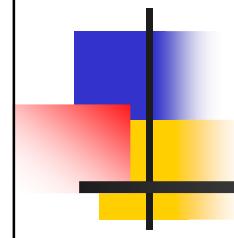




Asociación Civil para la Investigación,
Promoción y Desarrollo de los
Sistemas Electrónicos Embebidos

Seminario de Electrónica Sistemas Embebidos



Laboratorio de
Sistemas Embebidos

LPCXpresso – Yakindu SCT

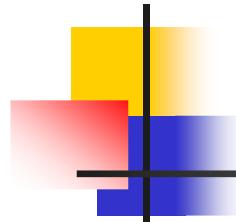


Ing. Juan Manuel Cruz (juanmanuel.cruz@hasar.com)
Gerente de Ingeniería de Cia. Hasar SAIC



Profesor Asociado Ordinario - Técnicas Digitales II TN-FRBA
Profesor Adjunto Interino - Sistemas Embebidos FIUBA

Buenos Aires, 8 de Junio de 2017



Temario

- Diagrama de estados: Yakindu SCT
- Yakindu SCT
-
- Generación de Código (LPCXpresso)
- Diagrama de estados: Paso a Paso
- Convención de Identificadores
- Generación de Código (LPCXpresso)
-
- Referencias

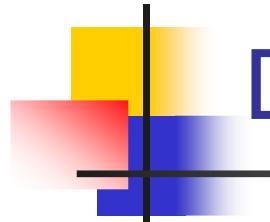
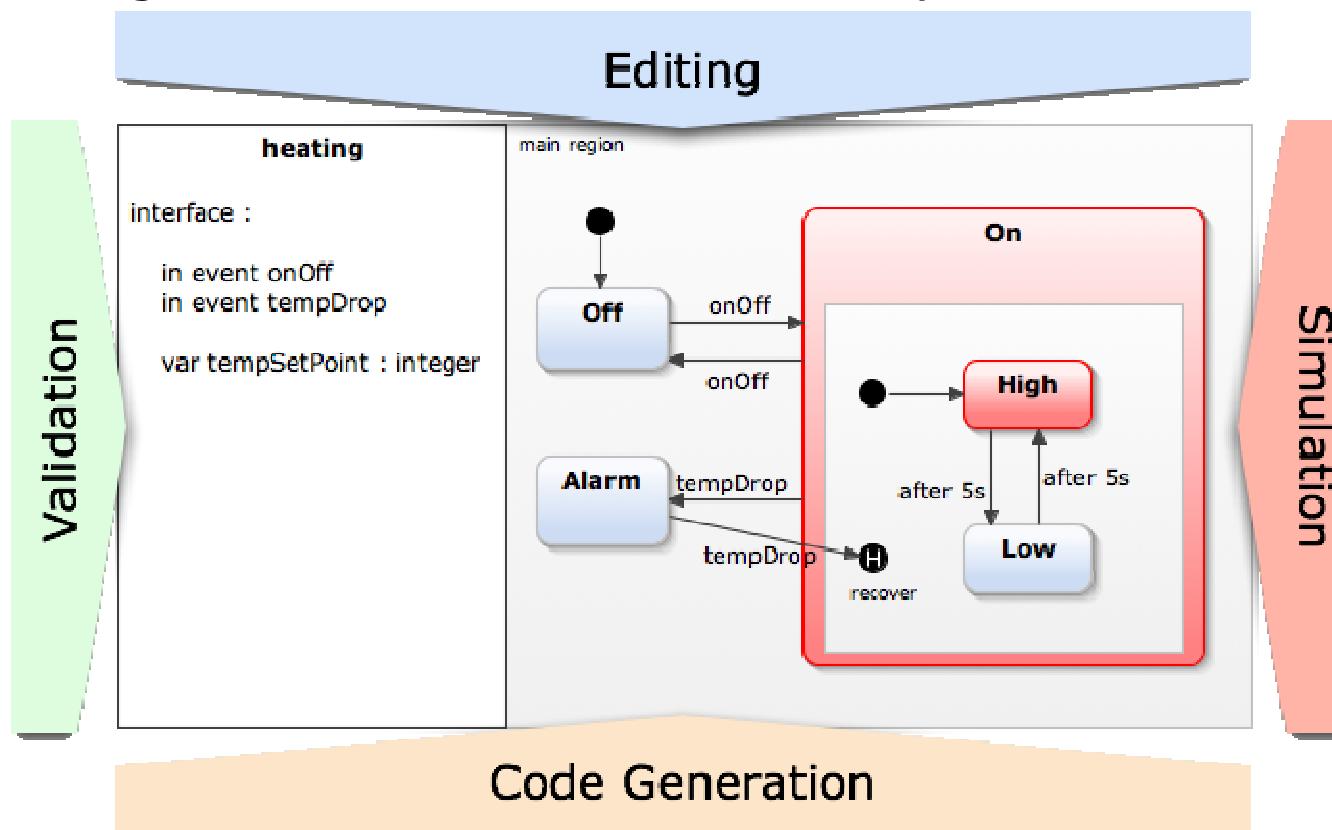


Diagrama de Estado: Yakindu SCT

- **Yakindu** es un kit de herramientas modular para el desarrollo de sistemas embebidos en base a modelos. Basado en la plataforma de desarrollo de código abierto Eclipse
- **Yakindu Statechart Tools (SCT)** es uno de los módulos esenciales del kit y proporciona las siguientes herramientas para hacer frente a los diagramas de máquinas de estado:
 - **Editor:** para crear y editar diagramas de estado
 - **Simulador:** para simular el comportamiento de los statecharts
 - **Generadores de código:** para transformar los statecharts en código Java, C, C ++
 - **Validador integrado:** para verificar problemas sintácticos o semánticos del modelo statechart

Diagrama de Estado: Yakindu SCT

- El gráfico muestra estas características y su relación entre sí:



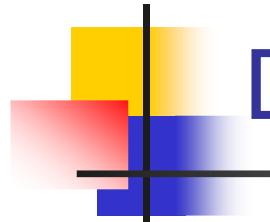
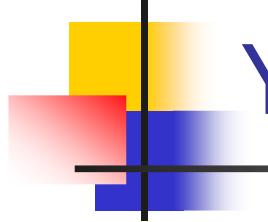


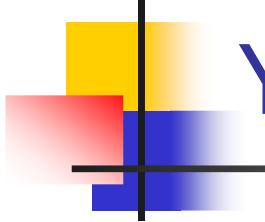
Diagrama de Estado: Yakindu SCT

- El código fuente de Yakindu Statechart Tools (SCT) se proporciona bajo la Licencia Pública de Eclipse
- Los generadores de código abierto que vienen con Yakindu SCT:
 - No implican restricciones de licencia en cuanto al código generado
 - El mismo es propiedad del usuario (como persona u organización)
 - No es necesario hacer del mismo código abierto
 - El usuario es libre de elegir el modelo de licencia del mismo
- La parte principal de Yakindu SCT es un proyecto de código abierto, disponible en www.yakindu.org
- Es útil para el diseño de Finite State Machines y Harel Statecharts (modelar comportamiento)



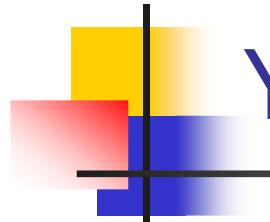
Yakindu SCT: plugin, New Statechart

- Se puede descargar el kit de herramientas completo o instalar un **plugin** (<http://updates.yakindu.org/sct/releases/>) a la distribución de LPCXpresso con que trabajamos en clase (siguiendo las instrucciones de la guía de Trabajos Prácticos)
- Para crear un nuevo archivo statechart:
 - En la vista **Project Explorer (Warkspace)**
 - Haga clic con el **botón derecho** del mouse en un proyecto o en una carpeta en la que desee crear el nuevo diagrama de estado
 - Surge un menú de contexto, en él seleccione Nuevo → **Otros**
 - Aparecerá el cuadro de diálogo **New**, en él seleccione YAKINDU SCT → **Statechart Model**



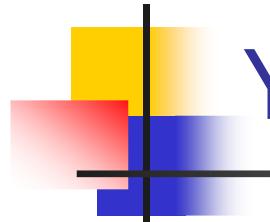
Yakindu SCT: Create statechart

- Aparecerá una ventana **New YAKINDU Statechart** (wizard)
- Introduzca un **nombre de archivo** para el archivo del statechart que se va a crear (su extensión debe ser **.sct**)
- Además puede editar el proyecto o la carpeta destino y haga clic en **Next>**
- Seleccione el dominio del statechart y haga clic en **Finish**
- Si aparece el cuadro de diálogo **Confirm Perspective Switch** responda **Yes** para incluir una nueva perspectiva: **SC Modeling**
- El resultado es que el nuevo archivo **statechart** ha sido **creado** en la ubicación que especificó y **abierto** por el editor de statechart
- Para copiar un nuevo archivo statechart existente:
 - Recurra a las opciones del menú de contexto: **Copy & Paste**



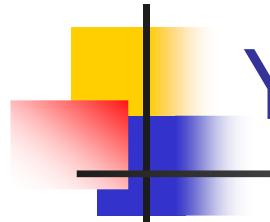
Yakindu SCT: Delete statechart

- Para eliminar un archivo statechart:
 - En la vista **Project Explorer (Warkspace)**
 - Haga clic con el **botón derecho** del mouse sobre el archivo que desee eliminar
 - Surge un menú de contexto, en él seleccione **Delete**
 - Aparecerá el cuadro de diálogo **Delete Resources**, que brinda tres opciones:
 - Haga clic en **Preview>** para inspeccionar lo que la operación de eliminación va a hacer (antes de confirmar)
 - Haga clic en **Cancel** para cancelar la operación de borrado (el archivo statechart permanecerá en el proyecto)
 - Haga clic en **OK** para eliminar realmente el archivo statechart



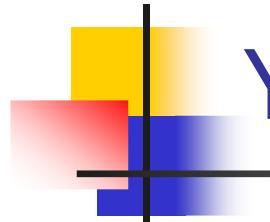
Yakindu SCT: SC Modeling

- Es una perspectiva de Eclipse que soporta el modelado de statecharts, ella define los siguientes puntos de vista:
 - **Project Explorer** (izquierda): muestra su workspace y proyectos, carpetas y archivos contenidos en él. Util para inspeccionar la estructura interna de los modelos statechart
 - **Properties** (abajo): muestra las propiedades en relación con el modelo semántico o la apariencia gráfica del elemento seleccionado en el editor de statechart. Permite editarlas
 - **Problems** (abajo): muestra los errores y advertencias existentes del workspace. Al hacer doble clic en una entrada normalmente se abre la ubicación del error o advertencia correspondiente
 - **Outline** (derecha): vista de ojo de pájaro del statechart abierto. También indica el punto de vista actual para una mejor orientación en modelos grandes



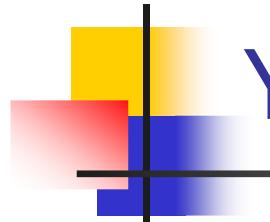
Yakindu SCT: SC Modeling

- El archivo del statechart (arriba) se presenta como dos áreas de edición:
 - Area **gráfica** (arriba derecha)
 - Agregar o eliminar una región
 - Hacer Zoom (Ctrl & girar rueda del mouse)
 - Abrir Menú de contexto (botón derecho del mouse)
 - Agregar elementos gráficos previamente seleccionados en **Palette** (arriba más a la derecha)
 - Area de **texto** (arriba izquierda)
 - Se recomienda completarla agregando/eliminando/editando elementos en la vista **Properties** (abajo)



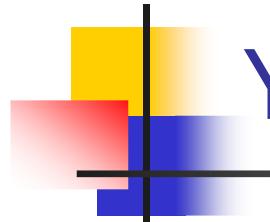
Yakindu SCT: SC Modeling

- Para Simular un archivo statechart:
 - En la vista **Project Explorer (Warkspace)**
 - Haga clic con el **botón derecho** del mouse en un proyecto o en una carpeta en la que desee crear el nuevo diagrama de estado
 - Surge un menú de contexto, seleccione Run As → **1 Statechart Simulation**
 - Aparecerá una nueva perspectiva: **SC Simulation**, habiéndose abierto una instancia de Debug de statecharts (se resalta en rojo el estado actual)
 - En la vista **Simulation** (abajo a la derecha) se puede excitar al statechart mediante eventos, modificar sus variables y ver las acciones que se ejecutan



Yakindu SCT: SC Modeling

- Para generar código es necesario agregar un nuevo archivo:
 - En la vista **Project Explorer (Warkspace)**
 - Haga clic con el **botón derecho** del mouse en la carpeta en la que ha creado el diagrama de estado
 - Surge un menú de contexto, en él seleccione Nuevo → **Otros**
 - Aparecerá el cuadro de diálogo **New**, en él seleccione YAKINDU SCT → **Code Generator Model**
 - Aparecerá una ventana **New YAKINDU Statechart** (wizard)
 - Surgirá un **nombre de archivo** para el archivo de generación (su extensión debe ser **.sgen**) y haga clic en **Next>**
 - Seleccione el lenguaje (**C**) y el archivo statechart fuente (**.sct**) fuente. Además puede editar el proyecto o la carpeta destino y haga clic en **Finish**

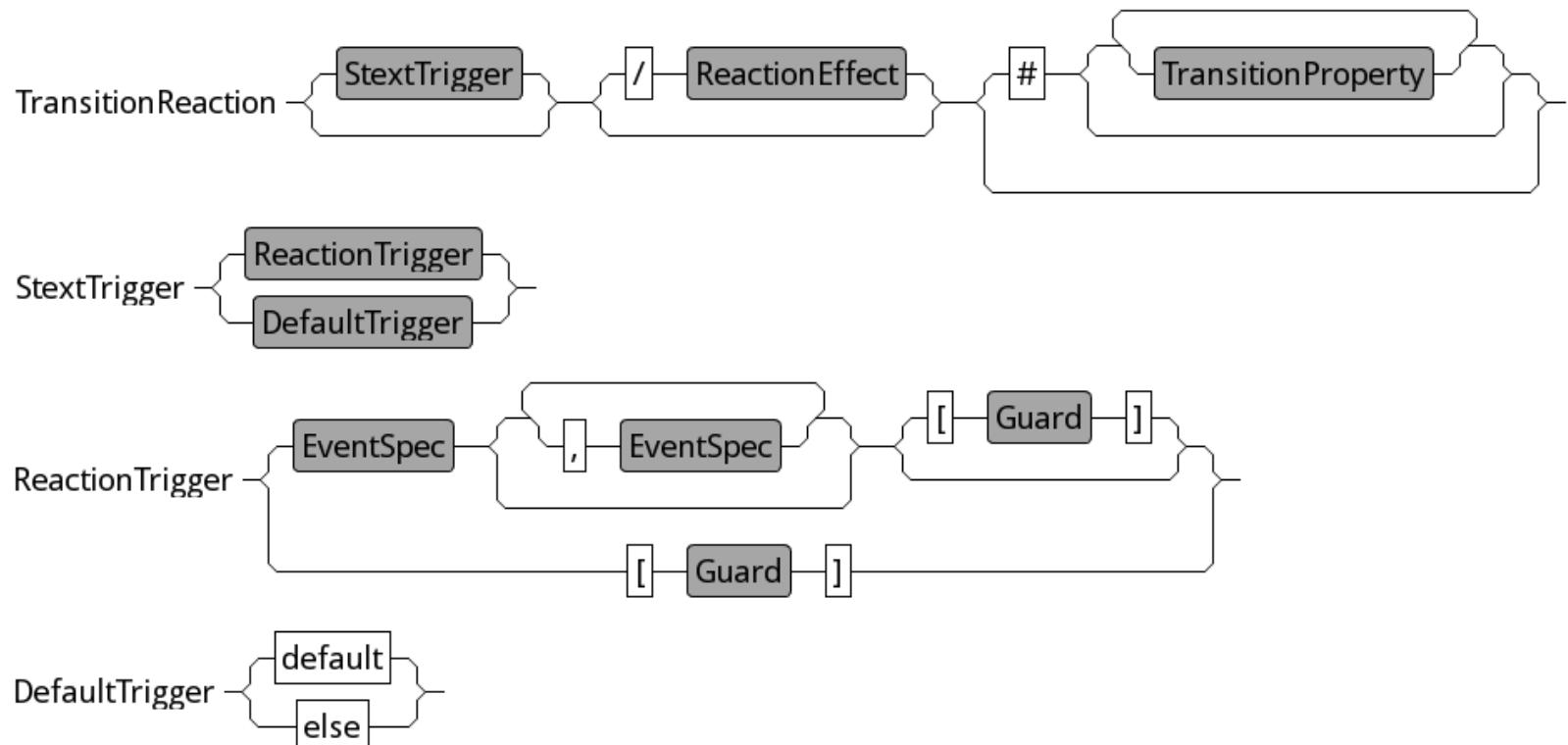


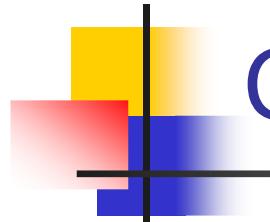
Yakindu SCT: SC Modeling

- En la vista **Project Explorer (Warkspace)**
- Haga clic con el **botón derecho** del mouse sobre el archivo **.sgen**
- Surge un menú de contexto, en él seleccione **Generate Code Artifacts** (preste atención a los mensajes de Consola)
- O bien mediante **Build Project** (o **Build All**)
- En la carpeta seleccionada se generarán:
 - **xxxx.c** y **xxxx.h** (fuentes con el nombre del statechart **xxxx.sct**)
 - **sc_types.h** (prototipos de variables)
 - **xxxxRequired.h** (prototipos de funciones)
- Resta vincular el código generado con los fuentes de la aplicación

Yakindu SCT: SC Modeling

- Syntax: Trigger [Guard] / Effect



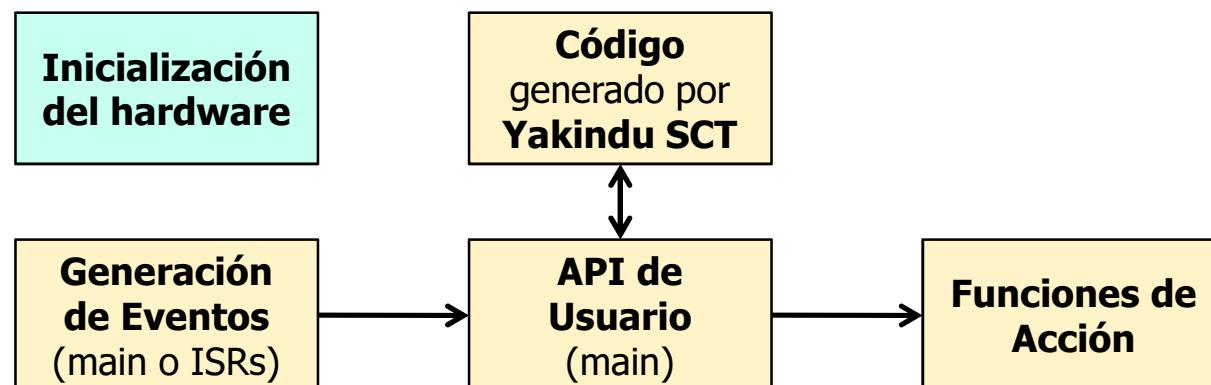


Generación de Código (LPCXpresso)

- Para **compilar** los fuentes del **código generado** por **Yakindu SCT** **con** nuestras propias herramientas (**LPCXpresso** - configurado para el micro elegido para la aplicación) se necesita **combinar código fuente**:
 - Código generado por el **usuario**
 - Código generado automáticamente por **Yakindu SCT**
- Por lo que el diseñador tiene que escribir el siguiente código:
 - Código p/inicializar el **hardware**
 - Código p/procesar dispositivos de **salida (funciones de acción / drivers de salida)**
 - Código p/procesar **entradas (generación de eventos / manejar cola de eventos / drivers de entrada)**
 - La función **main** e **ISRs** (rutinas de atención de interrupciones) que configuran la **API** (Application Programming Interface) de **usuario**

Generación de Código (LPCXpresso)

- La solución es simple y consiste en:
 - Asegurar la **Generación de Eventos** (producto del procesamiento del programa principal o de las interrupciones que se produzcan)
 - Dentro del main se ejecuta la **Aplicación de Usuario** que monitorea flags, setea **eventos**, invoca al **código generado por Yakindu SCT** e **iterar**
 - Con la consecuente ejecución en **secuencia** de las **Funciones de Acción**



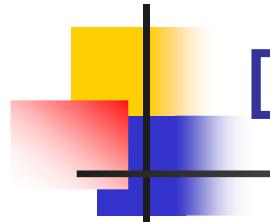
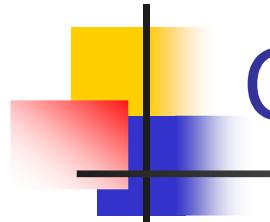


Diagrama de Estado: Paso a Paso

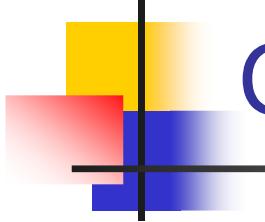
- Partir de las especificaciones del sistema
 - Primer paso Identificar los **eventos** y las **acciones**
 - Segundo paso Identificar los **estados**
 - Tercer paso Agrupar por **jerarquías**
 - Cuarto paso Agrupar por **concurrencia**
 - Quinto paso Añadir las **transiciones**
 - Sexto paso Añadir las **sincronizaciones**
- Elegida una herramienta de software podremos: **Editar**, **Verificar** y **Validar (Simular)** el diagrama de estado
- Culminando con la **generación del código** (opción posible dependiendo de la herramienta de software)



Convención de Identificadores

- A fin de no confundir los elementos del diagrama de estado adoptaremos la siguiente convención para identificarlos (afín a la documentación de **Yakindu SCT**)

■ Tipografía	Elemento	Prefijo
■ CAMELcASE:	State	
■ xxCamelCase:	Event (External)	(xx → ev)
	Action Function (External)	(xx → op)
	Signal (Internal Event)	(xx → si)
	Internal Variable	(xx → vi)
	External Variable	(xx →)
■ xxCAMELcASE:	Constant	



Generación de Código (LPCXpresso)

- P/simplificar nombraré al archivo statechart creado: **Prefix.sct**
- Resultando el archivo de generación **Prefix.sgen**
 - Si desea tener generar todos los fuentes en la misma carpeta modifique **targetFolder = ".../src-gen"** (path de la carpeta de fuentes de la aplicación dentro del proyecto)
 - Idem anterior: **libraryTargetFolder = ".../src-gen"**
 - Se generarán 4 (cuatro archivos), a saber:
 - **Prefix.c** y **Prefix.h** (fuentes con el nombre del statechart **Prefix.sct**)
 - **sc_types.h** (prototipos de variables)
 - **PrefixRequired.h** (prototipos de funciones)

Generación de Código (LPCXpresso)

The screenshot shows the LPCXpresso IDE interface. The title bar has tabs for "Welcome", ".c statechart.c", and "prefix.sct". The "prefix.sct" tab is highlighted with a red box. The main window displays a statechart diagram and its corresponding C code.

Statechart Prefix:

```
prefix
main region
```

C Code:

```
/* Blink LED3 EDU-CIA-NXP */
interface:
in event evTick
operation opLED(LEDNumber:integer,
State:boolean)

//const LEDR: integer = 0
//const LEDG: integer = 1
//const LEDB: integer = 2
//const LED1: integer = 3
//const LED2: integer = 4
const LED3: integer = 5

const LED_ON: boolean = false
const LED_OFF: boolean = true

internal:
var viTitilar: integer

const ci250mS: integer = 250
const ci500mS: integer = 500
```

Statechart Diagram:

- APAGADO:** entry / opLED(LED3, LED_OFF); viTitilar = ci250mS
evTick [viTitilar != 0] / viTitilar -= 1
- ENCENDIDO:** entry / opLED(LED3, LED_ON); viTitilar = ci500mS
evTick [viTitilar != 0] / viTitilar -= 1

A black dot representing an initial state has arrows pointing to both the APAGADO and ENCENDIDO states. Each state has a self-loop arrow labeled "evTick[viTitilar == 0]".

Generación de Código (LPCXpresso)

The screenshot shows the LPCXpresso IDE interface. The title bar says "prefix.sct". The left pane displays a "Statechart prefix" containing C code. The right pane shows a state transition diagram.

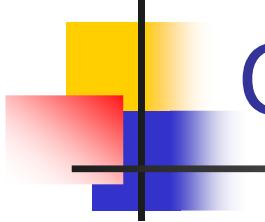
Code in prefix:

```
/* Blink TimeEvent LED3 EDU-CIA-NXP */  
interface:  
operation opLED(LEDNumber:integer,  
    State:boolean): void  
  
//const LEDR: integer = 0  
//const LEDG: integer = 1  
//const LEDB: integer = 2  
//const LED1: integer = 3  
//const LED2: integer = 4  
const LED3: integer = 5  
  
const LED_ON: boolean = false  
const LED_OFF: boolean = true  
  
internal:
```

State Transition Diagram (main region):

```
graph LR; Start(( )) --> APAGADO[APAGADO  
entry / opLED(LED3, LED_OFF)]; APAGADO --> ENCENDIDO[ENCENDIDO  
entry / opLED(LED3, LED_ON)]; ENCENDIDO --> APAGADO;  
APAGADO -- "after 250 ms" --> ENCENDIDO;  
ENCENDIDO -- "after 500 ms" --> APAGADO;
```

The diagram consists of two states: "APAGADO" and "ENCENDIDO". Transitions between them are triggered by "opLED" events. The "APAGADO" state has an entry action "opLED(LED3, LED_OFF)". The "ENCENDIDO" state has an entry action "opLED(LED3, LED_ON)". Transitions are labeled "after 250 ms" and "after 500 ms".



Generación de Código (LPCXpresso)

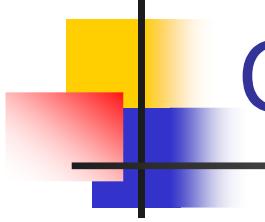
```
#include "board.h"
#include "src-gen/Prefix.h"
#include "TimerTicks.h"

/* Public types/enumerations/variables */
volatile bool SysTick_Time_Flag = false;

/*! This is a state machine */
Static Prefix statechart;

#define __USE_TIME_EVENTS (false) // "true" with TimerEvents or "false" without TimeEvents

/*! This is a state machine that requires timer services */
#if (__USE_TIME_EVENTS == true)
    #define NOF_TIMERS (sizeof(PrefixTimeEvents)/sizeof(sc_boolean))
#else
    #define NOF_TIMERS 0
#endif
```



Generación de Código (LPCXpresso)

```
TimerTicks ticks[NOF_TIMERS];

/* This state machine makes use of operations declared in the state machines interface or
internal scopes. */

void prefixIface_opLED(Prefix* handle, sc_integer LEDNumber, sc_boolean State)
{
    Board_LED_Set((uint8_t) LEDNumber, State);
}

#if (__USE_TIME_EVENTS == true)
// This function has to set up timers for the time events that are required by the state machine
void prefix_setTimer(Prefix* handle, const sc_eventid evid, const sc_integer time_ms, const
sc_boolean periodic)
{
    SetNewTimerTick(ticks, NOF_TIMERS, evid, time_ms, periodic);
}
```

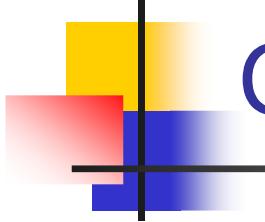


Generación de Código (LPCXpresso)

```
// This function has to unset timers for the time events that are required by the state machine
void prefix_unsetTimer(Prefix* handle, const sc_eventid evid)
{
    UnsetTimerTick(ticks, NOF_TIMERS, evid);
}
#endif

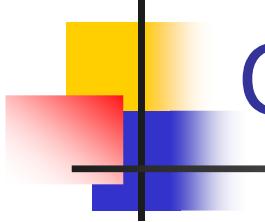
/* Handle interrupt from SysTick timer */
void SysTick_Handler(void)
{
    SysTick_Time_Flag = true;
}

int main(void)
{
    #if (__USE_TIME_EVENTS == true)
    uint32_t i;
    #endif
```



Generación de Código (LPCXpresso)

```
.....  
Board_Init();  
  
/* Statechart Initialization in main*/  
#if (__USE_TIME_EVENTS == true)  
InitTimerTicks(ticks, NOF_TIMERS);  
#endif  
  
prefix_init(&statechart);  
prefix_enter(&statechart);  
  
/* LEDs toggle in main loop */  
while (1) {  
    __WFI();  
  
    if (SysTick_Time_Flag == true) {  
        SysTick_Time_Flag = false;
```

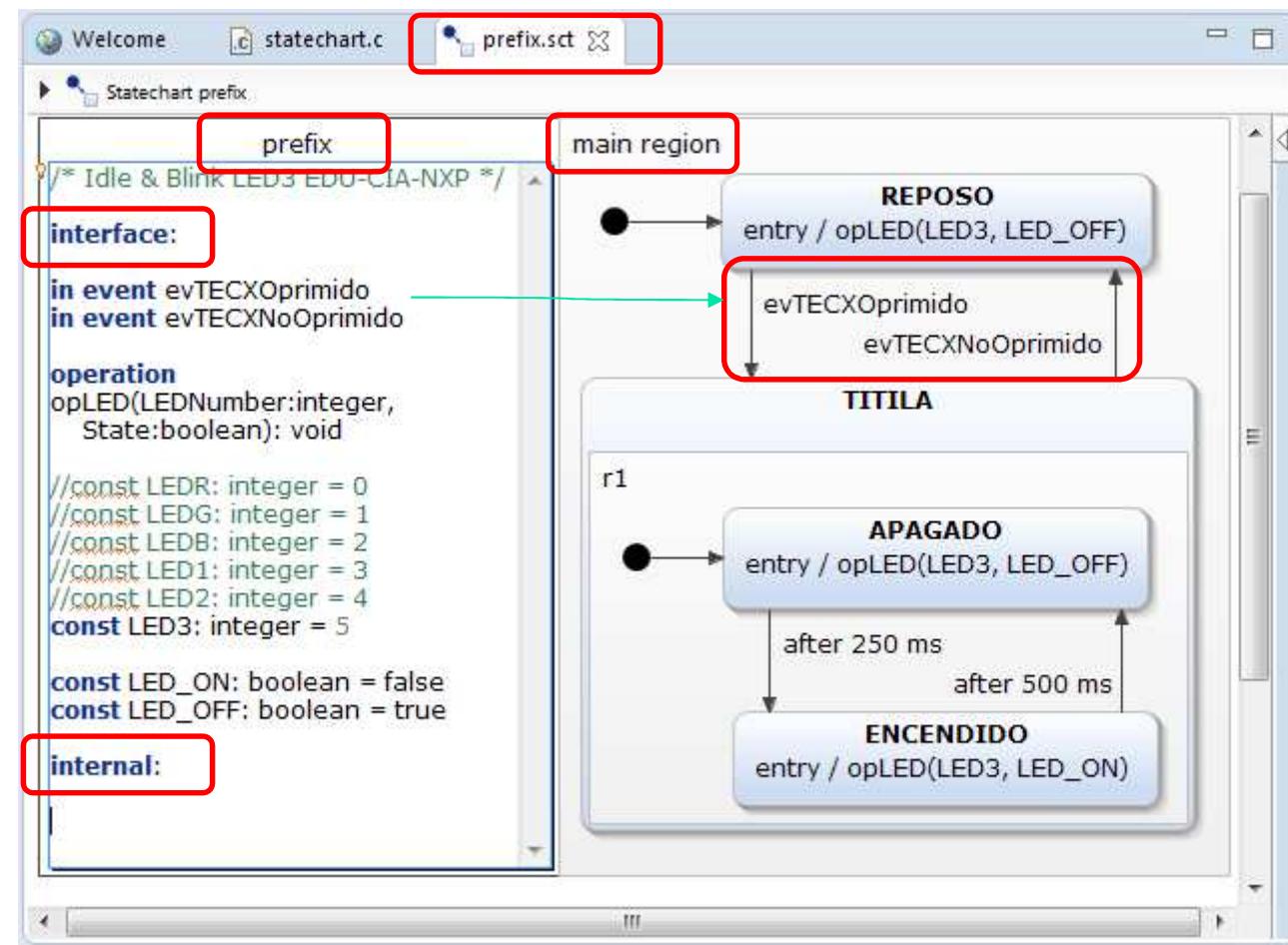


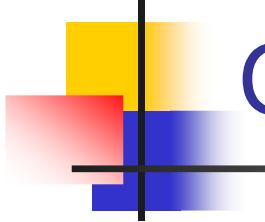
Generación de Código (LPCXpresso)

```
#if (__USE_TIME_EVENTS == true)
UpdateTimers(ticks, NOF_TIMERS);
for (i = 0; i < NOF_TIMERS; i++) {
    if (IsPendingEvent(ticks, NOF_TIMERS, ticks[i].evid) == true) {
        prefix_raiseTimeEvent(&statechart, ticks[i].evid); // Event->Ticks.evid=>OK
        MarkAsAttEvent(ticks, NOF_TIMERS, ticks[i].evid);
    }
    #else
        prefixIface_raise_evTick(&statechart);           // Event -> evTick => OK
    #endif

    prefix_runCycle(&statechart);                      // Run Cycle of Statechart
}
}
```

Generación de Código (LPCXpresso)

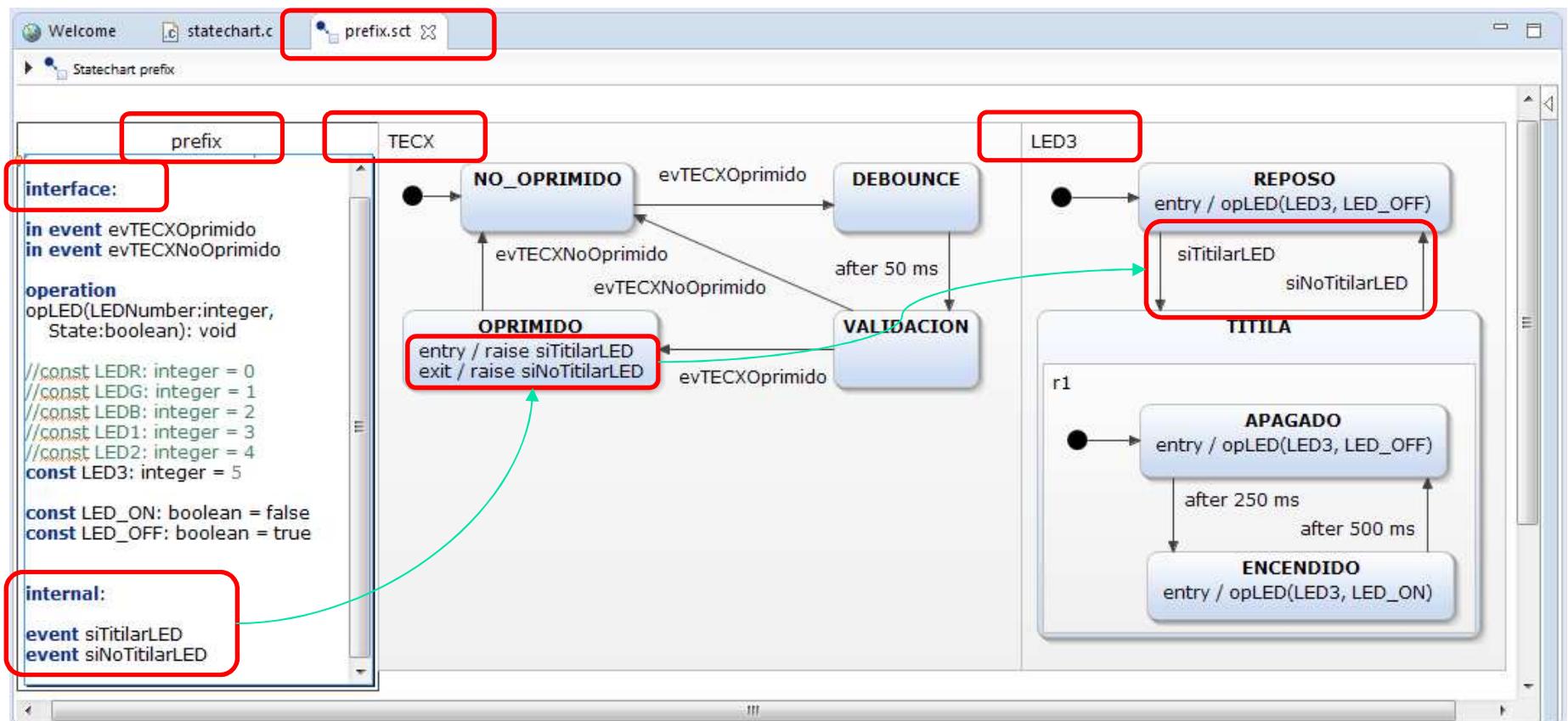


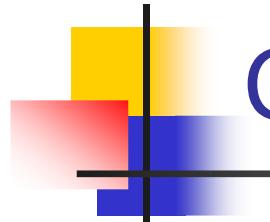


Generación de Código (LPCXpresso)

```
uint32_t BUTTON_Status;  
.....  
  
BUTTON_Status = Buttons_GetStatus();  
if (BUTTON_Status != 0)           // Event -> evTECXOprimido => OK  
    prefixIface_raise_evTECXOprimido(&statechart);  
else                            // Event -> evTECXNoOprimido => OK  
    prefixIface_raise_evTECXNoOprimido(&statechart);  
  
prefix_runCycle(&statechart);      // Run Cycle of Statechart  
}  
}  
}
```

Generación de Código (LPCXpresso)

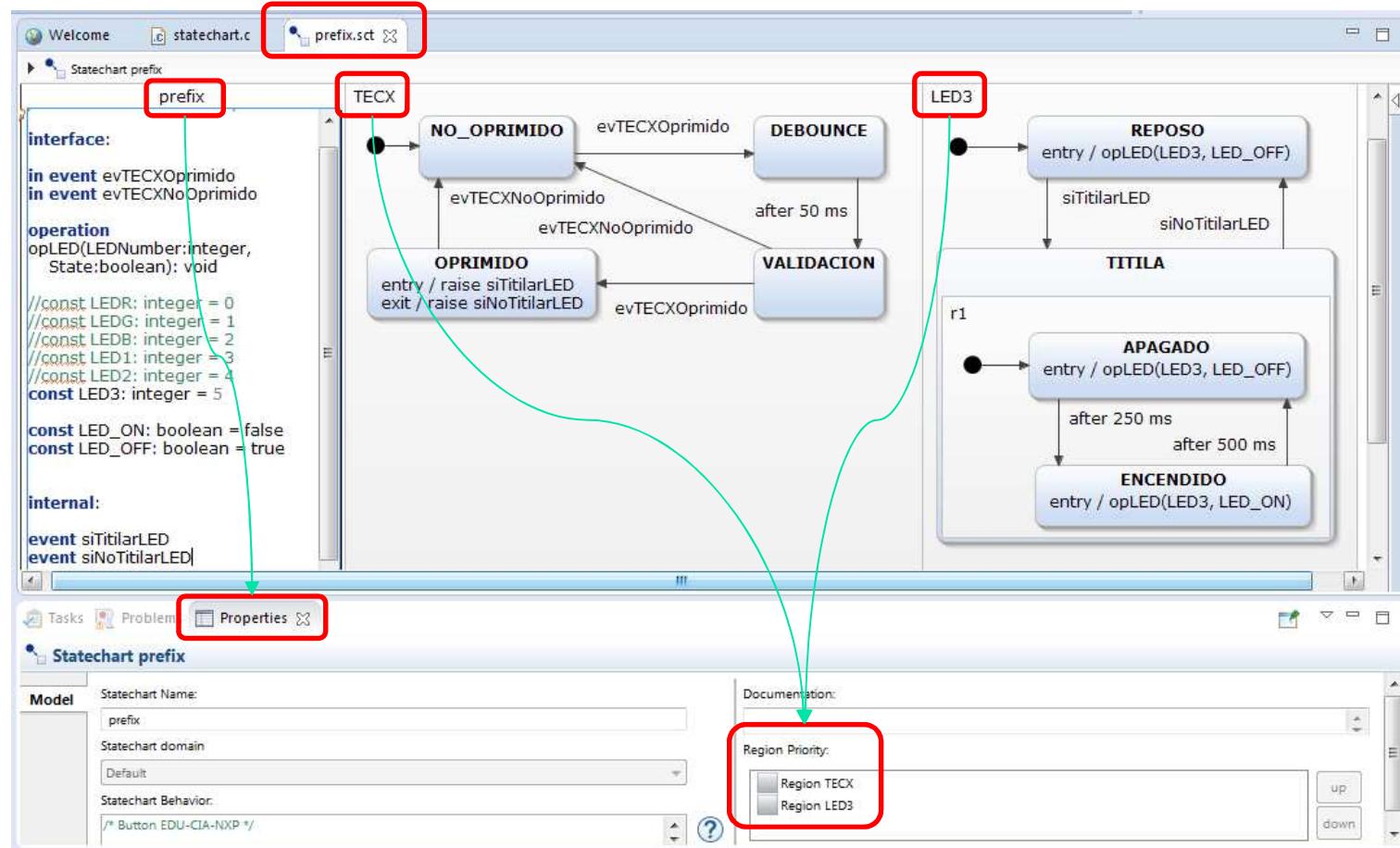




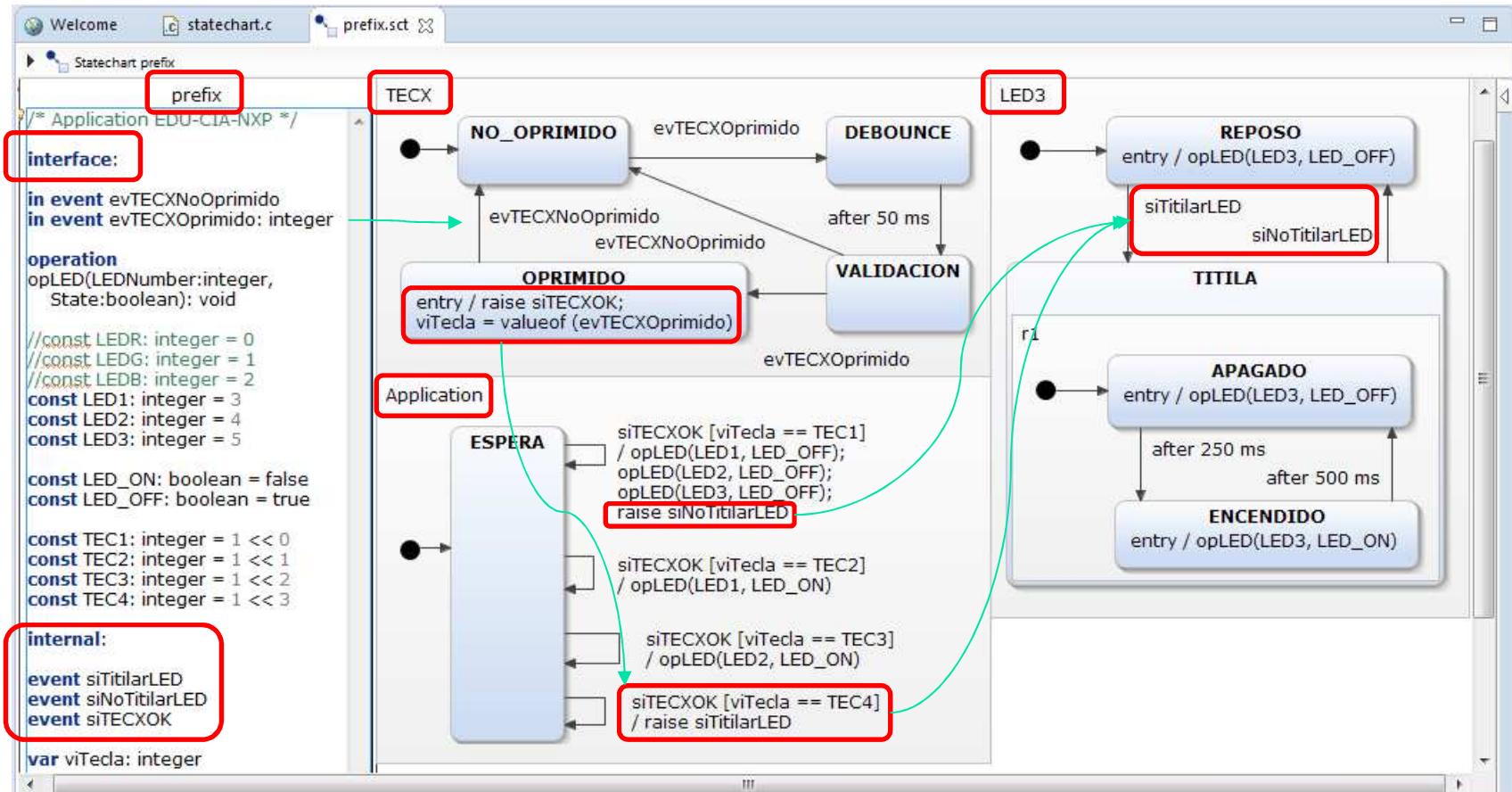
Generación de Código (LPCXpresso)

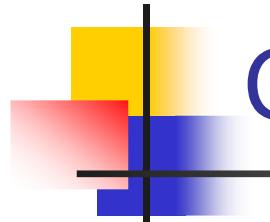
- ¿Porqué conviene diferenciar eventos de señales?
- ¿Cambia main() respecto al de Idle Blink LED3?
- Asigne prioridades a las regiones según el siguiente criterio:
 - **Mayor** prioridad a regiones que gestionan entradas
 - **Intermedia** prioridad a regiones que gestionan tareas
 - **Menor** prioridad a regiones que gestionan salidas
 - **¿Porqué?**
- ¿Qué criterio adoptaría para asignar prioridades a las transiciones salientes de un estado?, ¿porqué?

Generación de Código (LPCXpresso)



Generación de Código (LPCXpresso)

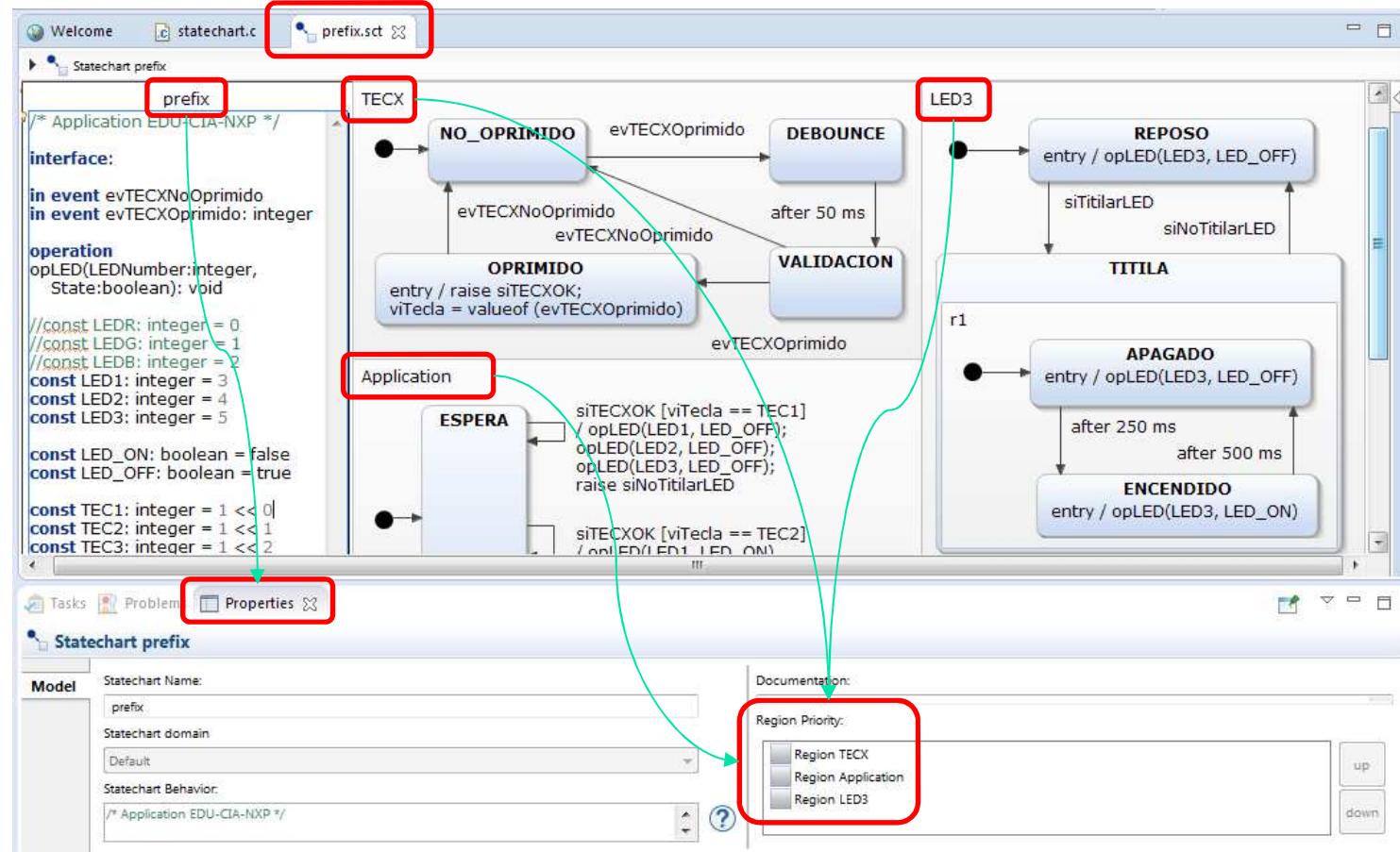




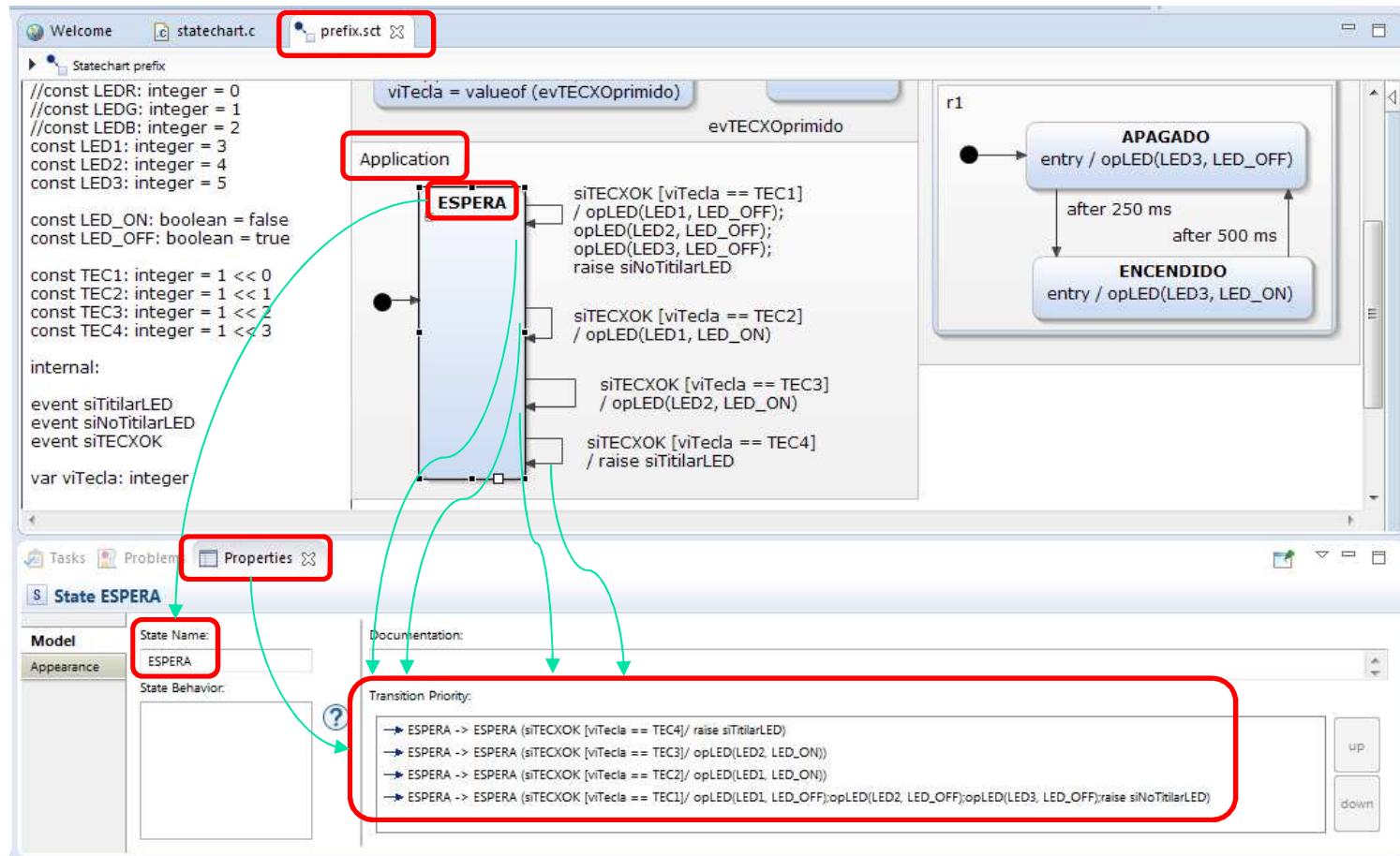
Generación de Código (LPCXpresso)

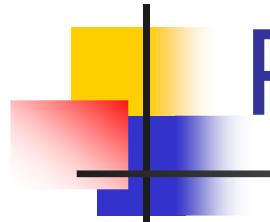
```
uint32_t BUTTON_Status;  
.....  
  
BUTTON_Status = Buttons_GetStatus();  
if (BUTTON_Status != 0)           // Event -> evTECXOprimido => OK  
    prefixIface_raise_evTECXOprimido(&statechart, BUTTON_Status);  
                                //Value:Tecla  
else                          // Event -> evTECXNoOprimido => OK  
    prefixIface_raise_evTECXNoOprimido(&statechart);  
  
    prefix_runCycle(&statechart);      // Run Cycle of Statechart  
}  
}  
}
```

Generación de Código (LPCXpresso)



Generación de Código (LPCXpresso)





Referencias

- <https://www.itemis.com/en/yakindu/statechart-tools/>
- <https://www.itemis.com/en/yakindu/statechart-tools/documentation/user-guide/>