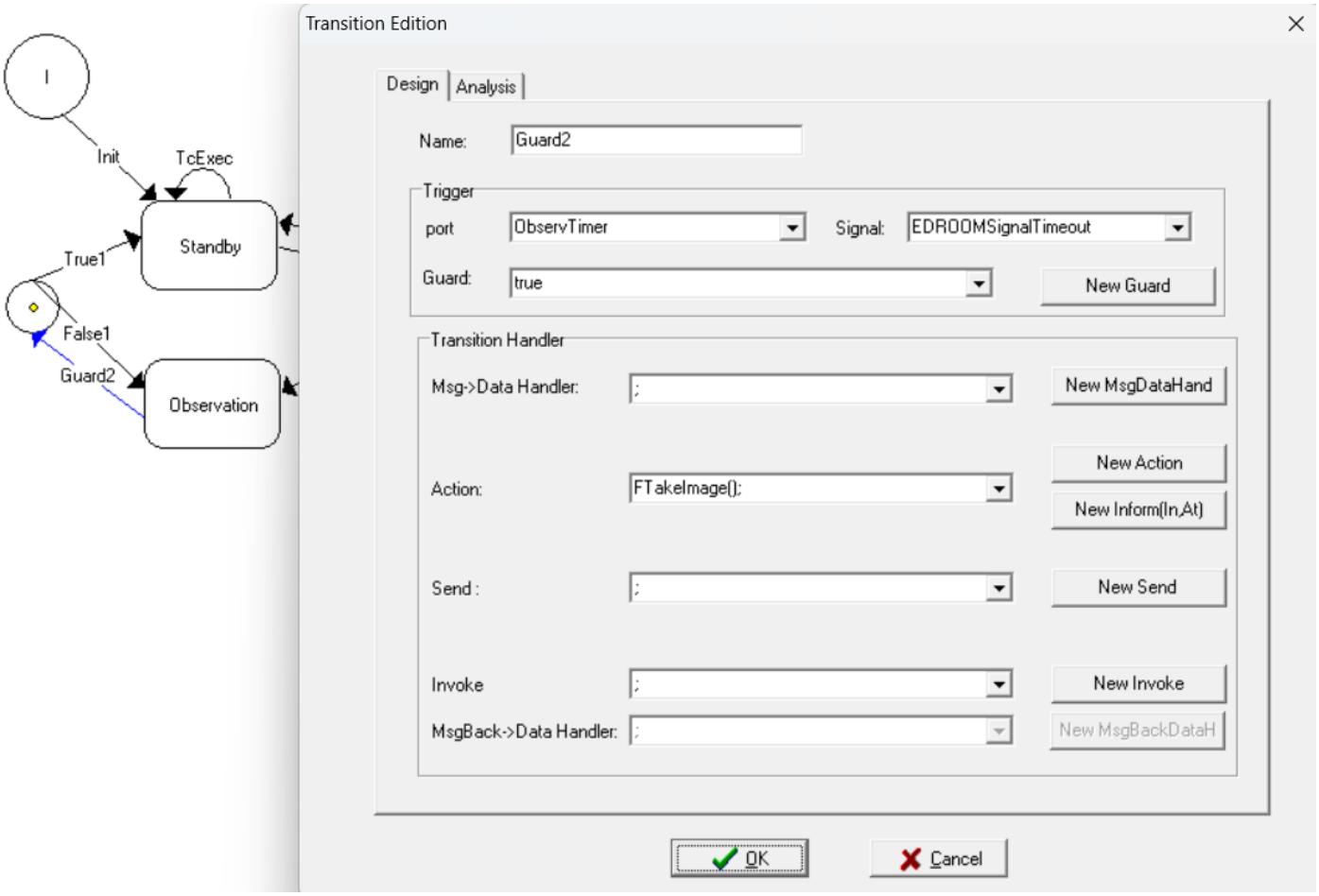


5.5 TRANSICIONES TRAS EL ESTADO OBSERVATION

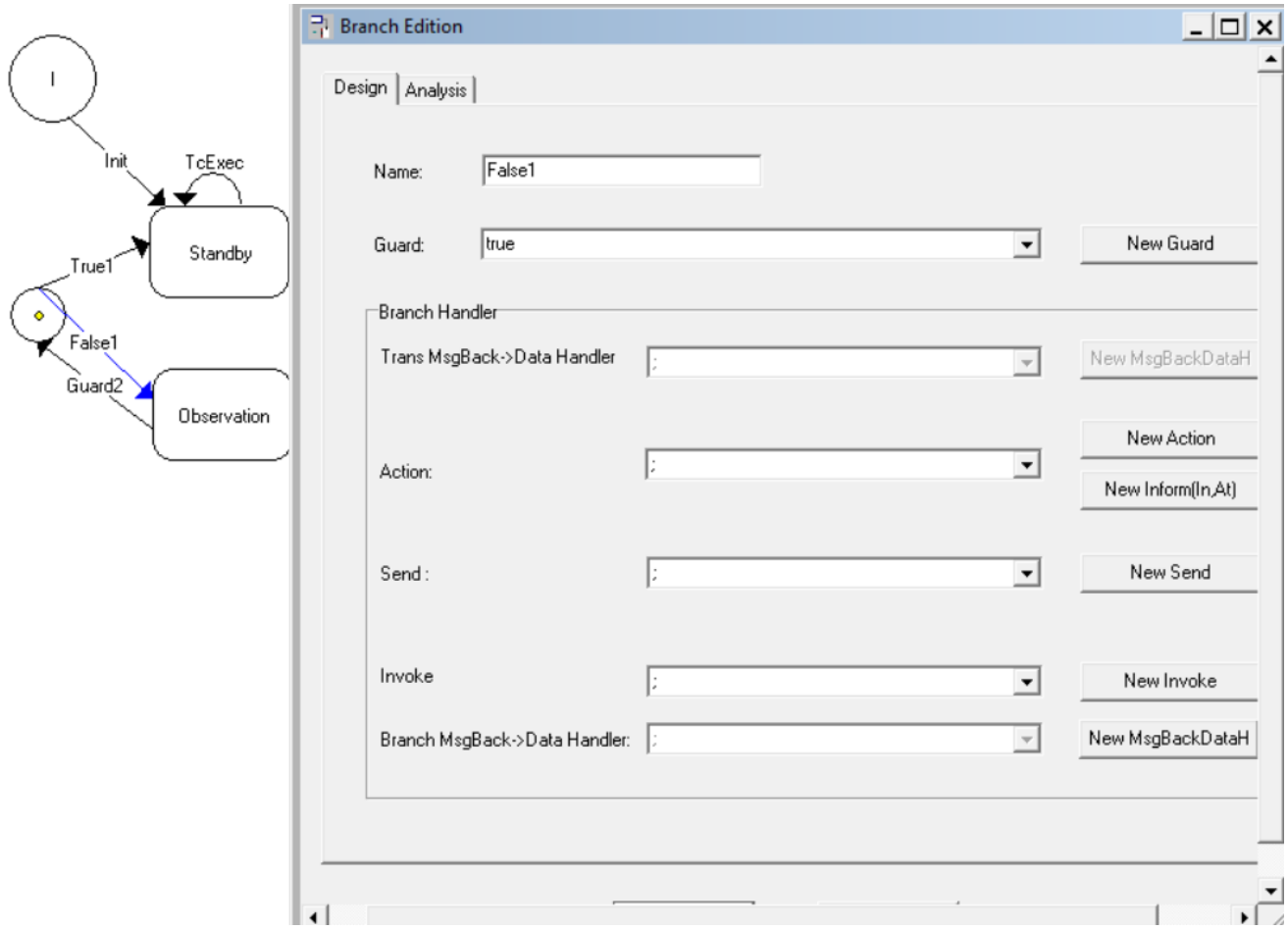
Antes de alcanzar el nodo de decisión **(ChoicePoint)**, se definen dos transiciones que transiciones que determinan si se continúa con la observación o se descarta.

- **Transición Guard2**: Se activa al recibir la señal **EDROOMSignalTimeout** del temporizador ObservTimer, ejecutando la acción **FTakeImage();** para realizar la toma de la imagen.





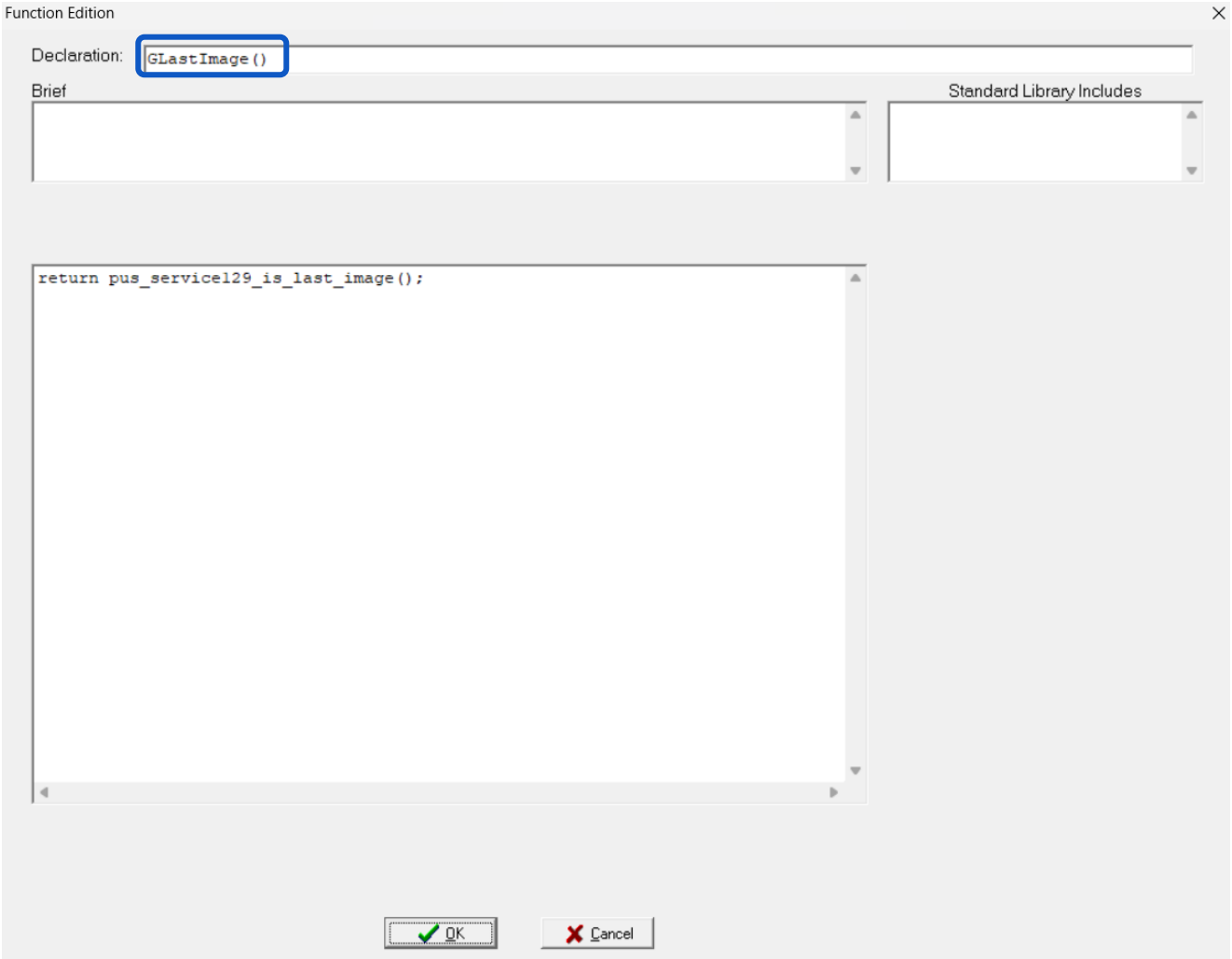
- **Rama False1**: Representa una ruta alternativa con guarda true, que no ejecuta ninguna acción. Sirve como salida de retorno si no se cumplen las condiciones para continuar con la observación.



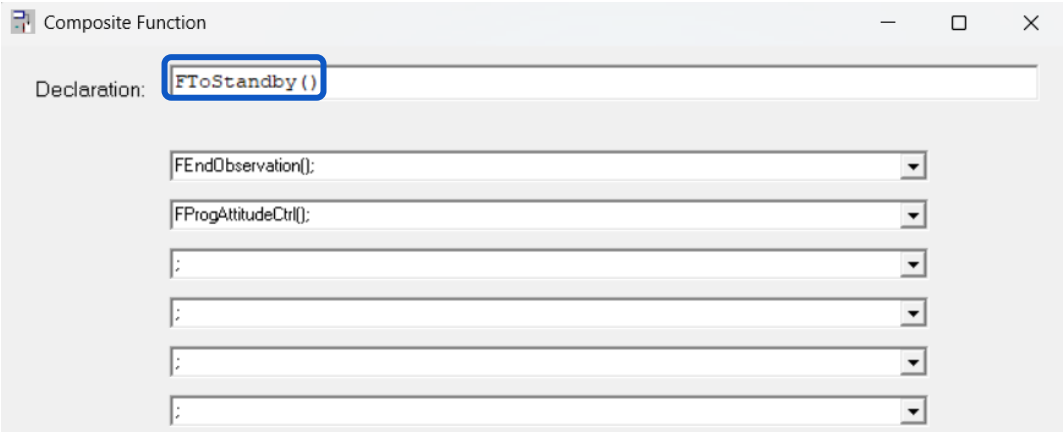
5.6 VUELTA AL ESTADO STANDBY

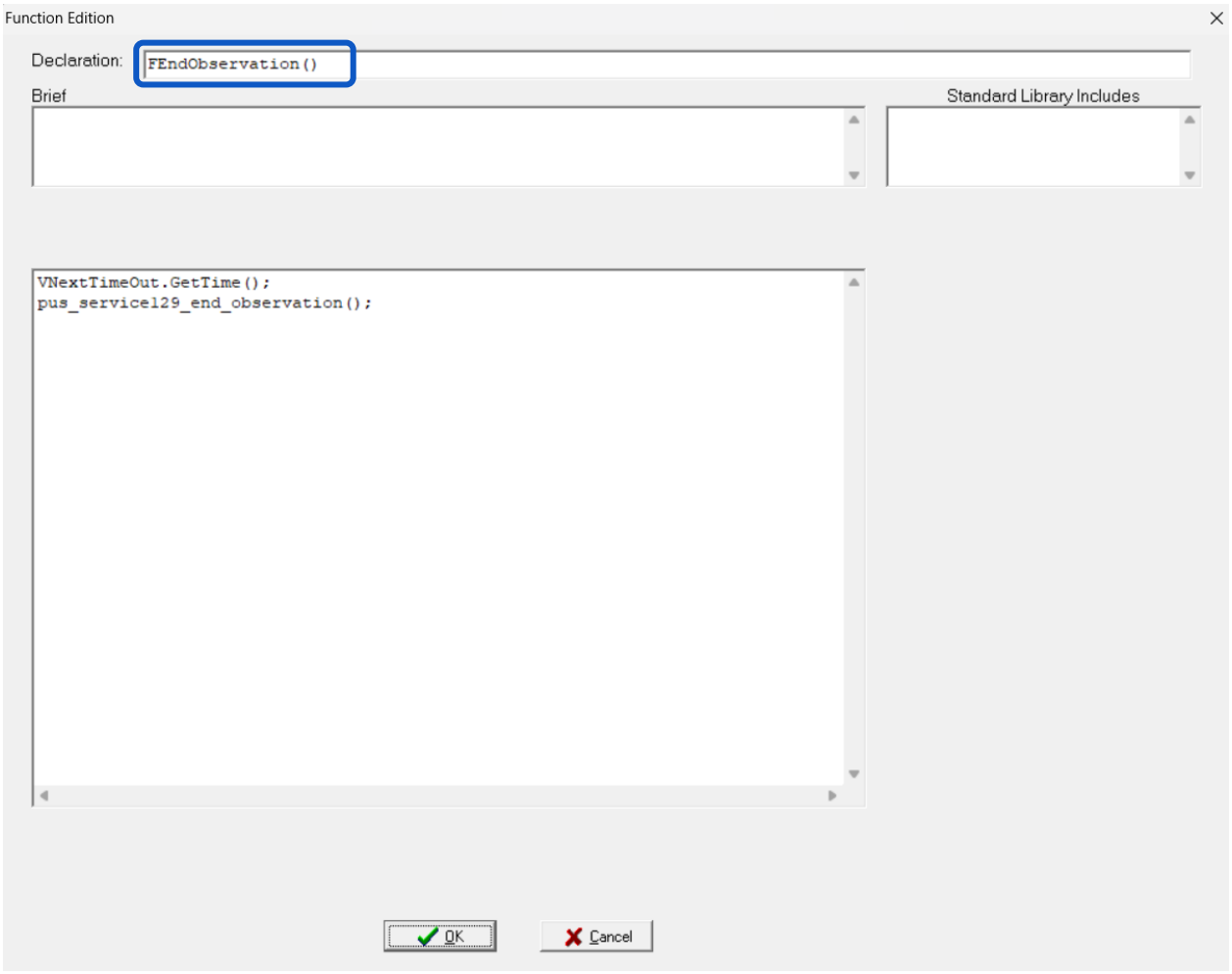
Tras la ejecución completa del ciclo, se vuelve al estado **Standby** mediante una transición con condición verdadera **(True1).**

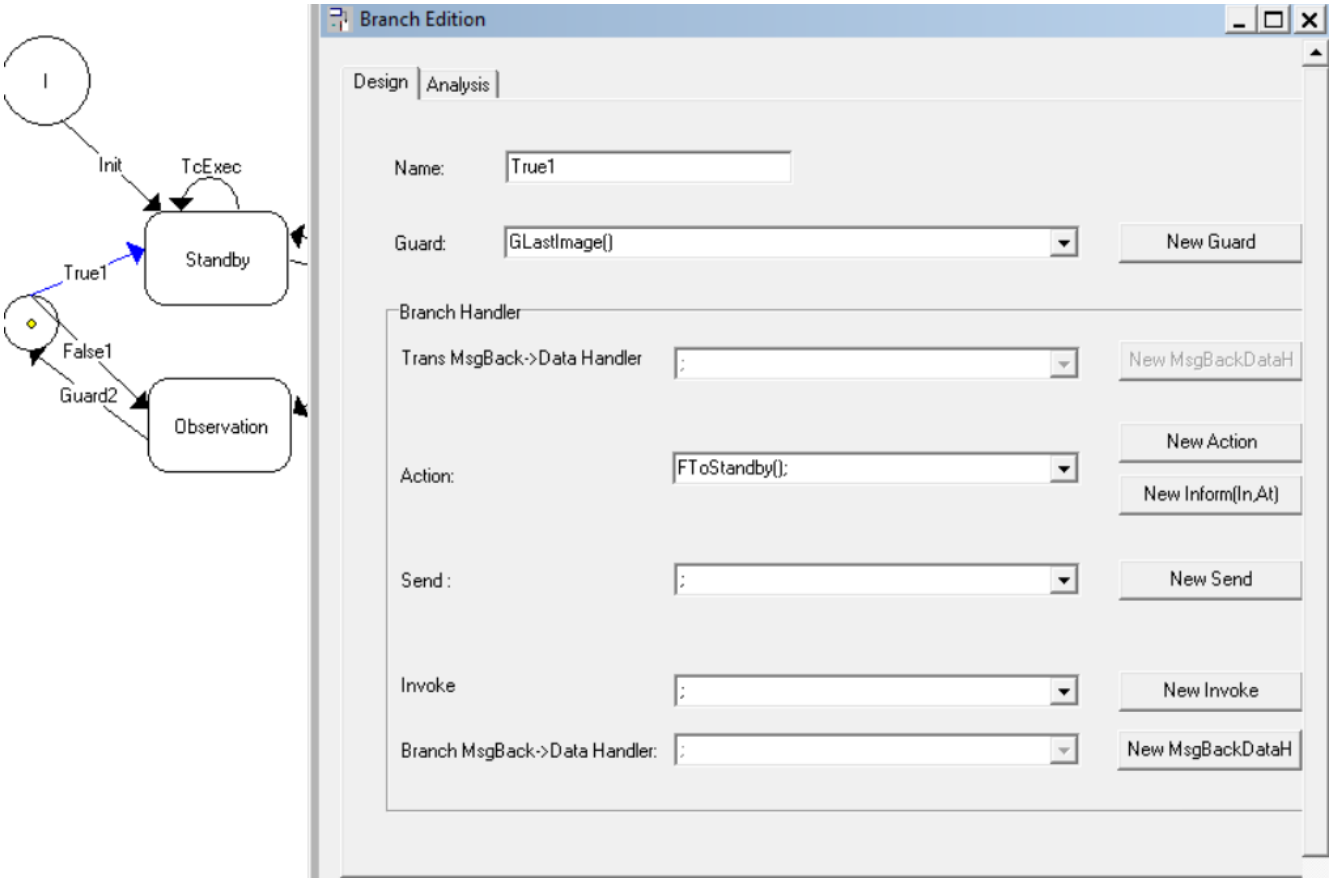
- Esta transición está controlada por la guarda **GLastImage()**, que comprueba si ya se ha tomado la última imagen planificada.



- Si se cumple la condición, se ejecuta la acción **FToStandby();**, encargada de devolver el sistema al estado de espera **(Standby)** y prepararlo para una futura observación.



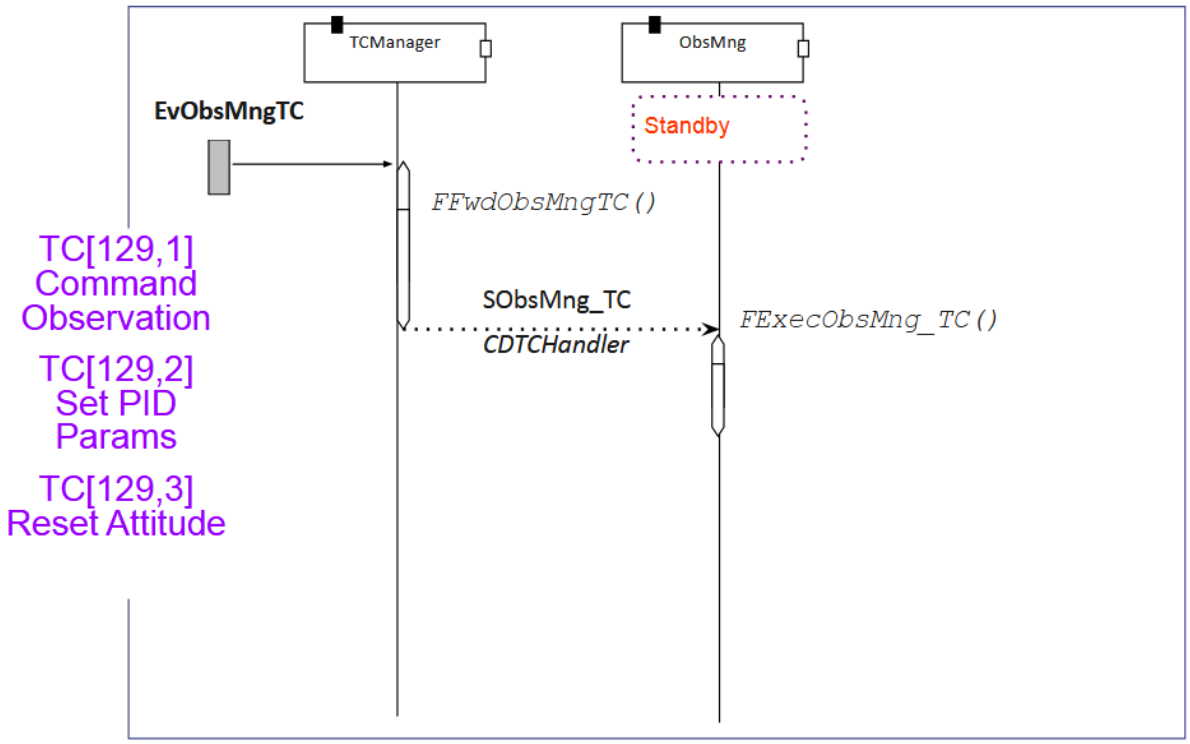

Function Edition
Declaration:
FEndObservation()
Brief
Standard Library Includes
VNextTimeOut.GetTime();
pus_service129_end_observation();
OK
Cancel


I
Init
TcExec
Standby
True1
False1
Guard2
Observation
Branch Edition
Design
Analysis
Name:
True1
Guard:
GLastImage()
New Guard
Branch Handler
Trans MsgBack->Data Handler
New MsgBackDataH
Action:
FToStandby();
New Action
New Inform(In,At)
Send :
New Send
Invoke
New Invoke
Branch MsgBack->Data Handler:
New MsgBackDataH

6. AJUSTES EN EL TCMANAGER

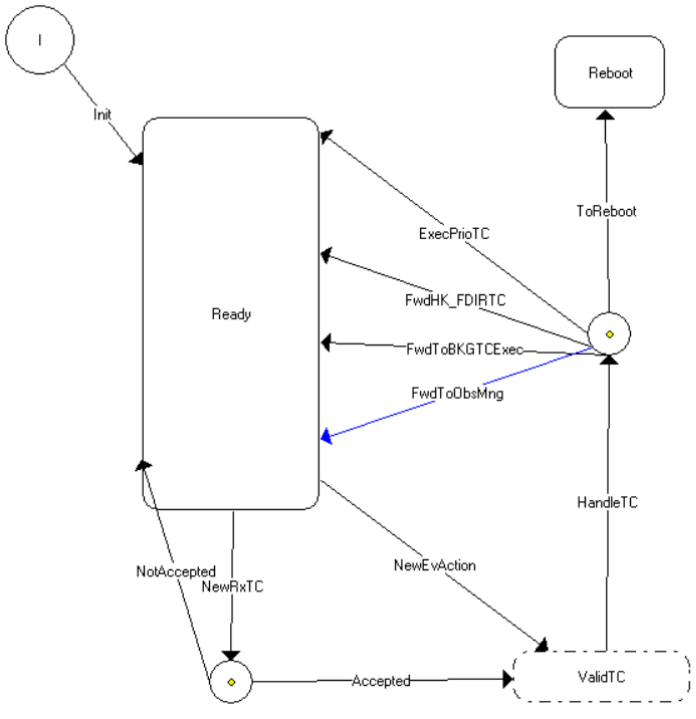
6.1 REVISIÓN DEL DIAGRAMA DE FLUJO

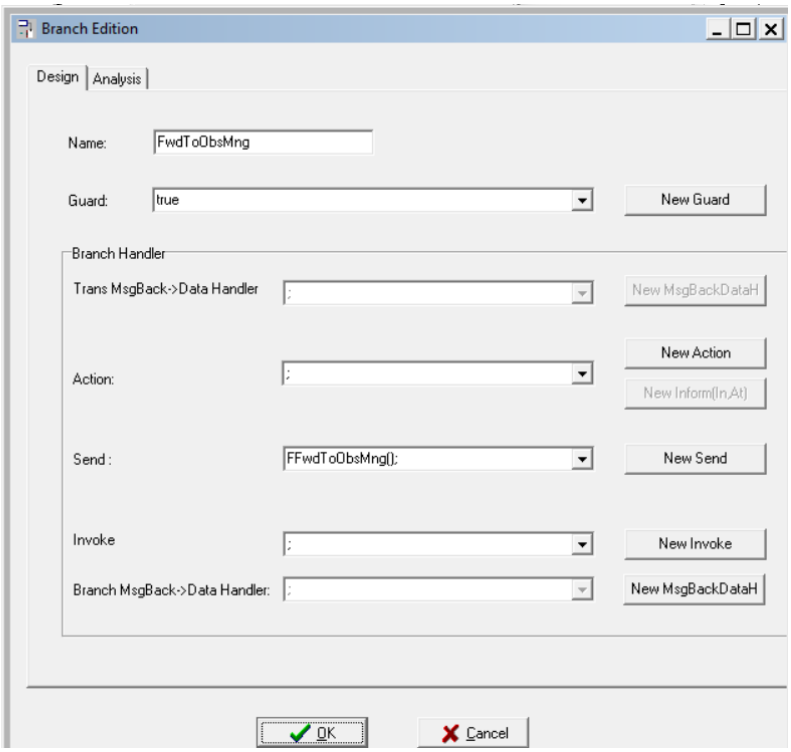
Se analiza el diagrama de flujo del TcManager, prestando especial atención al evento que se envía al ObsMng.



6.2 DEFINICIÓN DEL EVENTO

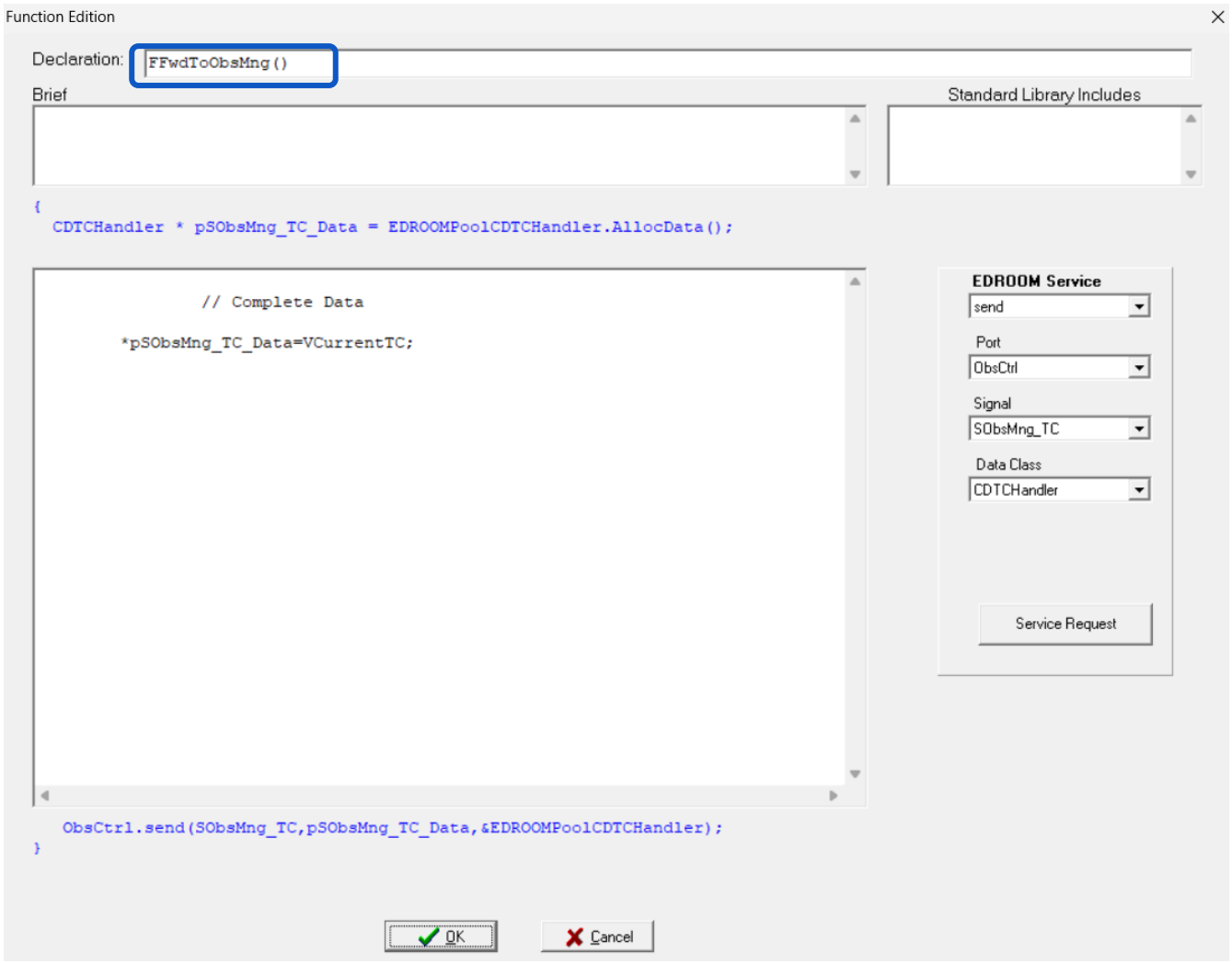
Aunque el evento se lanza correctamente, no estaba definido inicialmente. Se ha procedido a definir formalmente el evento que conecta TcManager con ObsMng.





6.3 COMPROBACIÓN DEL SEND

Finalmente, se valida el envío del evento desde el TcManager al ObsMng, confirmando que se realiza a través de la función Send que la hemos llamado como **FFwdToObsMng().**



7. CONCLUSIONES

La implementación del componente ObsMng se ha realizado siguiendo una arquitectura clara y modular. Se han creado y conectado correctamente los estados, transiciones, funciones y eventos necesarios para que el sistema responda ante un comando de observación enviado desde el TcManager. Esta estructura permite una extensión y mantenimiento sencillo del sistema, cumpliendo con los requisitos del proyecto.