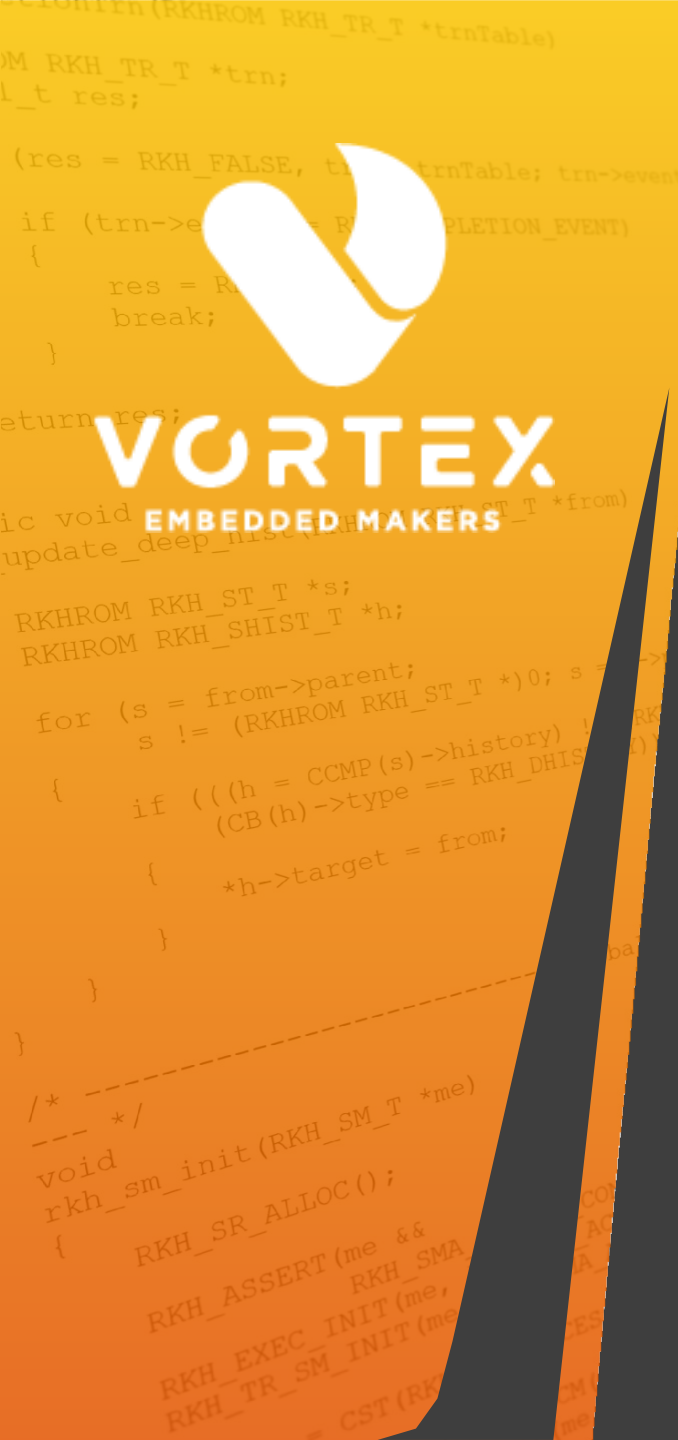


SASE 2017
SIMPOSIO ARGENTINO DE SISTEMAS EMBEBIDOS

RKH
STATE MACHINE FRAMEWORK

Workshop: Programando la CIAA con Statecharts usando RKH

Ing. Darío Baliña (dariosb@vortexmakes.com)

10 de Agosto de 2017

Agenda

- Introducción RKH
- Desarrollo de aplicaciones RKH
- Aplicaciones RKH sobre la CIAA
- Entrenamiento Hands-On

https://github.com/dariosb/EduCIAA-RKH-demos/releases/tag/sase2017

RKH™

- RKH es una infraestructura de software reutilizable para la ejecución simultánea de Statecharts dentro del dominio de los embedded systems de tiempo-real, basado en el modelo computacional Objeto Activo.

RKH
STATE MACHINE FRAMEWORK

AGENCIA
NACIONAL DE PROMOCION
CIENTIFICA Y TECNOLOGICA

Reconocido y apoyado por la Agencia Nacional de Promoción Científica y Tecnológica (NA013/12)

Licencia GNU GPLV3
Download: http://sourceforge.net/projects/rkh-reactivesys/
Reference Manual: http://rkh-reactivesys.sourceforge.net/
Repositorio GIT: https://git.code.sf.net/p/rkh-reactivesys/code

source forge
git

Problemática actual

- **Complejidad**.
  - Recursos limitados
  - Interacción con hardware
  - Tiempo-real
  - Herramientas de depuración
- **Reducir tiempos y costo.**

La industria del embedded software requiere de herramientas, procesos y métodos que mitiguen estos factores para mantener la competitividad de las empresas.

Objetivos tecnológicos

- Simple, genérico, flexible, modular, multiplataforma, configurable, eficiente en términos de consumo de recursos, de tiempo-real y compatible con lenguajes C/C++.

- Aplicable a limitadas plataformas de 8bits baremetal, como también a plataformas de 32bits corriendo RTOS multitarea preemptives, si así se requiere.

- Promueva la adopción de un lenguaje y técnicas comunes entre los desarrolladores, generando un nivel de abstracción tal que el diseño resulte más claro y fácil de modificar, manteniendo oculto los detalles de la implementación.

RKH: ¿Qué es? ¿Qué nos dá?

- **RKH** es un **Framework**, para el desarrollo de aplicaciones utilizando técnicas de desarrollo de software, como:
  - UML 2.0
  - Statecharts.
  - Paradigma de la programación dirigida por eventos.
  - modelo **Objeto Activo.**
- Provee los servicios necesarios para el desarrollo completo del sistema de software:
  - ejecución simultánea de Statecharts
  - manejo de eventos
  - temporizadores
  - gestor de memoria dinámica
  - trazador para validación y depuración durante ejecución.

¿Por qué un framework?

Framework

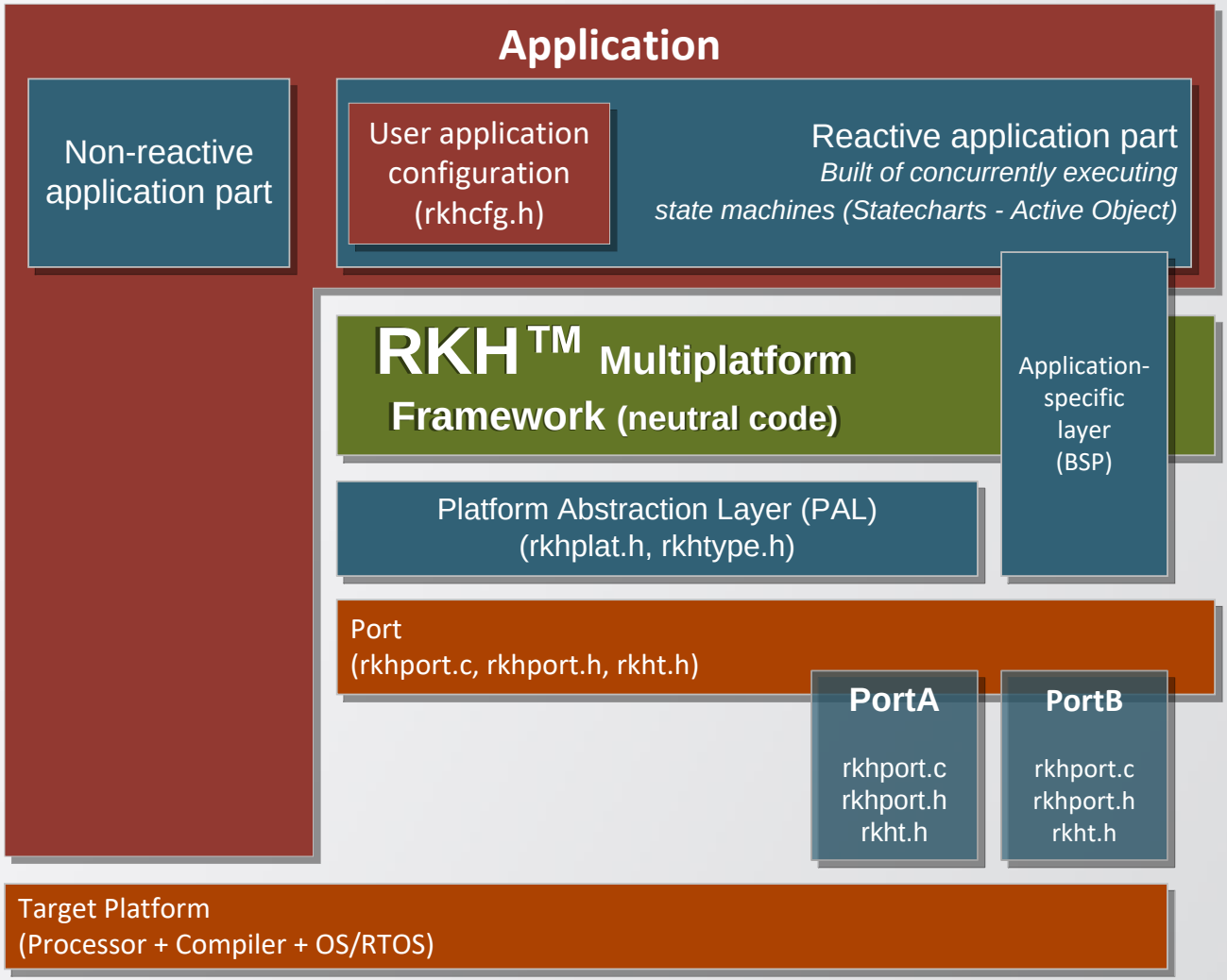
Es una abstracción que permite que una funcionalidad genérica pueda reutilizarse, especializarse o reemplazarse selectivamente mediante código adicional escrito por el usuario.

No es una biblioteca de funciones, se basa en el principio de **Inversión de Control** (IoC). El flujo de ejecución es controlado por el framework siguiendo las directivas definidas por la aplicación.

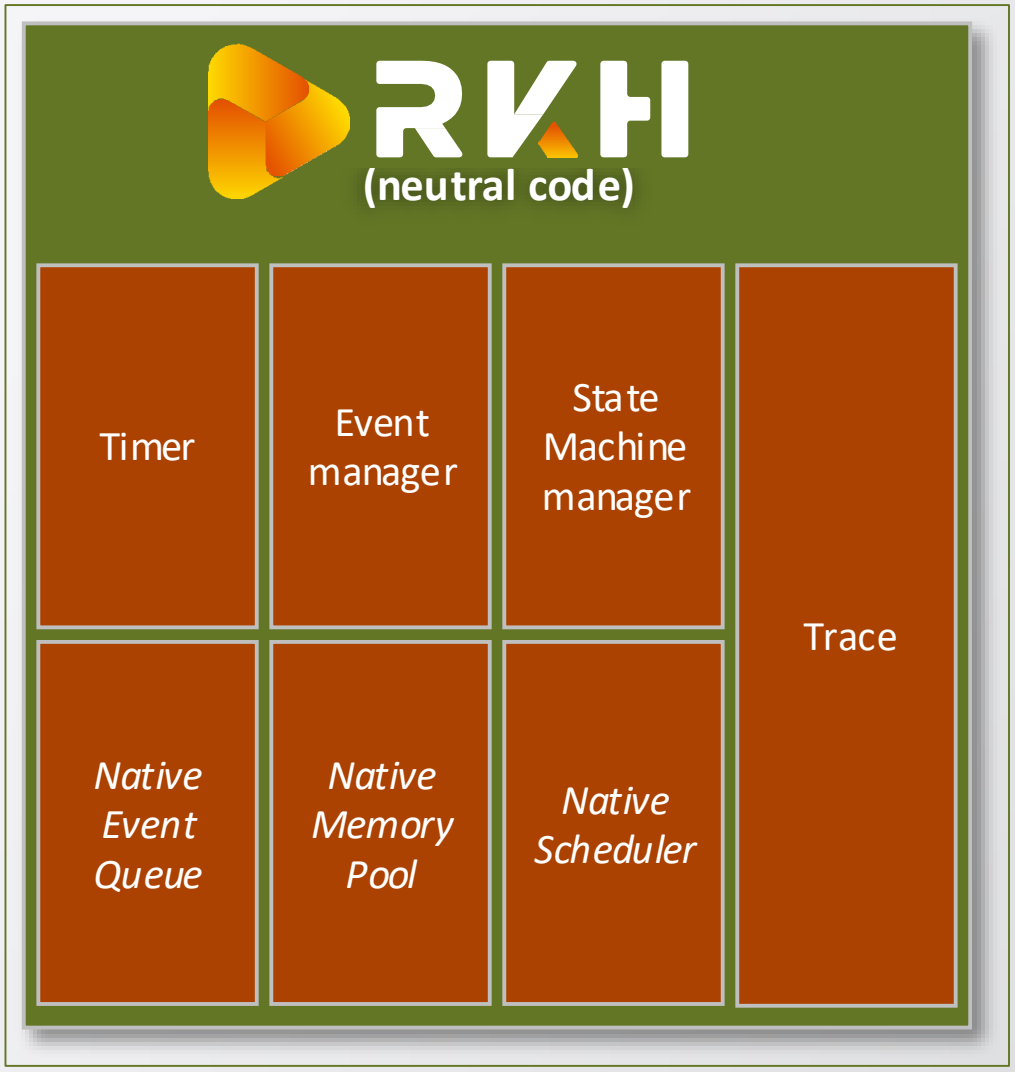
Framework: beneficios

- Brinda la infraestructura del sistema, permitiendo al usuario concentrarse solo en el requerimientos de la aplicación.
- Favorece la reutilización de código que ya ha sido construido, probado y usado por otros programadores, incrementando la confiabilidad y reduciendo los tiempos de codificación y validación.
- Focaliza los grupos de desarrollo en conseguir los requerimientos funcionales específicos del producto.
- Estandariza la codificación y la documentación, generando un **lenguaje común** entre los desarrolladores.

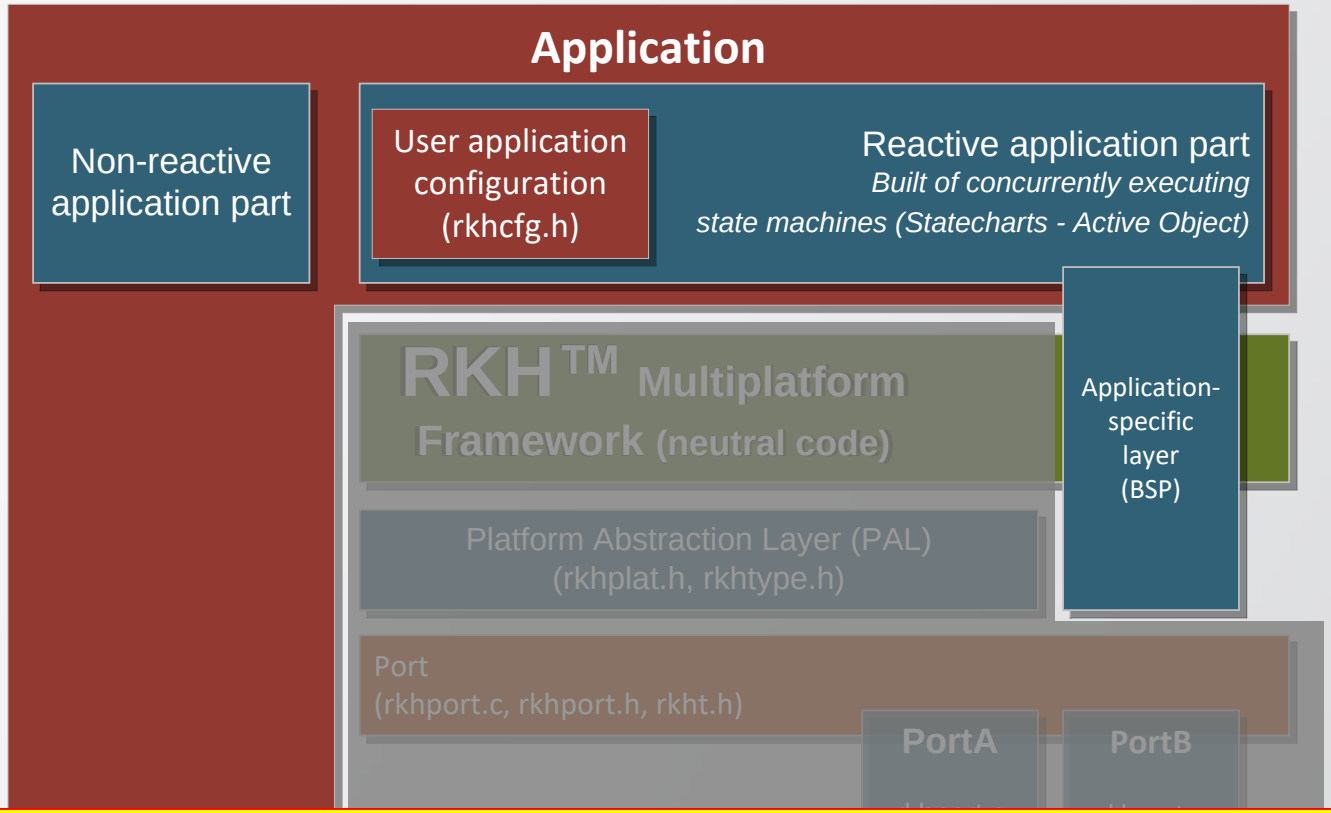
RKH: Estructura



RKH: Servicios nativos



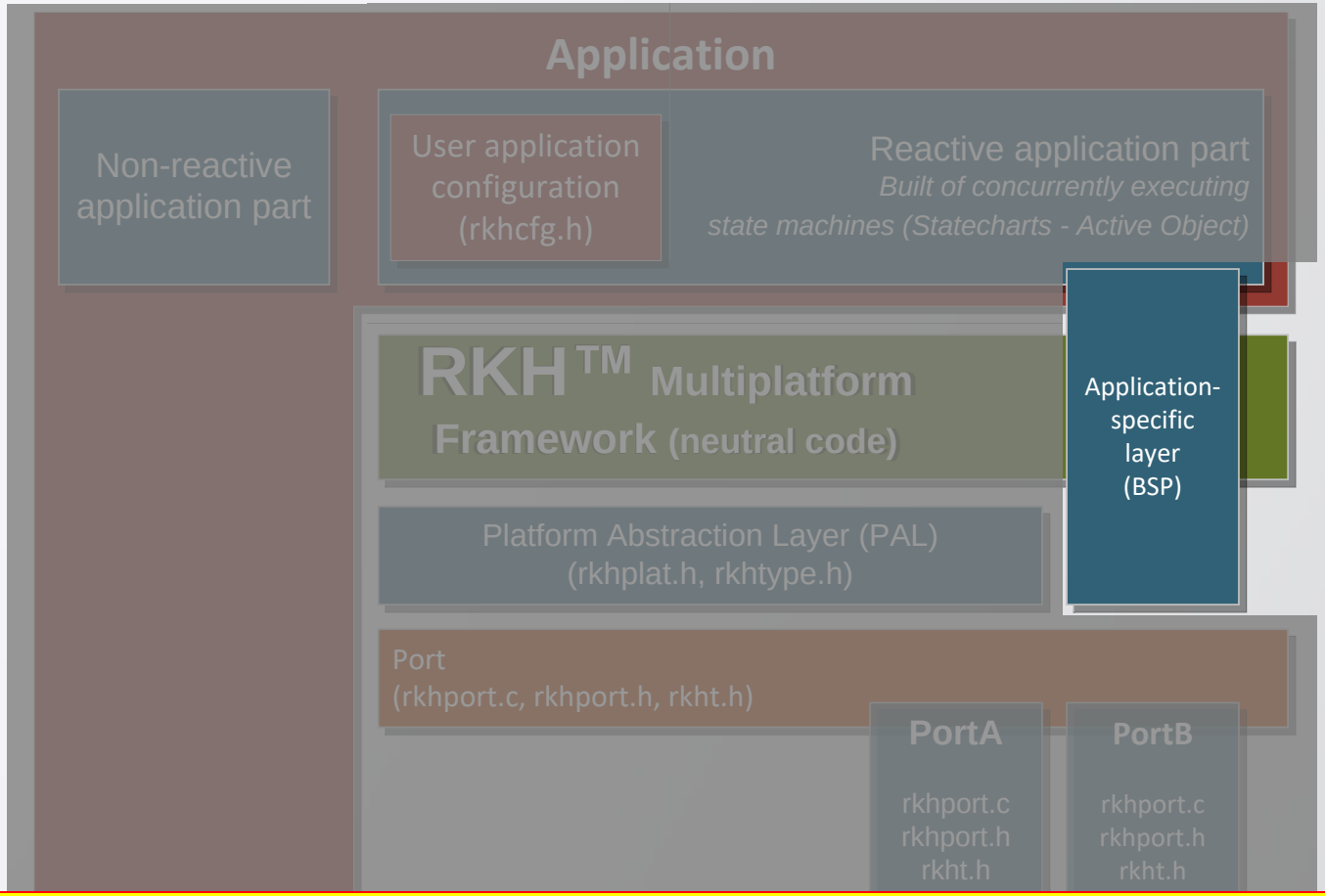
RKH: Aplicación de usuario



Aplicación reactiva

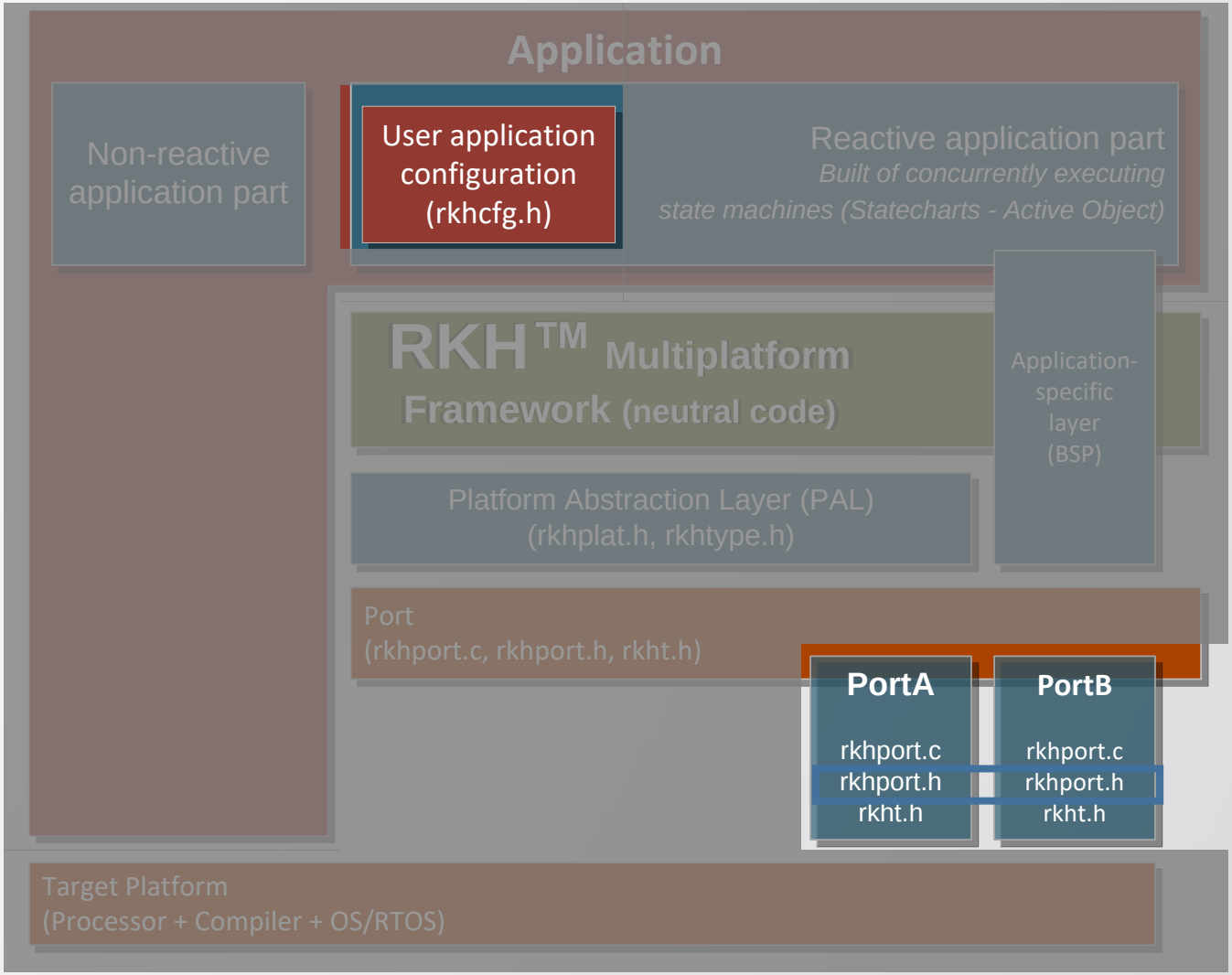
Se constituye por uno o más Statecharts que se ejecutan en forma simultánea en contexto de sus AO, que colaboran entre sí y con su entorno, enviando y recibiendo mensajes asincrónicos.

RKH: BSP (Board Support Package)



BSP
Concentra las dependencias contra la plataforma, que no están definidas por el port de RKH.

RKH: Configuración



RKH: Configuración

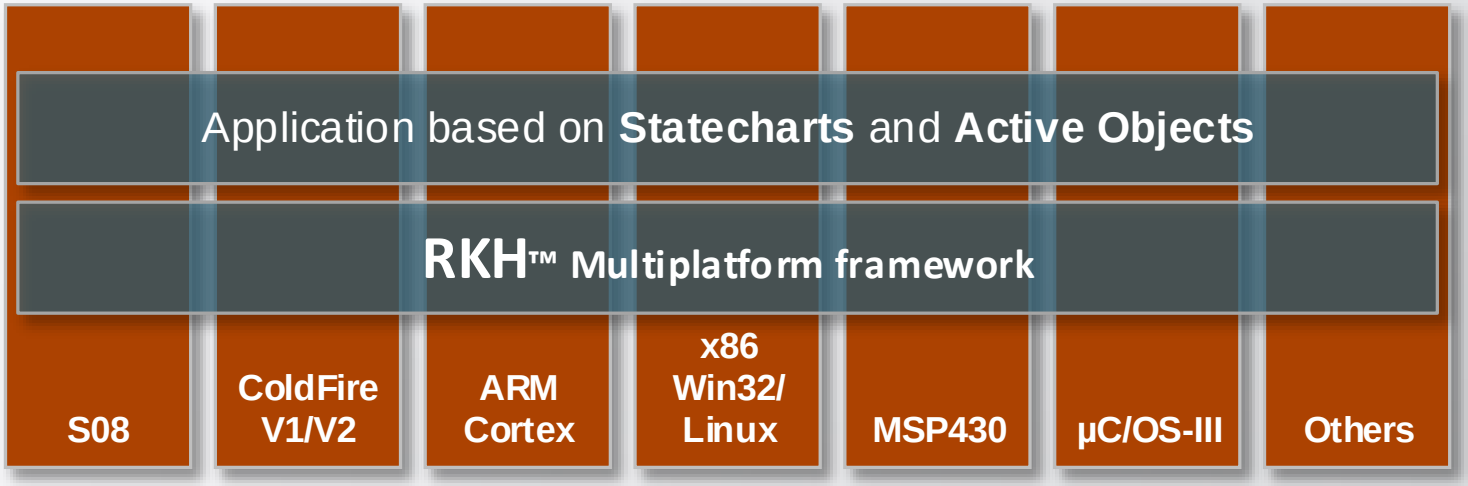
- Consta de más de 100 configuraciones.
- (**rkhcfg.h**) necesidades de la aplicación.
- (**rkhport.h**) ajustes dependientes de la plataforma.
- Validación en tiempo de compilación.
- Permite reducir el consumo de recursos ROM/RAM y aumentar la eficiencia de operación.

Footprints de referencia:

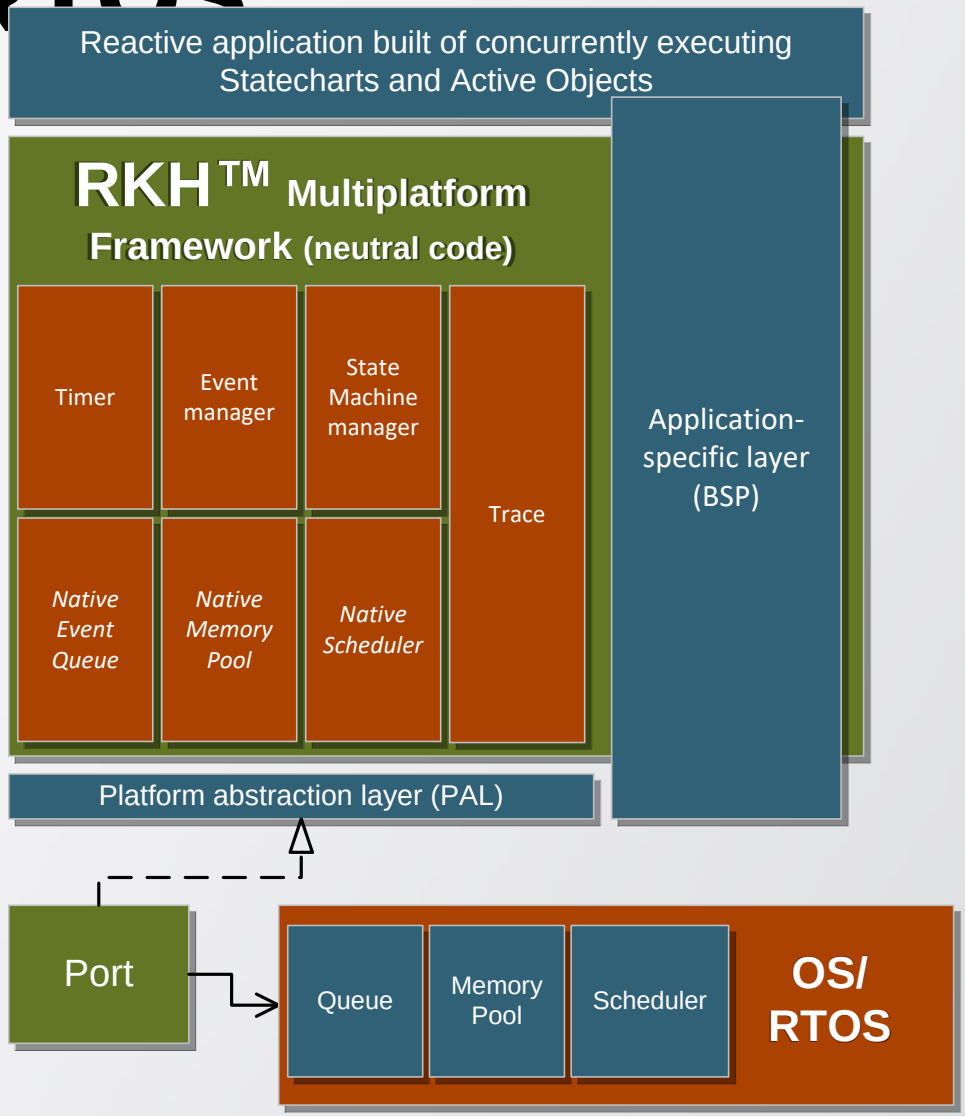
- Freescale S08 (8-bits) - **1KB** ~ 4.8KB
- Coldfire V1 (32-bits) - **1.3KB** ~ 6.3KB
- Cortex M4 (32-bits) - **1.5KB** ~ 6.9KB

RKH: Multiplataforma

No sólo implica ser independiente del procesador y compilador sino también de su entorno de ejecución, ya que puede acoplarse y trabajar en conjunto con cualquier RTOS/OS tradicional, heredando sus **capacidades y servicios**.

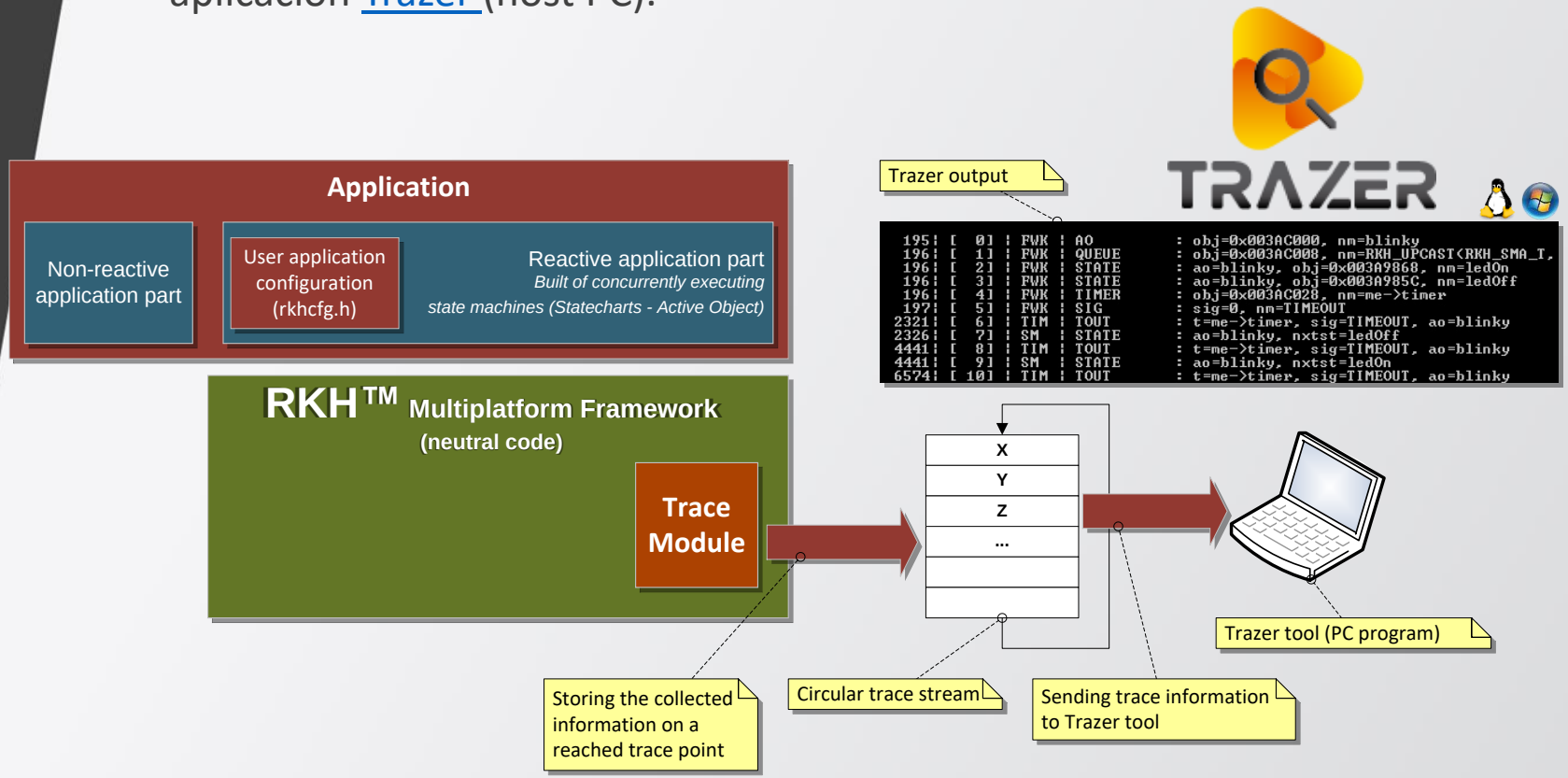


RKH: Colaboración OS/RTOS



Verificación y validación

El comportamiento de toda aplicación basada en RKH puede verificarse y validarse, con un alto grado de detalle, en tiempo de ejecución mediante la aplicación Trazer (host PC).

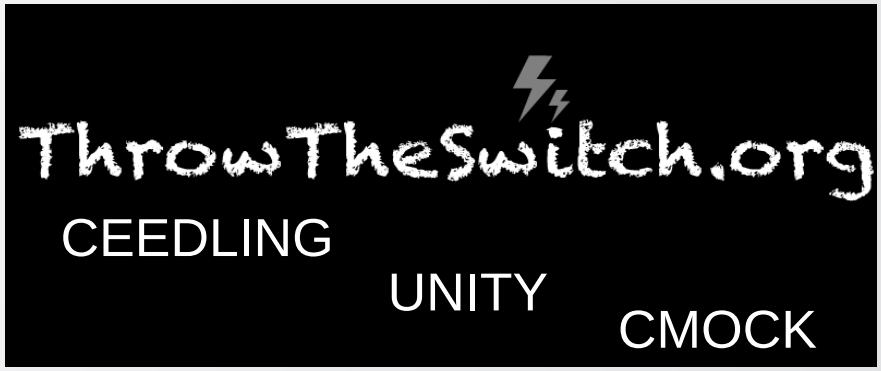


Trazer

- Decodifica el trace stream y lo muestra en formato legible.
- Medios de captura:
  - serial port.
  - TCP socket.
  - binary stream file.
- Interprete de símbolos, eventos y configuración.
- Medición de tiempos de respuesta.

TDD Harness

Desarrollo evolutivo e incremental dirigido por pruebas



https://github.com/dariosb/InyectorControl

Otras características

- Instanciar Máquinas de Estados sin AO asociado.
- Asociar comportamiento en Transición Inicial.
- AO polimórficos, Vtable in C. (activate, task, post_fifo, post_lifo)
- Múltiple Transiciones habilitadas desde el mismo disparador.
- Runtime constructors para AO y SM, múltiples o simple instancias.
- Soporte de Uncrustify (embellecedor código), reglas de RKH.
- Templates para fuentes y encabezados.
- **Ports**:
  - OS/RTOS: CIAA-OSEK, uCOS III, Linux, Windows.
  - BareMetal: S08, Coldfire V1/V2, Kinetis: K6x K6xF KL2x, LPC176x, LPC433x.

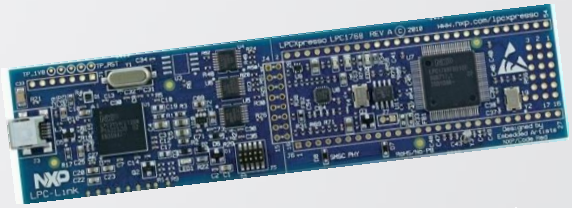
Licenciamiento

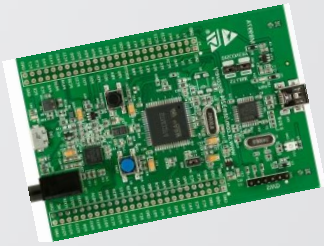
- RKH es un software licenciado bajo los términos de **GNU GPLv3**.

- Esta licencia puede utilizarse tanto para investigación y desarrollo como para propósitos académicos.

- git://git.code.sf.net/p/rkh-reactivesys/code

Desarrollo de aplicaciones RKH





BARE METAL

RKH

Linux

µC/OS-III®
The Real-Time Kernel

freeRTOS

Microsoft® Windows®

Desarrollo de aplicaciones RKH

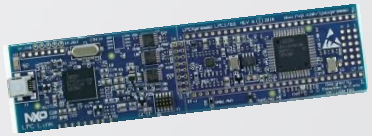
www.vortexmakes.com

Desarrollo de aplicaciones RKH

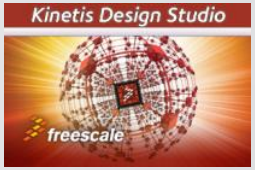
cross-platform

- EduCIAA, CIAA-IDE.

- LPCXpresso.



- FRDMK64F, KDS



- Linux, GNU.

Windows, Visual Studio.

Aplicaciones RKH sobre la CIAA

Aplicaciones RKH sobre la CIAA

¿ Que hay que hacer?

1. Preparar entorno de desarrollo.
2. Configurar RKH. **(rkhcfg.h)**
3. Trasladar modelo a código **(app.c/h)**
    1. Declarar AO, señales y acciones.
    2. Crear AO y estructura de statechart.
    3. Implementar acciones.
4. Iniciar plataforma y poner en ejecución. **(main.c)**
5. Codificar BSP
6. Validar.

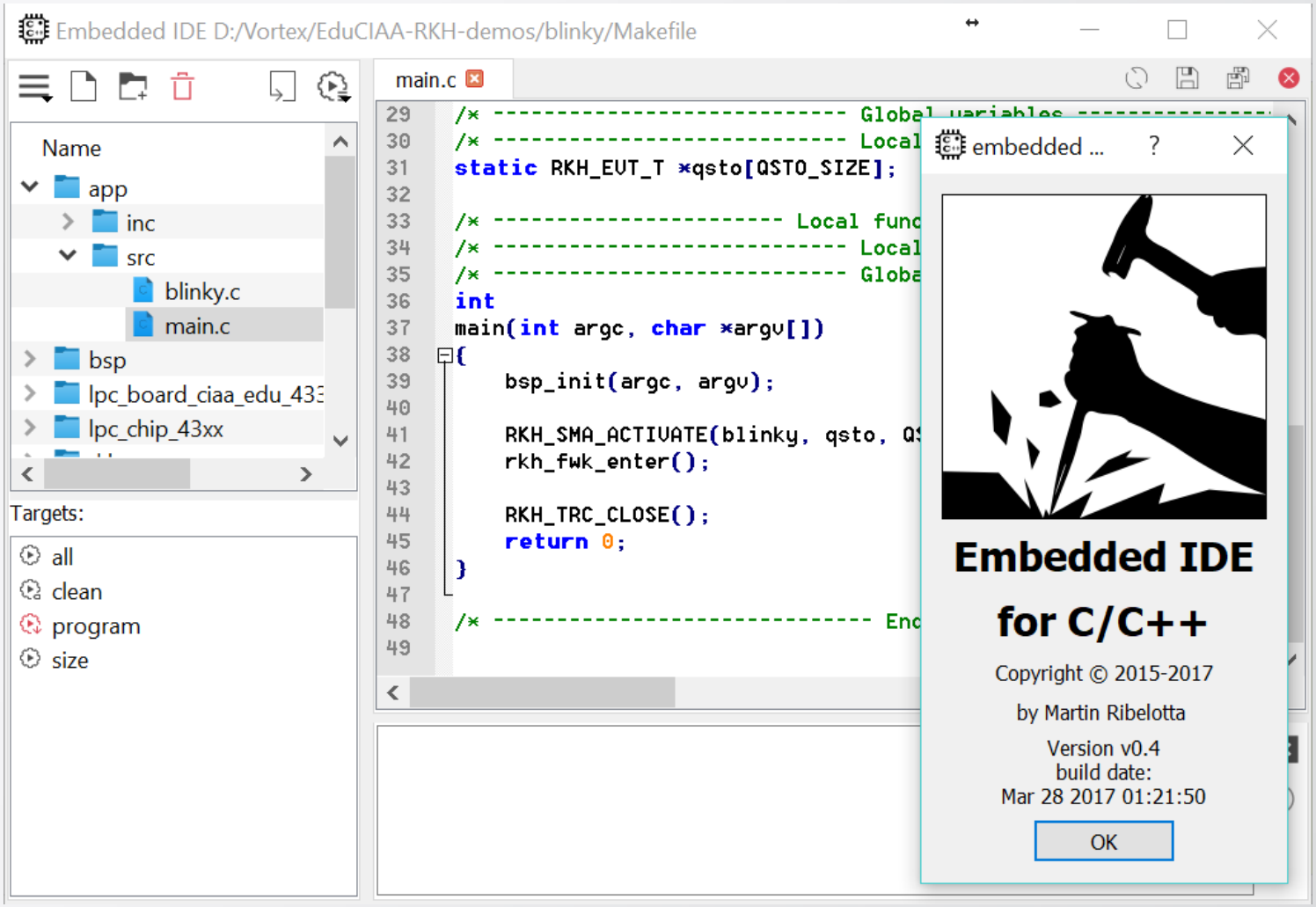
Aplicaciones RKH sobre la CIAA

1

Preparar entorno de desarrollo

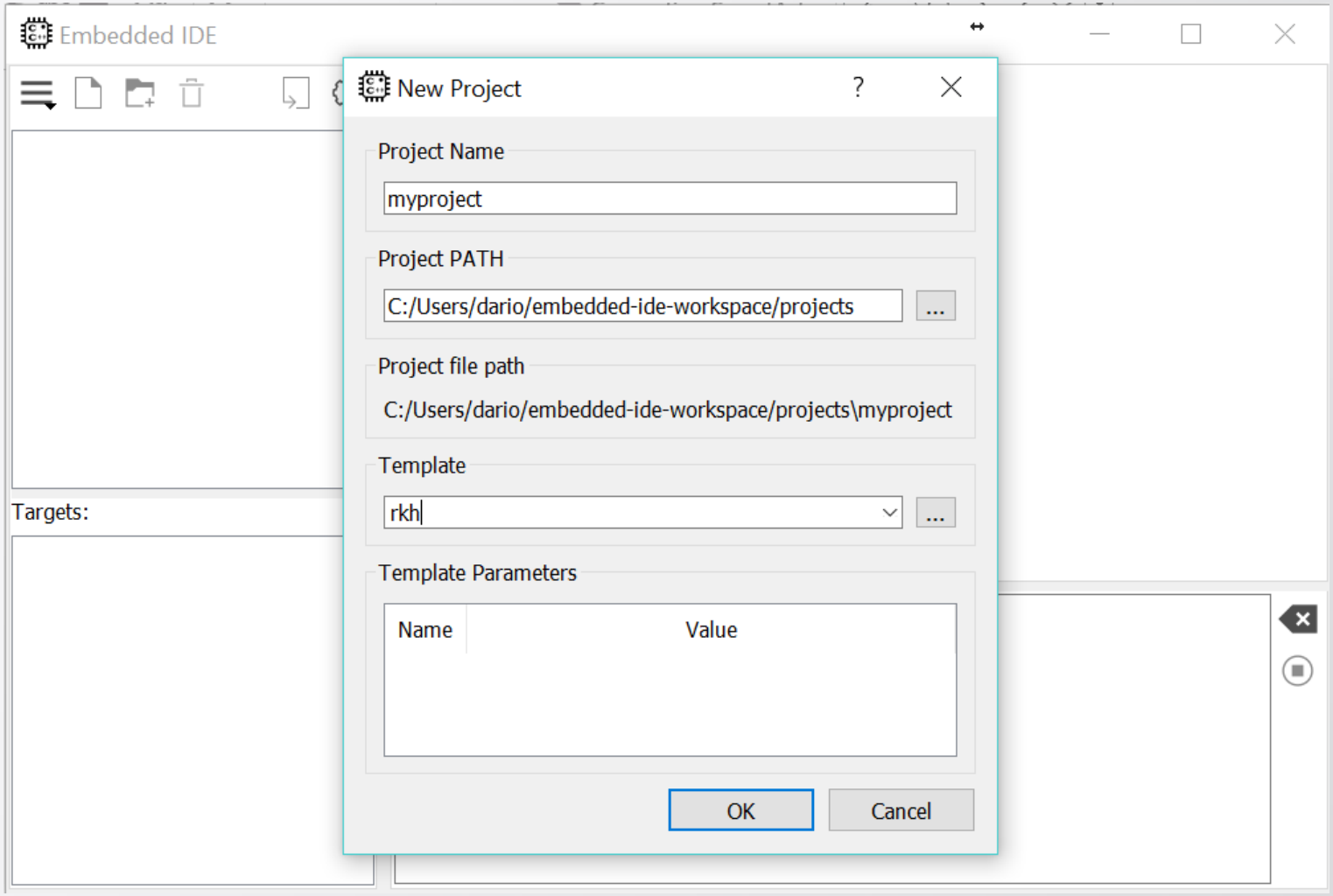
1. Preparar entorno de desarrollo

CIAA Embedded IDE



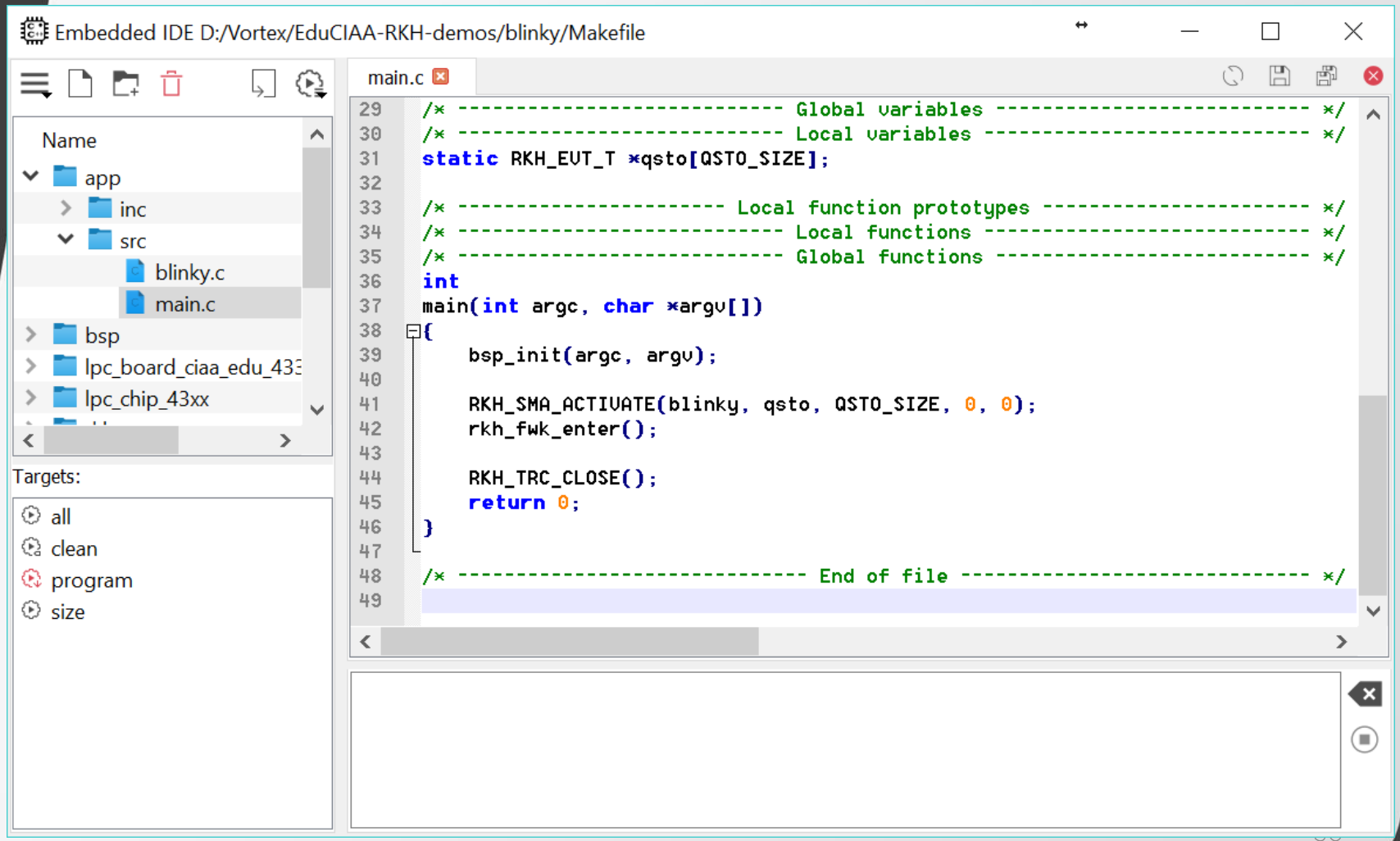
1. Preparar entorno de desarrollo

Crear nuevo proyecto y seleccionar rkh.template

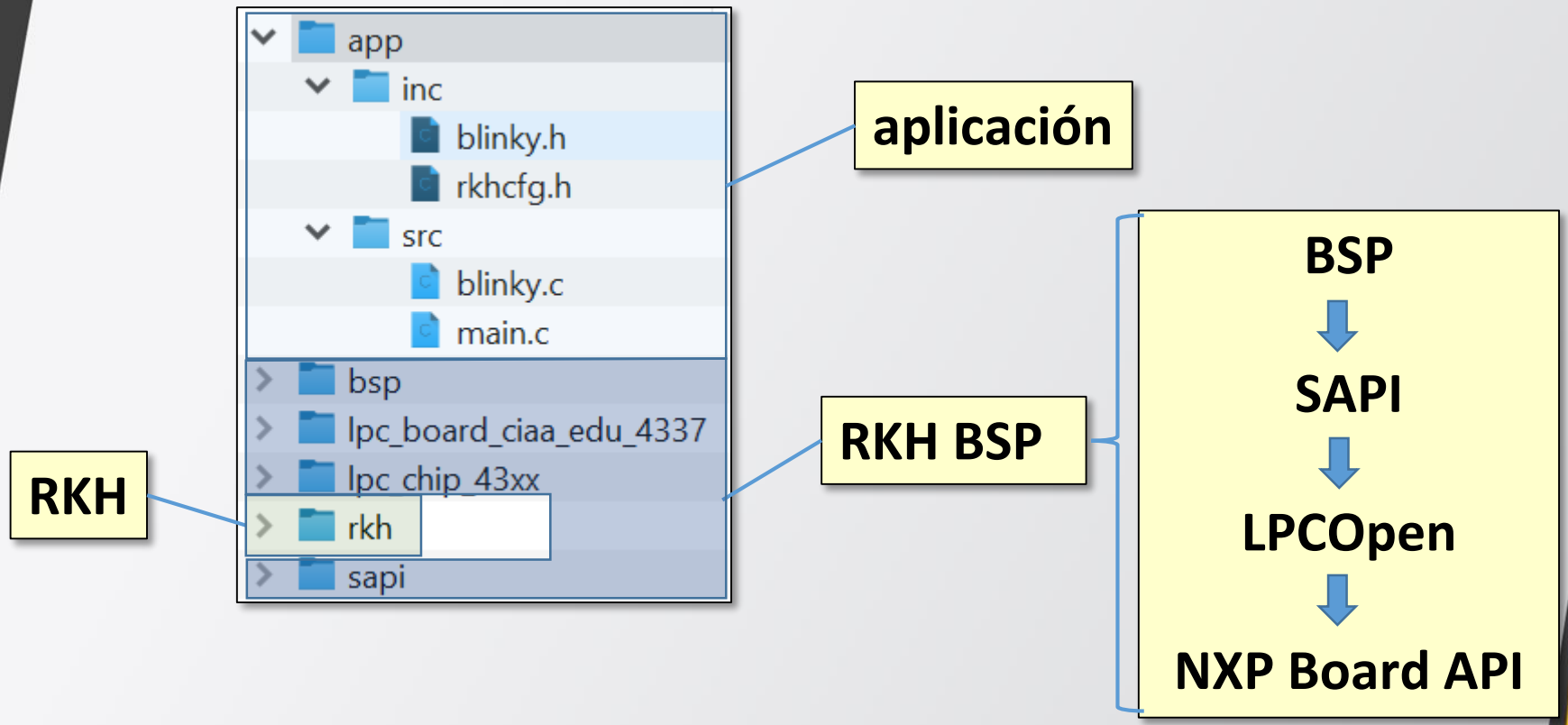


1. Preparar entorno de desarrollo

Aplicación demo RKH blinky



1. Preparar entorno de desarrollo



Aplicaciones RKH sobre la CIAA

¿ Que hay que hacer?

1. ✓ Preparar entorno de desarrollo.
2. Configurar RKH. **(rkhcfg.h)**
3. Trasladar modelo a código **(app.c/h)**
    1. Declarar AO, señales y acciones.
    2. Crear AO y estructura de statechart.
    3. Implementar acciones.
4. Iniciar plataforma y poner en ejecución. **(main.c)**
5. Codificar BSP
6. Validar.

2

Configurar RKH (rkhcfg.h, rkhport)

2. Configurar RKH (rkhcfg.h)

- **Framework.**
- **Statecharts Application.**
- **Trace.**
- **Queue.**
- **Fixed-sized memory block.**
- **Software timers.**

- *Toda aplicación RKH debe tener su propio archivo de configuración.*
- *Adapta y configura RKH por opciones de compilación y macros.*
- *Permite reducir el consumo de ROM y RAM.*
- *Optimiza la performance del sistema en una manera substancial.*
- *Mas de 100 opciones de configuración.*
- *http://rkh-reactivesys.sourceforge.net/cfg.html*

2. Configurar RKH (rkhcfg.h)

Related with Framework (FWK)				
Option	*Type*	*Range*	*Default*	*Description*
RKH_CFG_FWK_MAX_SMA	integer	[1..64]	4	Specify the maximum number of state machine applications (SMA) to be used by the application (can be a number in the range [1..64]).
RKH_CFG_FWK_DYN_EVT_EN	boolean		RKH_DISABLED	If the dynamic event support (see RKH_CFG_FWK_DYN_EVT_EN) is set to 1, RKH allows to use event with parameters, defer/recall, allocating and recycling dynamic events, among other features.
RKH_CFG_FWK_MAX_EVT_POOL	integer	[0..255]	RKH_DISABLED	If the dynamic event support is enabled (see RKH_CFG_FWK_DYN_EVT_EN) then the RKH_CFG_FWK_MAX_EVT_POOL can be used to specify the maximum number of fixed-size memory block pools to be used by the application (can be a number in the range [0..256]). Note that a value of 0 will completely suppress the memory pool services.
RKH_CFG_FWK_SIZEOF_EVT	integer	[8,16,32]	8	Specify the size of the event signal. The valid values [in bits] are 8, 16 or 32. Default is 8. The higher the signal size, the higher the event structure size and therefore more memory consumption. See RKH_SIG_T data type.
RKH_CFG_FWK_SIZEOF_EVT_SIZE	integer	[8,16,32]	8	Specify the data type of event size. The valid values [in bits] are 8, 16 or 32. Default is 8. See RKH_ES_T, rkh_fwk_epool_register(), and RKH_ALLOC_EVT(). Use a 8 value if the bigger event size is minor to 256 bytes.
RKH_CFG_FWK_DEFER_EVT_EN	boolean		RKH_DISABLED	If the RKH_CFG_FWK_DEFER_EVT_EN is set to 1 and the dynamic event support is enabled (see RKH_CFG_FWK_DYN_EVT_EN), RKH enables the defer and recall event features.
RKH_CFG_FWK_ASSERT_EN	boolean		RKH_ENABLED	If the RKH_CFG_FWK_ASSERT_EN is set to 0 the checking assertions are disabled. In particular macros RKH_ASSERT(), RKH_REQUIRE(), RKH_ENSURE(), RKH_INVARIANT(), and RKH_ERROR() do NOT evaluate the test condition passed as the argument to these macros. One notable exception is the macro RKH_ALLEGE(), that still evaluates the test condition, but does not report assertion failures when the RKH_CFG_FWK_ASSERT_EN is enabled.
RKH_CFG_HOOK_DISPATCH_EN	boolean		RKH_DISABLED	If the RKH_CFG_HOOK_DISPATCH_EN is set to 1, RKH will invoke the dispatch hook function rkh_hook_dispatch() when dispatching an event to a SMA. When this is set the application must provide the hook function.

2. Configurar RKH (rkhport.h)

Seleccionar el port de acuerdo a:

- plataforma
- compilador
- OS.

rkhplat.h / rkhtype.h

```
#endif

#ifdef __CFV1CW63__
    #include "..\..\portable\cfv1\rkhs\cw6_3\rkhport.h"
#endif

#ifdef __ARM_CM0CW10__
    #include "../../portable/arm-cortex/rkhs/arm_cm0/cw_v10/rk
#endif

#ifdef __ARM_CM4FCW10__
    #include "../../portable/arm-cortex/rkhs/arm_cm4f/cw_v10/r
#endif

#ifdef __KSDK_KDS__
    #include "..\..\portable\arm-cortex\rkhs\ksdk\kds\rkhport.
#endif

#ifdef __KSDK_OS_KDS__
    #include "..\..\portable\arm-cortex\ksdk_os\ucosiii\kds\rk
#endif

#ifdef __UCOS_V3_03_01__
    #include "..\..\portable\ucos\v3.03.01\rkhport.h"
#endif
```

Con CIAA-Embedded IDE usando templates, NO es necesario realizar la configuración del port.

Aplicaciones RKH sobre la CIAA

¿ Que hay que hacer?

1. ✓ Preparar entorno de desarrollo.
2. ✓ Configurar RKH. **(rkhcfg.h)**
3. Trasladar modelo a código **(app.c/h)**
    1. Declarar AO, señales y acciones.
    2. Crear AO y estructura de statechart.
    3. Implementar acciones.
4. Iniciar plataforma y poner en ejecución. **(main.c)**
5. Codificar BSP
6. Validar.

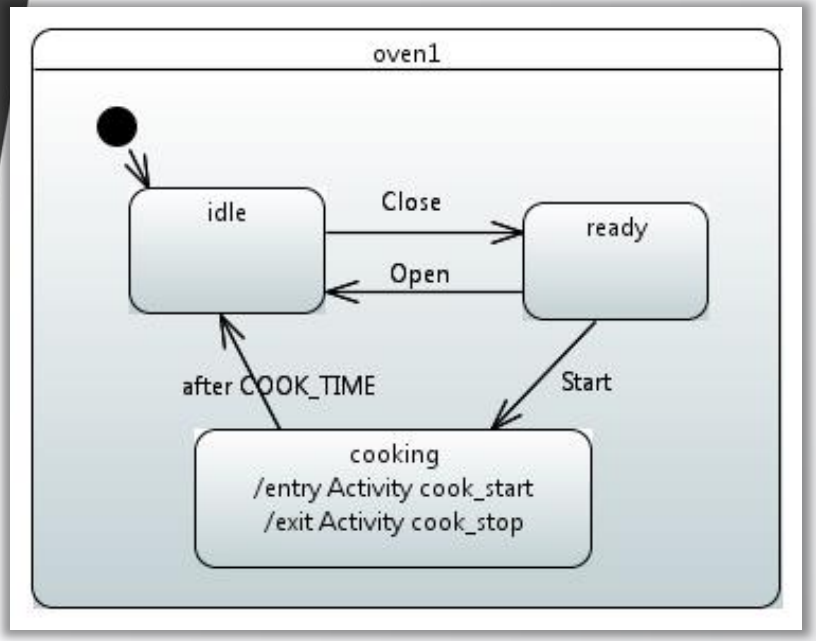
Aplicaciones RKH sobre la CIAA

3

Trasladar el modelo a código (myapp.c/h)

3. Trasladar el modelo a código

1. Declarar AO, señales y acciones *(oven.h)*




```
/* ---------------------------------- Macros -------
#define COOK_TIME       RKH_TIME_SEC(5)     /* Cook

/* --------------------------------- Constants -----
/* ================================== Signals =======
typedef enum ov_sigs_t
{
    OPEN,   /* door is open  */
    CLOSE,  /* door is close */
    START,  /* start button pressed */
    TMREVT, /* timer expired */
    TERM,   /* to close application in x86 */
} OV_SIGS_T;

/* ======================== Declares active object =
RKH_SMA_DCLR(oven);

/* ================== Declares states and pseudosta
RKH_DCLR_BASIC_STATE idle, ready, cooking;

/* ------------------------------ Data types -----
typedef struct Oven Oven;

/* -------------------------- External variables ---
/* -------------------------- Function prototypes --
/* ============================ Initial action =====
void oven_init(Oven *const me);

/* ============================ Effect actions =====
/* ============================ Entry actions =====
void cook_start(Oven *const me, RKH_EVT_T *pe);

/* ============================ Exit actions ======
void cook_stop(Oven *const me, RKH_EVT_T *pe);
```

3. Trasladar el modelo a código

2. Crear AO y estructura de statechart *(oven.c)*

#define **RKH_SMA_CREATE**(type, name, prio, ppty, initialState, initialAction, initialEvt)

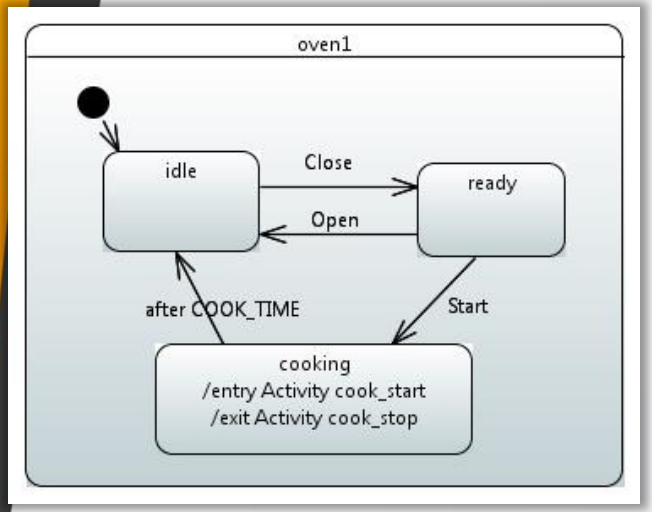
Declare and allocate a SMA (active object) derived from **RKH_SMA_T**. Also, initializes and assigns a state machine to previously declared SMA.

Parameters

[in]	**type**	Data type of the SMA. Could be derived from **RKH_SMA_T**.
[in]	**name**	Name of state machine application. Also, it represents the top state of state diagram.
[in]	**prio**	State machine application priority. A unique priority number must be assigned to each SMA from 0 to RKH_LOWEST_PRIO. The lower the number, the higher the priority.
[in]	**ppty**	State machine properties. The available properties are enumerated in RKH_HPPTY_T enumeration in the **rkh.h** file.
[in]	**initialState**	Pointer to initial state. This state could be defined either composite or basic (not pseudo-state).
[in]	**initialAction**	Pointer to initialization action (optional). The function prototype is defined as RKH_INIT_ACT_T. This argument is optional, thus it could be declared as NULL.
[in]	**initialEvt**	Pointer to an event that will be passed to state machine application when it starts. Could be used to pass arguments to the state machine like an argc/argv. This argument is optional, thus it could be declared as NULL or eliminated in compile-time with RKH_CFG_SMA_INIT_EVT_EN = 0.

3. Trasladar el modelo a código

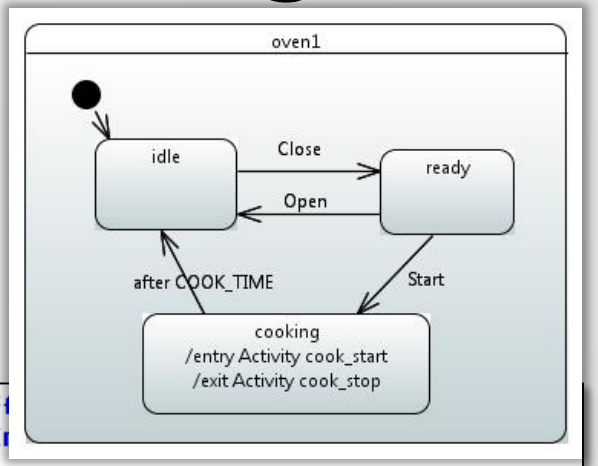
2. Crear AO y estructura de statechart *(oven.c)*




```
/* ------------------------------ Constants -------------------------------
/* ======================== States and pseudostates ========================
RKH_CREATE_BASIC_STATE(idle, NULL, NULL,  RKH_ROOT, NULL);
RKH_CREATE_TRANS_TABLE(idle)
    RKH_TRREG(CLOSE,   NULL,   NULL,   &ready),
RKH_END_TRANS_TABLE

RKH_CREATE_BASIC_STATE(ready, NULL, NULL,  RKH_ROOT, NULL);
RKH_CREATE_TRANS_TABLE(ready)
    RKH_TRREG(OPEN,    NULL,   NULL,   &idle),
    RKH_TRREG(START,   NULL,   NULL,   &cooking),
RKH_END_TRANS_TABLE

RKH_CREATE_BASIC_STATE(cooking, cook_start, cook_stop,  RKH_ROOT, NULL);
    RKH_CREATE_TRANS_TABLE(cooking)
    RKH_TRREG(TMREVT,  NULL,   NULL,   &idle),
RKH_END_TRANS_TABLE

/* --------------------------- Local data types ---------------------------
struct Oven
{
    RKH_SMA_T sma;  /* base structure */
    RKH_TMR_T timer;
};

/* --------------------------- Global variables ---------------------------
/* ============================ Active object ============================
RKH_SMA_CREATE(Oven, oven, 0, HCAL, &idle, oven_init, NULL);
RKH_SMA_DEF_PTR(oven);
```

3. Trasladar el modelo a código

3. Implementar acciones *(oven.c)*




```
/* ---------------------- Local function prototypes ---
/* ---------------------------- Local functions -------
/* ---------------------------- Global functions -------
/* ============================ Initial action =========
void
oven_init(Oven *const me)
{
    bsp_ovenInit();

    /* send objects to trazer */
    RKH_TR_FWK_AO(me);
    RKH_TR_FWK_STATE(me, &idle);
    RKH_TR_FWK_STATE(me, &ready);
    RKH_TR_FWK_STATE(me, &cooking);
    RKH_TR_FWK_OBJ(&me->timer);
    RKH_TR_FWK_FUN(&oven_init);
    RKH_TR_FWK_FUN(&cook_start);
    RKH_TR_FWK_FUN(&cook_stop);

    /* send signals to trazer */
    RKH_TR_FWK_SIG(OPEN);
    RKH_TR_FWK_SIG(CLOSE);
    RKH_TR_FWK_SIG(START);
    RKH_TR_FWK_SIG(TMREVT);
    RKH_TR_FWK_SIG(TERM);

    RKH_TMR_INIT(&me->timer, &e_tout, NULL);
}
```

```
/* ============================ Ef
/* ============================ E
void
cook_start(Oven *const me, RKH_EVT_T *pe)
{
    (void)pe;

    RKH_TMR_ONESHOT(&me->timer, RKH_UPCAST(RKH_SMA_T, me), COOK_TIME);
    bsp_emitterOn();
}

/* ============================ Exit actions ===========================
void
cook_stop(Oven *const me, RKH_EVT_T *pe)
{
    (void)pe;

    rkh_tmr_stop(&me->timer);
    bsp_emitterOff();
}

/* ============================== Guards ===============================
```

Aplicaciones RKH sobre la CIAA

¿ Que hay que hacer?

1. ✓ Preparar entorno de desarrollo.
2. ✓ Configurar RKH. **(rkhcfg.h)**
3. ✓ Trasladar modelo a código **(app.c/h)**
    1. ✓ Declarar AO, señales y acciones.
    2. ✓ Crear AO y estructura de statechart.
    3. ✓ Implementar acciones.
4. Iniciar plataforma y poner en ejecución. **(main.c)**
5. Codificar BSP
6. Validar.

Aplicaciones RKH sobre la CIAA

4

Iniciar plataforma y poner en ejecución. (main.c)

4. Iniciar plataforma y poner en ejecución (main.c)

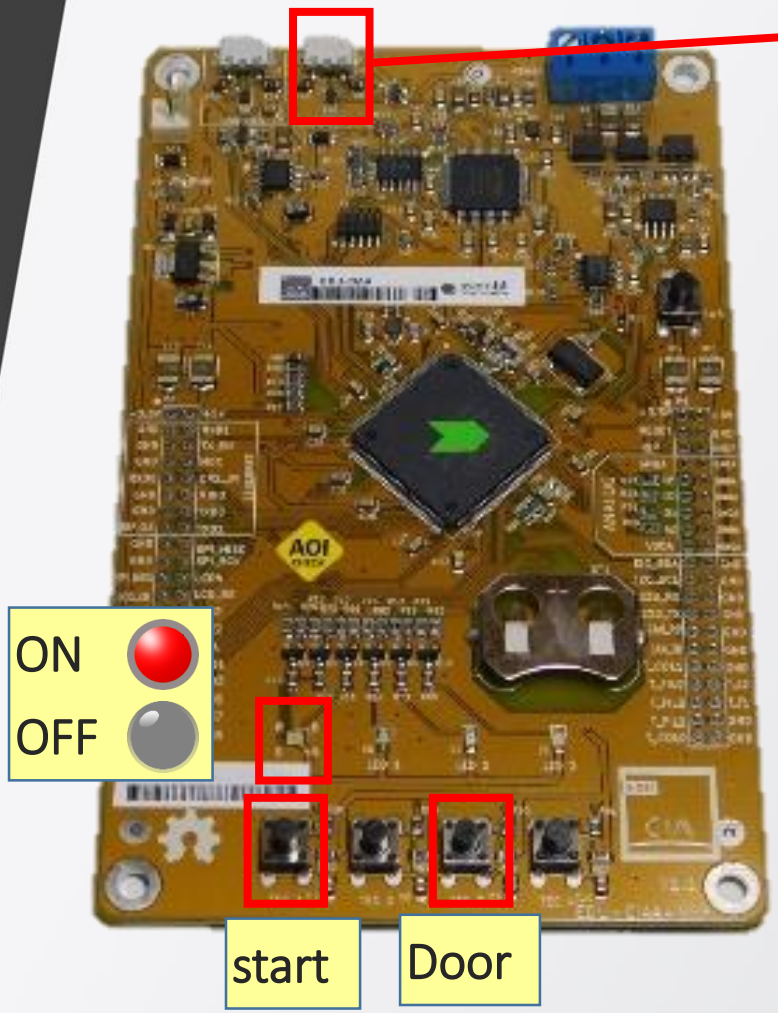
```c
/* ---------------------------------- Notes -----------
/* ------------------------------ Include files -------
#include "rkh.h"
#include "oven.h"
#include "bsp.h"

/* ------------------------------ Local macros --------
/* -------------------------------- Constants ---------
#define QSTO_SIZE           4

/* ---------------------------- Local data types -----
/* ---------------------------- Global variables -----
/* ---------------------------- Local variables ------
static RKH_EVT_T *qsto[QSTO_SIZE];

/* ----------------------- Local function prototypes -
/* ---------------------------- Local functions ------
/* ---------------------------- Global functions -----
int
main(int argc, char *argv[])
{
    bsp_init(argc, argv);

    RKH_SMA_ACTIVATE(oven, qsto, QSTO_SIZE, 0, 0);

    rkh_fwk_enter();

    RKH_TRC_CLOSE();

    return 0;
}
```

Aplicaciones RKH sobre la CIAA

¿ Que hay que hacer?

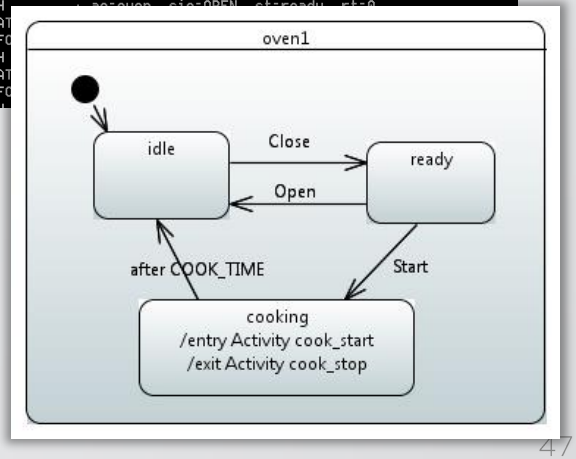
1. ✓ Preparar entorno de desarrollo.
2. ✓ Configurar RKH. **(rkhcfg.h)**
3. ✓ Trasladar modelo a código **(app.c/h)**
    1. ✓ Declarar AO, señales y acciones.
    2. ✓ Crear AO y estructura de statechart.
    3. ✓ Implementar acciones.
4. ✓ Iniciar plataforma y poner en ejecución. **(main.c)**
5. Codificar BSP
6. Validar.

DEMO EduCIAA



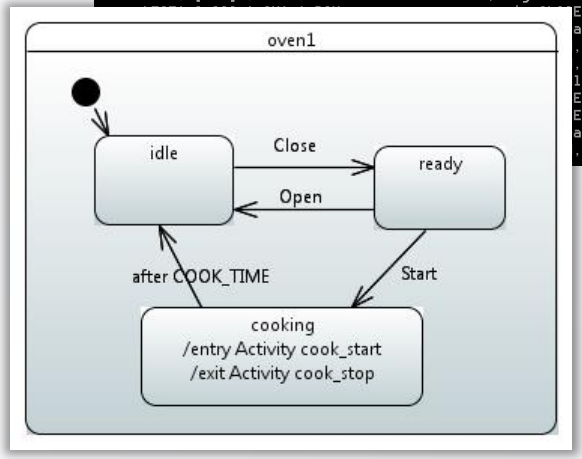
TRAZER

```
     0|  [  5] | FWK | STATE      : ao=oven, obj=0x012E8280, nm=idle
     0|  [  6] | FWK | STATE      : ao=oven, obj=0x012E8248, nm=ready
     0|  [  7] | FWK | STATE      : ao=oven, obj=0x012E8200, nm=cooking
     0|  [  8] | FWK | OBJ        : obj=0x012EA780, nm=oventim
     0|  [  9] | FWK | SIG        : sig=0, nm=OPEN
     0|  [ 10] | FWK | SIG        : sig=1, nm=CLOSE
     0|  [ 11] | FWK | SIG        : sig=2, nm=START
     0|  [ 12] | FWK | SIG        : sig=3, nm=TMREVT
     0|  [ 13] | FWK | SIG        : sig=4, nm=TERM
  2972|  [ 14] | SMA | FIFO       : ao=oven, sig=OPEN, snr=door, pid=0, rc=0
  2972|  [ 15] | SMA | DCH        : ao=oven, sig=OPEN, st=idle, rt=0
  2992|  [ 16] | SMA | FIFO       : ao=oven, sig=CLOSE, snr=door, pid=0, rc=0
  2992|  [ 17] | SMA | DCH        : ao=oven, sig=CLOSE, st=idle, rt=0
  2992|  [ 18] | SM  | STATE      : ao=oven, nxtst=ready
  3030|  [ 19] | SMA | FIFO       : ao=oven, sig=START, snr=panel, pid=0, rc=0
  3030|  [ 20] | SMA | DCH        : ao=oven, sig=START, st=ready, rt=0
  3030|  [ 21] | SM  | STATE      : ao=oven, nxtst=cooking
  3530|  [ 22] | SMA | FIFO       : ao=oven, sig=TMREVT, snr=rkh_tick, pid=0, rc=0
  3530|  [ 23] | SMA | DCH        : ao=oven, sig=TMREVT, st=cooking, rt=0
  3530|  [ 24] | SM  | STATE      : ao=oven, nxtst=idle
  4511|  [ 25] | SMA | FIFO       : ao=oven, sig=OPEN, snr=door, pid=0, rc=0
  4511|  [ 26] | SMA | DCH        : ao=oven, sig=OPEN, st=idle, rt=0
  4535|  [ 27] | SMA | FIFO       : ao=oven, sig=CLOSE, snr=door, pid=0, rc=0
  4535|  [ 28] | SMA | DCH        : ao=oven, sig=CLOSE, st=idle, rt=0
  4535|  [ 29] | SM  | STATE      : ao=oven, nxtst=ready
  4632|  [ 30] | SMA | FIFO       : ao=oven, sig=OPEN, snr=door, pid=0, rc=0
  4632|  [ 31] | SMA | DCH
  4632|  [ 32] | SM  | STAT
  4673|  [ 33] | SMA | FIFO
  4673|  [ 34] | SMA | DCH
  4673|  [ 35] | SM  | STAT
  4699|  [ 36] | SMA | FIFO
```



DEMO x86

```
     0| [  5] | FWK | STATE      : ao=oven, obj=0x012E8280, nm=idle
     0| [  6] | FWK | STATE      : ao=oven, obj=0x012E8248, nm=ready
     0| [  7] | FWK | STATE      : ao=oven, obj=0x012E8200,
     0| [  8] | FWK | OBJ        : obj=0x012EA780, nm=
     0| [  9] | FWK | SIG        : sig=0, nm=OPEN
     0| [ 10] | FWK | SIG        : sig=1, nm=CLOSE
     0| [ 11] | FWK | SIG        : sig=2, nm=START
     0| [ 12] | FWK | SIG        : sig=3, nm=TMREVT
     0| [ 13] | FWK | SIG        : sig=4, nm=TERM
  2972| [ 14] | SMA | FIFO       : ao=oven, sig=OPEN, snr=door, pid=0, rc=0
  2972| [ 15] | SMA | DCH        : ao=oven, sig=OPEN, st=idle, rt=0
  2992| [ 16] | SMA | FIFO       : ao=oven, sig=CLOSE, snr=door, pid=0, rc=0
  2992| [ 17] | SMA | DCH        : ao=oven, sig=CLOSE, st=idle, rt=0
  2992| [ 18] | SM  | STATE      : ao=oven, nxtst=ready
  3030| [ 19] | SMA | FIFO       : ao=oven, sig=START, snr=panel, pid=0, rc=0
  3030| [ 20] | SMA | DCH        : ao=oven, sig=START, st=ready, rt=0
  3030| [ 21] | SM  | STATE      : ao=oven, nxtst=cooking
  3530| [ 22] | SMA | FIFO       : ao=oven, sig=TMREVT, snr=rkh_tick, pid=0, rc=0
  3530| [ 23] | SMA | DCH        : ao=oven, sig=TMREVT, st=cooking, rt=0
  3530| [ 24] | SM  | STATE      : ao=oven, nxtst=idle
  4511| [ 25] | SMA | FIFO       : ao=oven, sig=OPEN, snr=door, pid=0, rc=0
  4511| [ 26] | SMA | DCH        : ao=oven, sig=OPEN, st=idle, rt=0
  4535| [ 27] | SMA | FIFO       : ao=oven, sig=CLOSE, snr=door, pid=0, rc=0
```



TRAZER

Linux

GCC

Microsoft Windows

Microsoft Visual Studio 2010 Express

La misma aplicación corre en cualquier plataforma

Aplicaciones RKH sobre la CIAA

¿ Que hay que hacer?

1. ✓ Preparar entorno de desarrollo.
2. ✓ Configurar RKH. **(rkhcfg.h)**
3. ✓ Trasladar modelo a código **(app.c/h)**
    1. ✓ Declarar AO, señales y acciones.
    2. ✓ Crear AO y estructura de statechart.
    3. ✓ Implementar acciones.
4. ✓ Iniciar plataforma y poner en ejecución. **(main.c)**
5. Codificar BSP
6. Validar.

Aplicaciones RKH sobre la CIAA

5

Codificar BSP

5. Codificar BSP

El BSP concentra las dependencias contra la plataforma

Assert (`rkhassert.h`)
- `rkh_assert`

BSP
- Init RKH
- acceso al hardware

Hooks (`rkhfwk_hook.h`)
- `rkh_hook_dispatch`
- `rkh_hook_signal`
- `rkh_hook_timeout`
- `rkh_hook_start`
- `rkh_hook_exit`
- `rkh_hook_idle`
- `rkh_hook_timetick`
- `rkh_hook_putTrcEvt`

-acceso a periféricos
-stacks
-middleware
-3trd Party libs

Trace (`rkhtrc_out.h`)
- `rkh_trc_open`
- `rkh_trc_close`
- `rkh_trc_flush`
- `rkh_trc_getts`

5. Codificar BSP

EduCIAA BSP

5. Codificar BSP EduCIAA (bsp.h)

```
/* ---------------------------------- Macros ------------------------------- */
/* --------------------------------- Constants ----------------------------- */
#define BSP_TICK_RATE_MS    (1000/RKH_CFG_FWK_TICK_RATE_HZ)

/* ---------------------------------- Data types --------------------------- */
/* ------------------------- External variables ---------------------------- */
/* ------------------------- Function prototypes --------------------------- */
void bsp_init(int argc, char *argv[]);
void bsp_ovenInit(void);
void bsp_timeTick(void);
void bsp_publishSwitchEvt(ruint s, ruint debsw);
void bsp_emitterOn(void);
void bsp_emitterOff(void);
```

5. Codificar BSP EduCIAA (bsp.c)

Inicialización de hardware

Inicialización RKH

Ajuste Trace Runtime Filters

```
void
bsp_init(int argc, char *argv[])
{
    (void)argc;
    (void)argv;

    boardConfig();

    rkh_fwk_init();

    RKH_FILTER_ON_GROUP(RKH_TRC_ALL_GROUPS);
    RKH_FILTER_ON_EVENT(RKH_TRC_ALL_EVENTS);
    RKH_FILTER_OFF_EVENT(RKH_TE_TMR_TOUT);
    RKH_FILTER_OFF_EVENT(RKH_TE_SM_DCH);
    RKH_FILTER_OFF_EVENT(RKH_TE_SM_STATE);
    RKH_FILTER_OFF_EVENT(RKH_TE_SM_EXE_ACT);
    RKH_FILTER_OFF_SMA(oven);
    RKH_FILTER_OFF_ALL_SIGNALS();

    RKH_TRC_OPEN();

#if defined(RKH_USE_TRC_SENDER)
    RKH_TR_FWK_OBJ(&panel);
    RKH_TR_FWK_OBJ(&door);
#endif

    RKH_ENA_INTERRUPT();
}
```

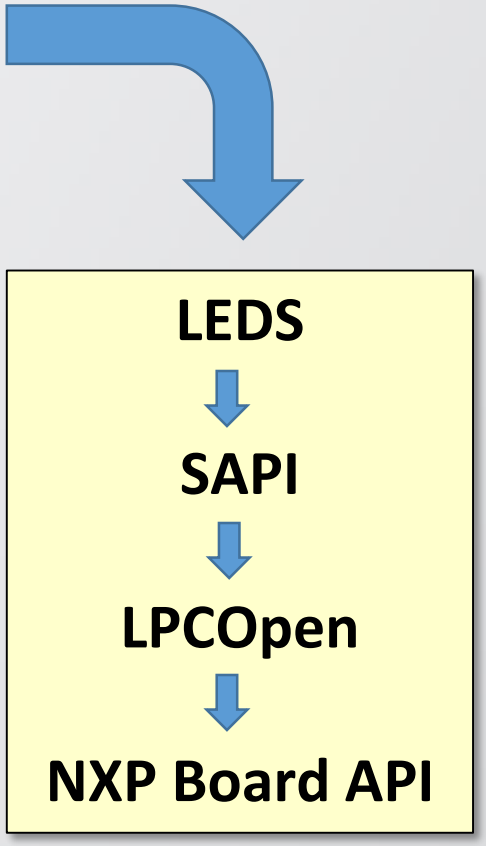
```
void
bsp_ovenInit(void)
{
    leds_init();
    switch_init();
}
```

```
void
switch_init( void )
{
    gpioConfig(TEC1, GPIO_INPUT);
    gpioConfig(TEC2, GPIO_INPUT);
    gpioConfig(TEC3, GPIO_INPUT);
}
```

```
void
leds_init(void)
{
    gpioConfig(LEDR, GPIO_OUTPUT);
    gpioConfig(LEDG, GPIO_OUTPUT);
    gpioConfig(LEDB, GPIO_OUTPUT);
}
```

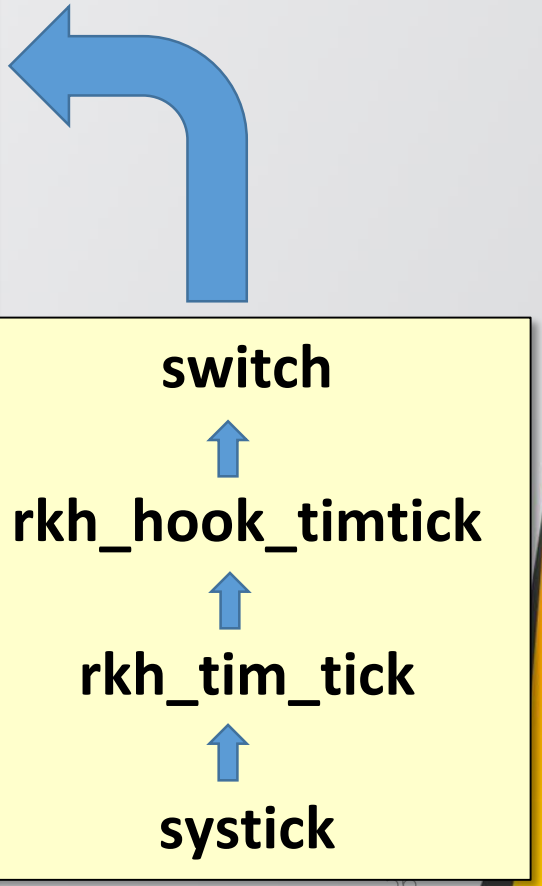
5. Codificar BSP EduCIAA (bsp.c)

```
void
bsp_emitterOn(void)
{
    leds_rgbSet(RGB_RED);
}

void
bsp_emitterOff(void)
{
    leds_rgbSet(RGB_BLACK);
}
```



5. Codificar BSP EduCIAA (bsp.c)

```
void
bsp_publishSwitchEvt(ruint s, ruint debsw)
{
    switch (s)
    {
        case START_SW:
            RKH_SMA_POST_FIFO(oven, &e_start, &panel);
            break;

#if (__STOP_BUTTON__ == RKH_ENABLED)
        case STOP_SW:
            RKH_SMA_POST_FIFO(oven, &e_stop, &panel);
            break;
#endif

        case DOOR_SW:
            if (debsw == SW_PRESS)
            {
                RKH_SMA_POST_FIFO(oven, &e_close, &door);
            }
            else
            {
                RKH_SMA_POST_FIFO(oven, &e_open, &door);
            }
            break;
    }
}
```



Aplicaciones RKH sobre la CIAA

¿ Que hay que hacer?

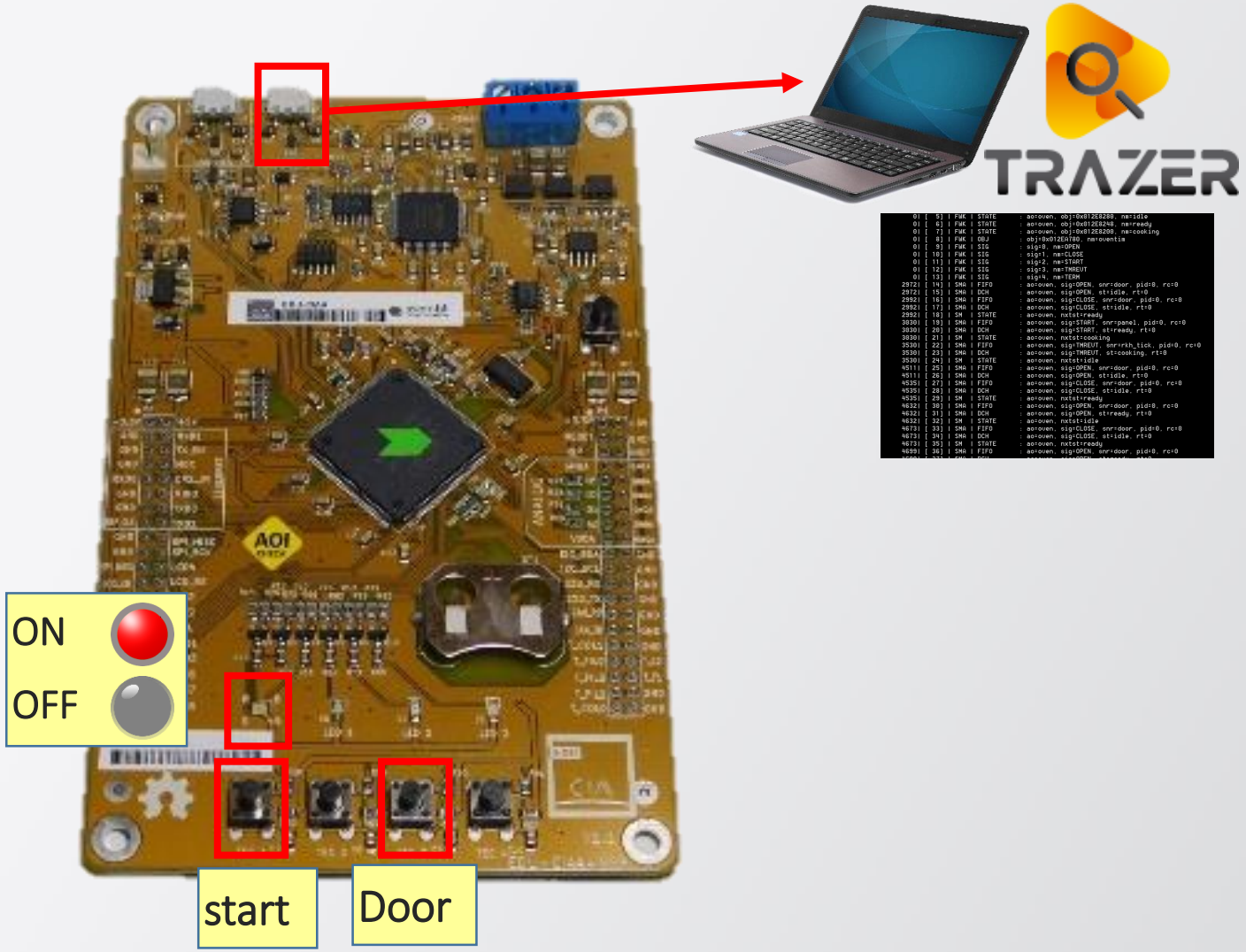
1. ✓ Preparar entorno de desarrollo.
2. ✓ Configurar RKH. **(rkhcfg.h)**
3. ✓ Trasladar modelo a código **(app.c/h)**
    1. ✓ Declarar AO, señales y acciones.
    2. ✓ Crear AO y estructura de statechart.
    3. ✓ Implementar acciones.
4. ✓ Iniciar plataforma y poner en ejecución. **(main.c)**
5. ✓ Codificar BSP
6. Validar.

Aplicaciones RKH sobre la CIAA

6

Validar

6. Validar (trazer)



Aplicaciones RKH sobre la CIAA

¿ Que hay que hacer?

1. ✓ Preparar entorno de desarrollo.
2. ✓ Configurar RKH. **(rkhcfg.h)**
3. ✓ Trasladar modelo a código **(app.c/h)**
    1. ✓ Declarar AO, señales y acciones.
    2. ✓ Crear AO y estructura de statechart.
    3. ✓ Implementar acciones.
4. ✓ Iniciar plataforma y poner en ejecución. **(main.c)**
5. ✓ Codificar BSP
6. ✓ Validar.

?

Entrenamiento Hands-on

Entrenamiento Hands-on

Oven demo

- El horno realiza la cocción por intervalos de tiempo fijos pre-programados.
- La cocción inicia al presionar Start con la puerta cerrada y se detiene al finalizar el intervalo programado o bien abriendo la puerta.
- Al finalizar un ciclo de cocción es posible iniciar uno nuevo presionando Start.
- Si la cocción se detiene abruptamente por apertura de la puerta, al cerrarla el ciclo reinicia automáticamente.

GitHub

https://github.com/dariosb/EduCIAA-RKH-demos

Oven demo:
Análisis - Especificaciones

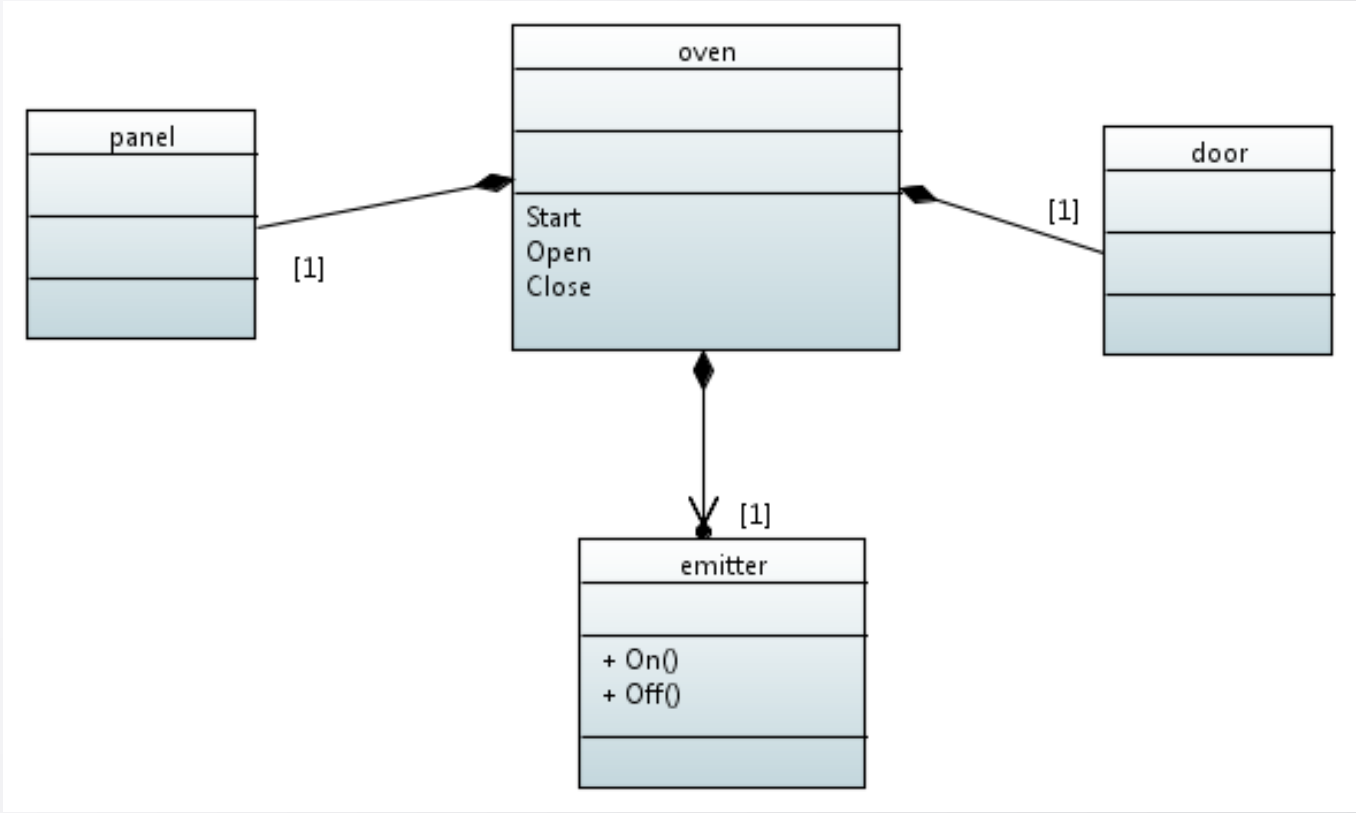
OV1: El usuario cierra la puerta y presiona Start, el horno enciende el emisor durante el intervalo de tiempo prefijado.

OV2: Si durante la cocción, el usuario abre la puerta, el horno apaga inmediatamente el emisor.

OV3: Al finalizar el ciclo, el usuario presiona Start, el horno enciende el emisor durante el intervalo de tiempo prefijado.

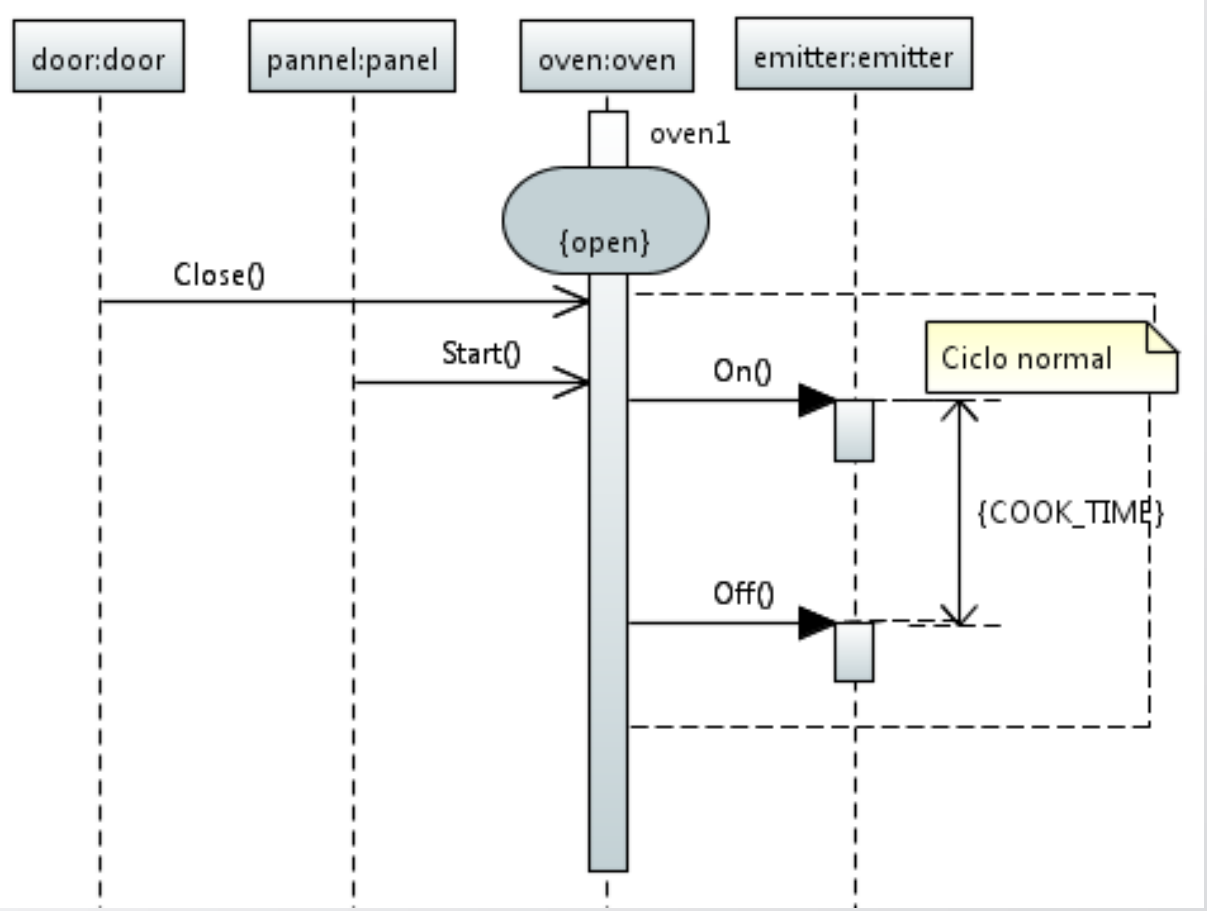
OV4: Si durante la cocción, el usuario abre la puerta y la vuelve a cerrar, el horno enciende el emisor durante el intervalo de tiempo prefijado.

Análisis – Colaboración

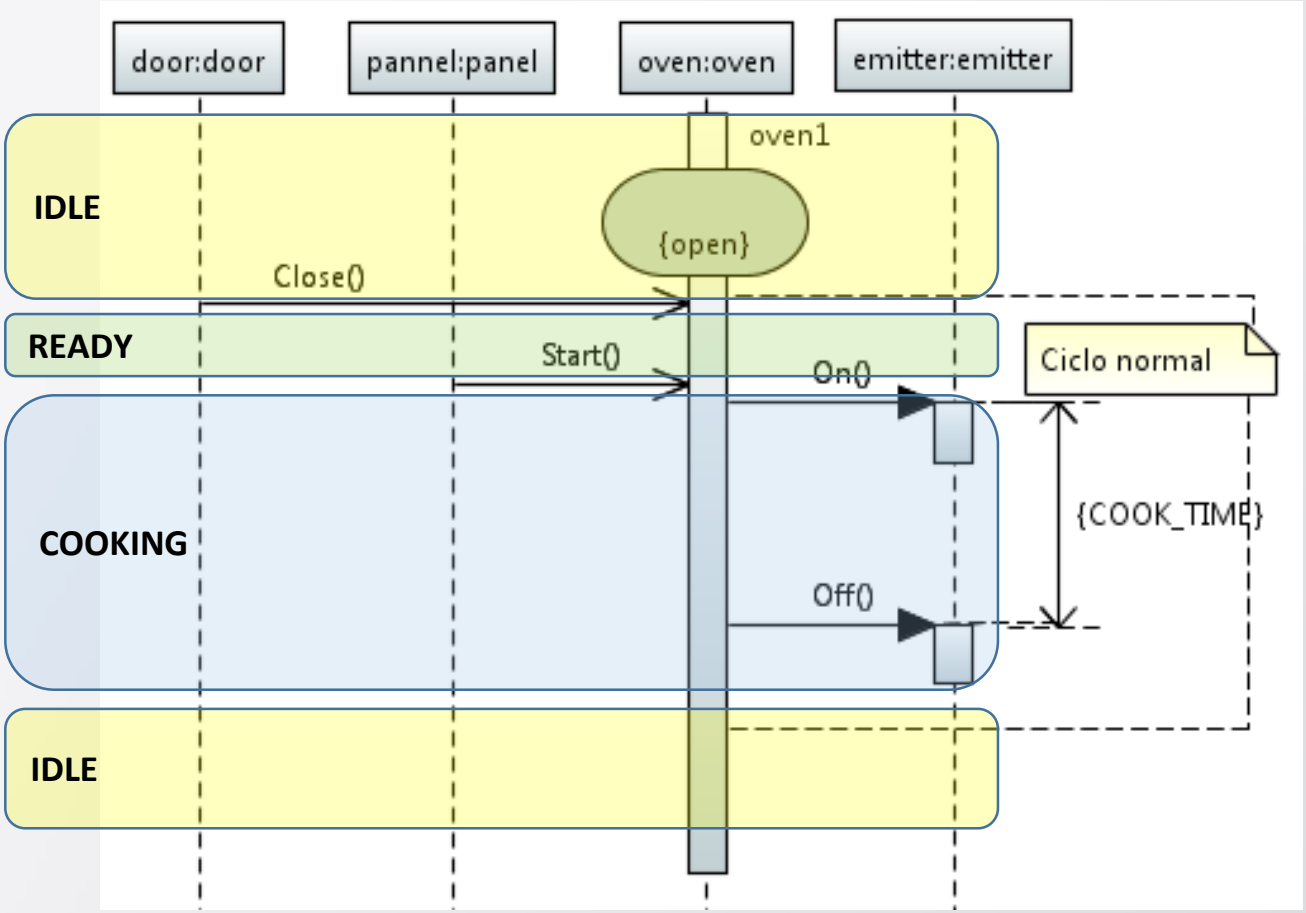


OV1: Análisis – Interacción

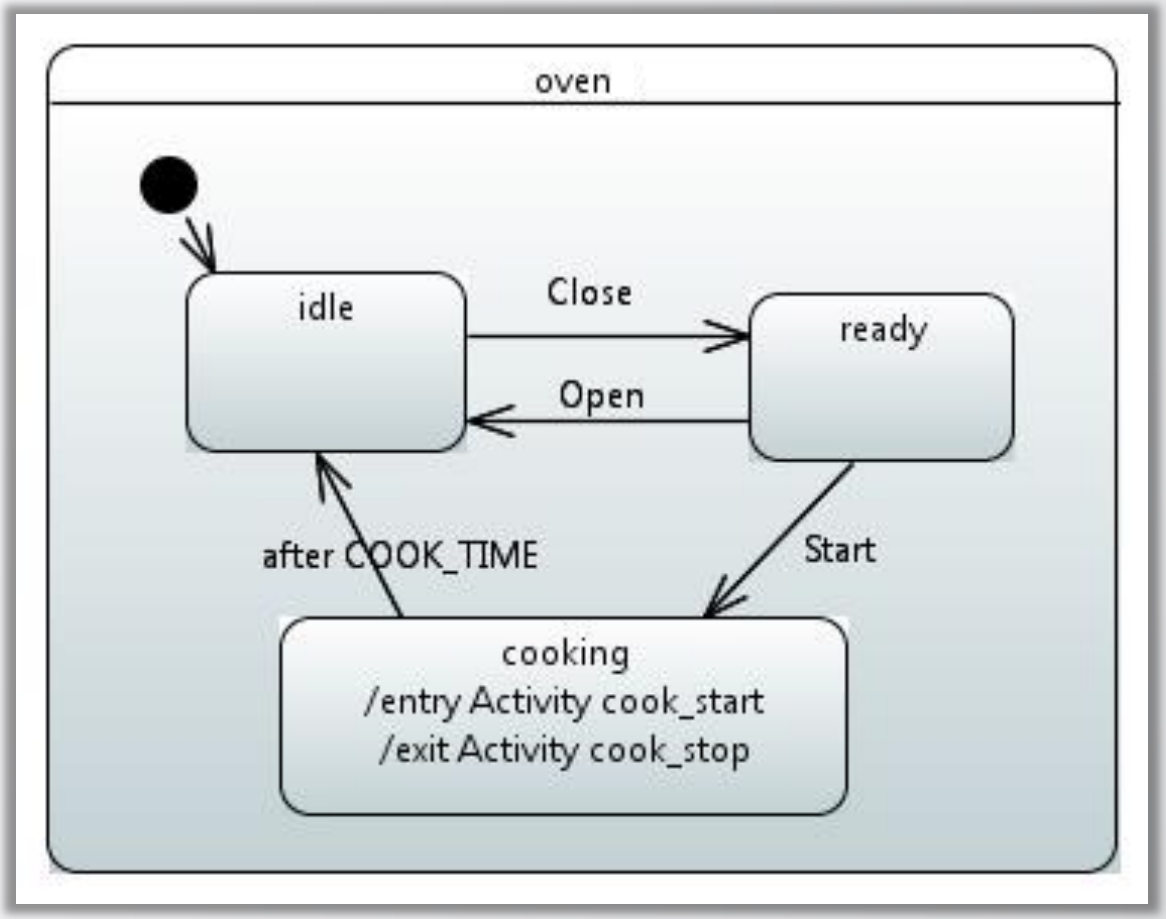
OV1: El usuario cierra la puerta y presiona Start, el horno enciende el emisor durante el intervalo de tiempo prefijado.



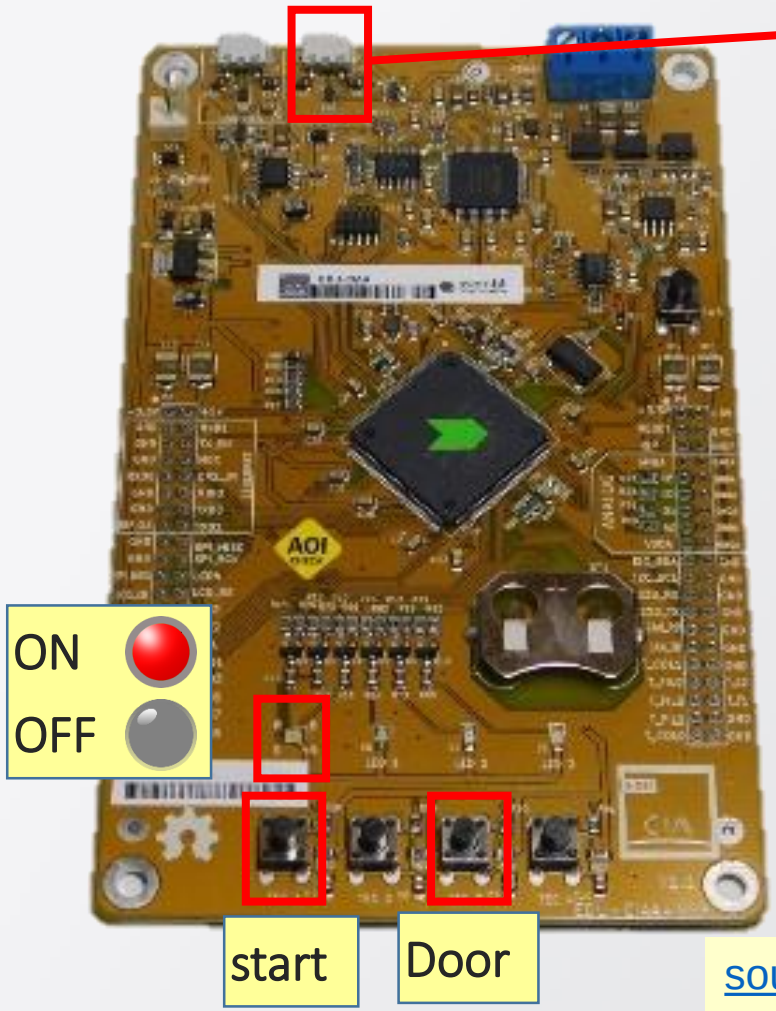
OV1: Análisis – Interacción

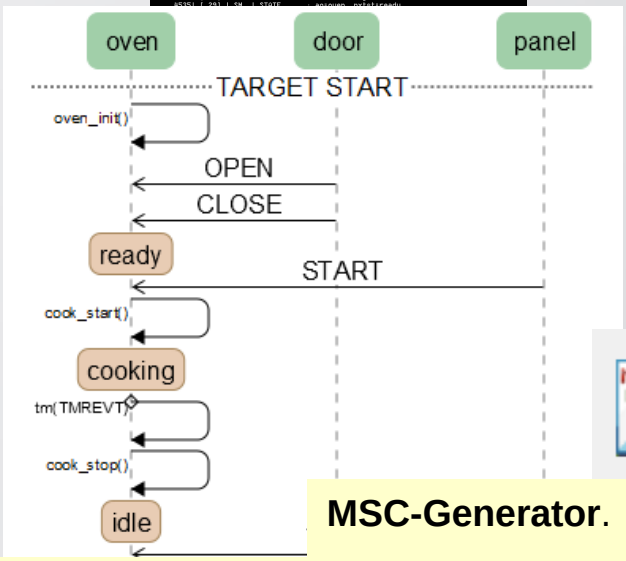


OV1: Análisis – Comportamiento



OV1: Validación



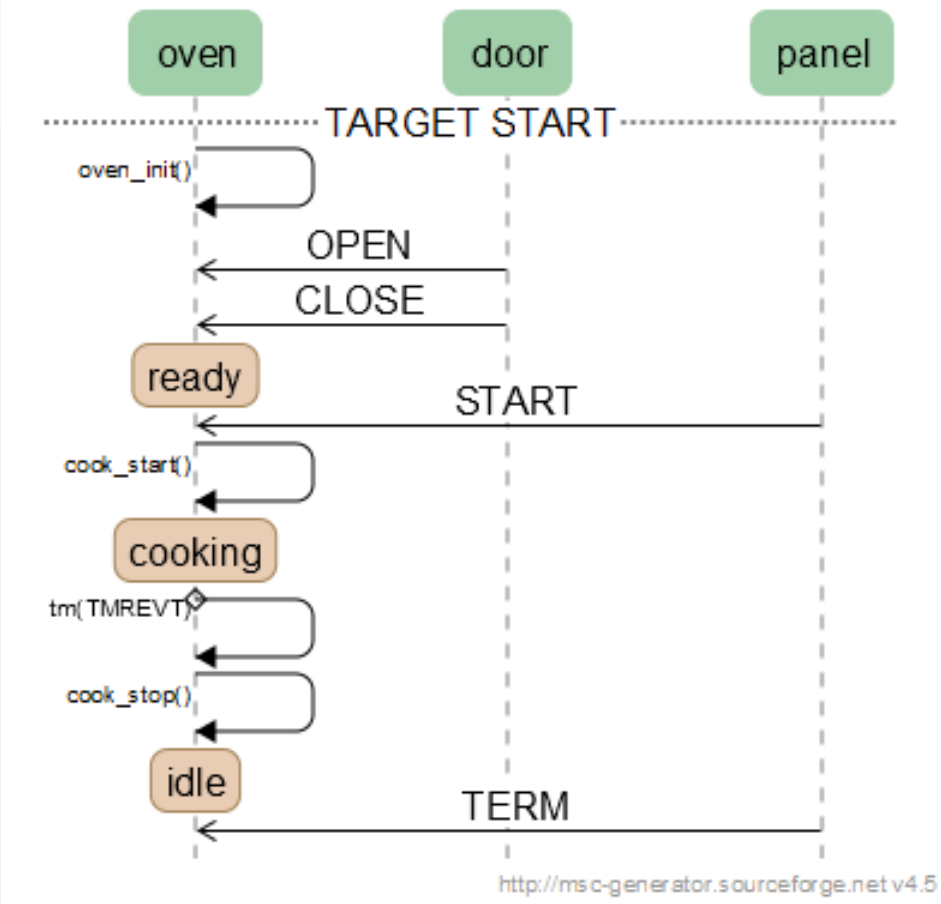


MSC-Generator.

sourceforge.net/projects/msc-generator

OV1: Validación

OV1: El usuario cierra la puerta y presiona Start, el horno enciende el emisor durante el intervalo de tiempo prefijado.



Oven demo:
Análisis - Especificaciones

OV1: El usuario cierra la puerta y presiona Start, el horno enciende el emisor durante el intervalo de tiempo prefijado.

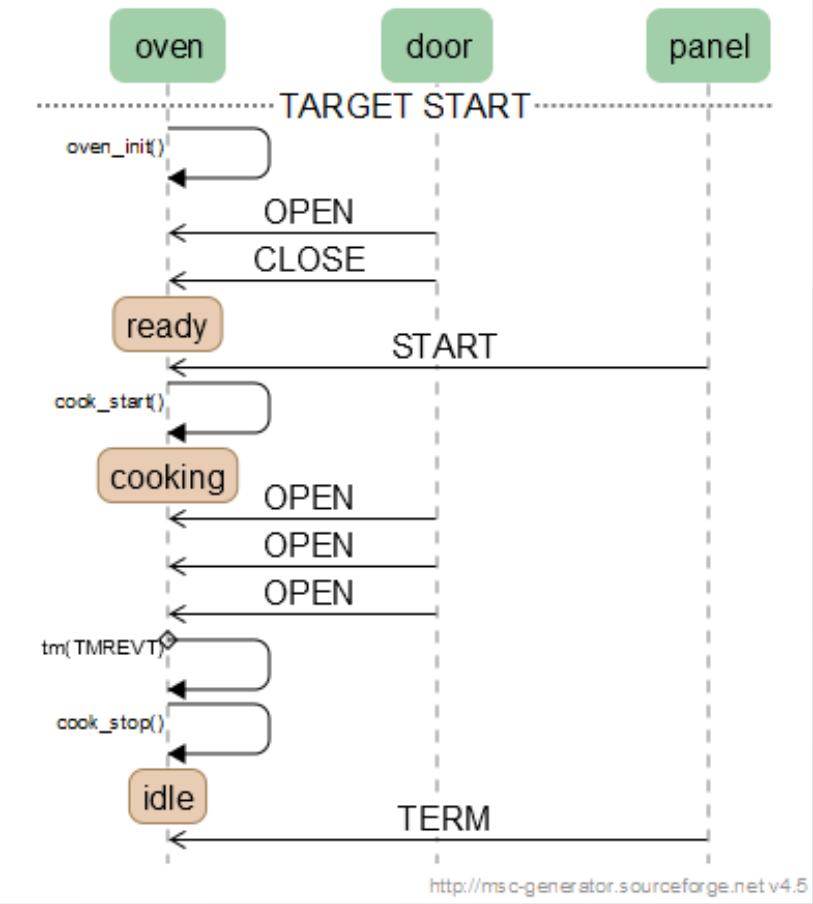
OV2: Si durante la cocción, el usuario abre la puerta, el horno apaga inmediatamente el emisor.

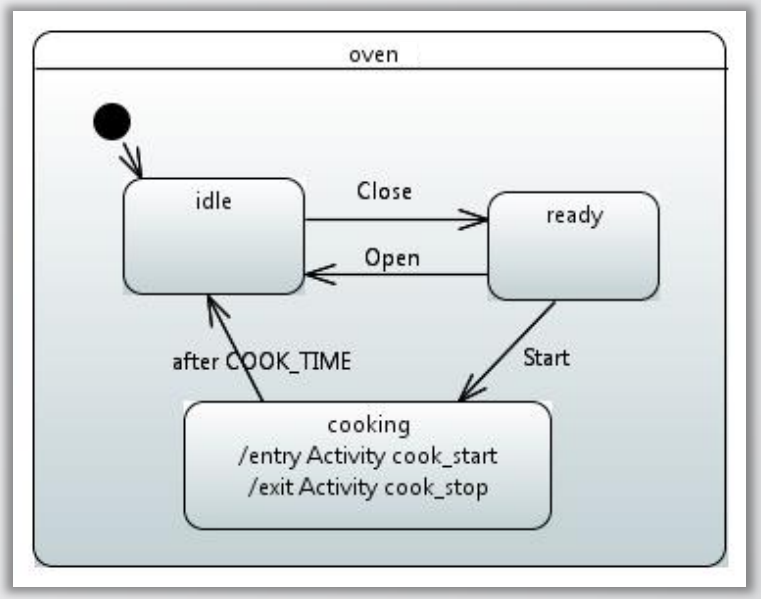
OV3: Al finalizar el ciclo, el usuario presiona Start, el horno enciende el emisor durante el intervalo de tiempo prefijado.

OV4: Si durante la cocción, el usuario abre la puerta y la vuelve a cerrar, el horno enciende el emisor durante el intervalo de tiempo prefijado.

OV2: Ejecución / Evaluación

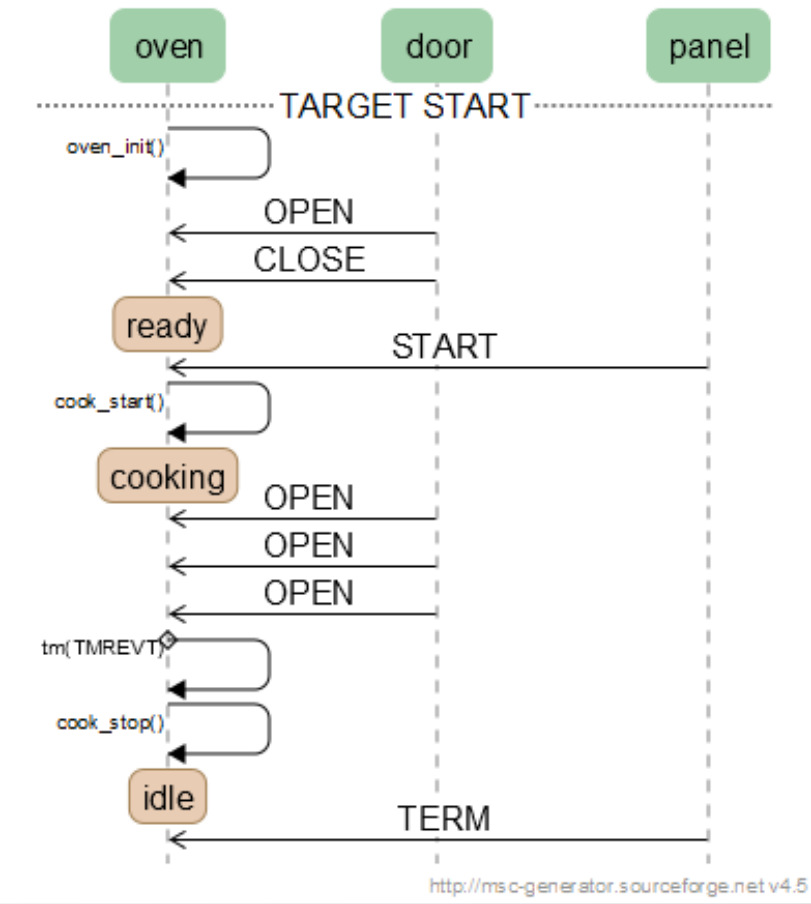
Ejecutar el escenario en la plataforma y verificar si funciona



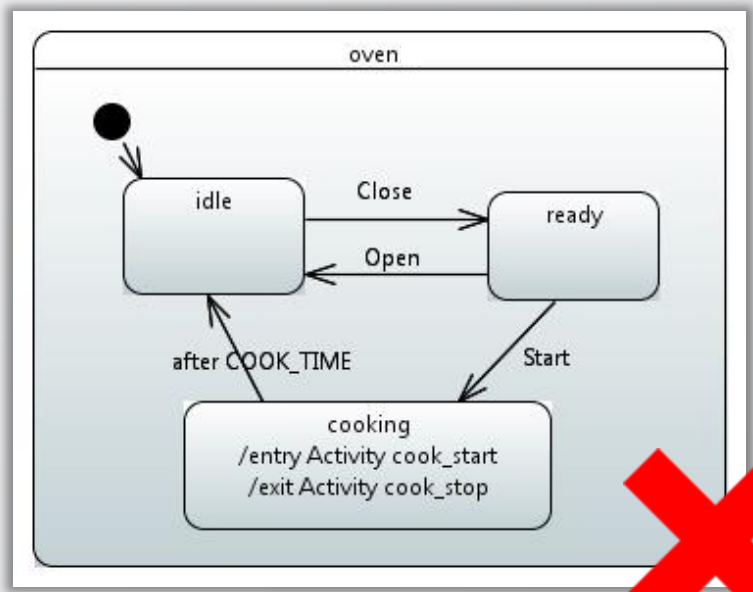


OV2: Ejecución / Evaluación

Ejecutar el escenario en la plataforma y verificar si funciona

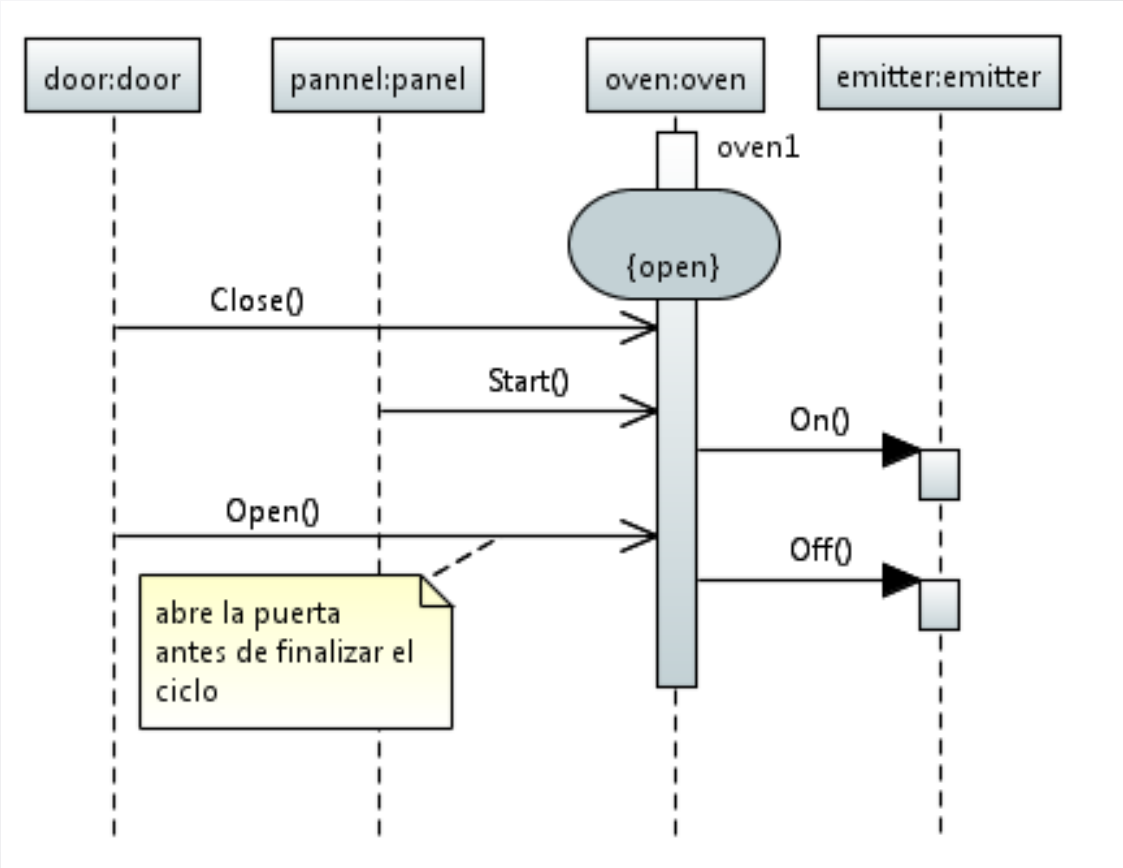


No funciona!!!!
No reacciona a OPEN mientras está en cooking.



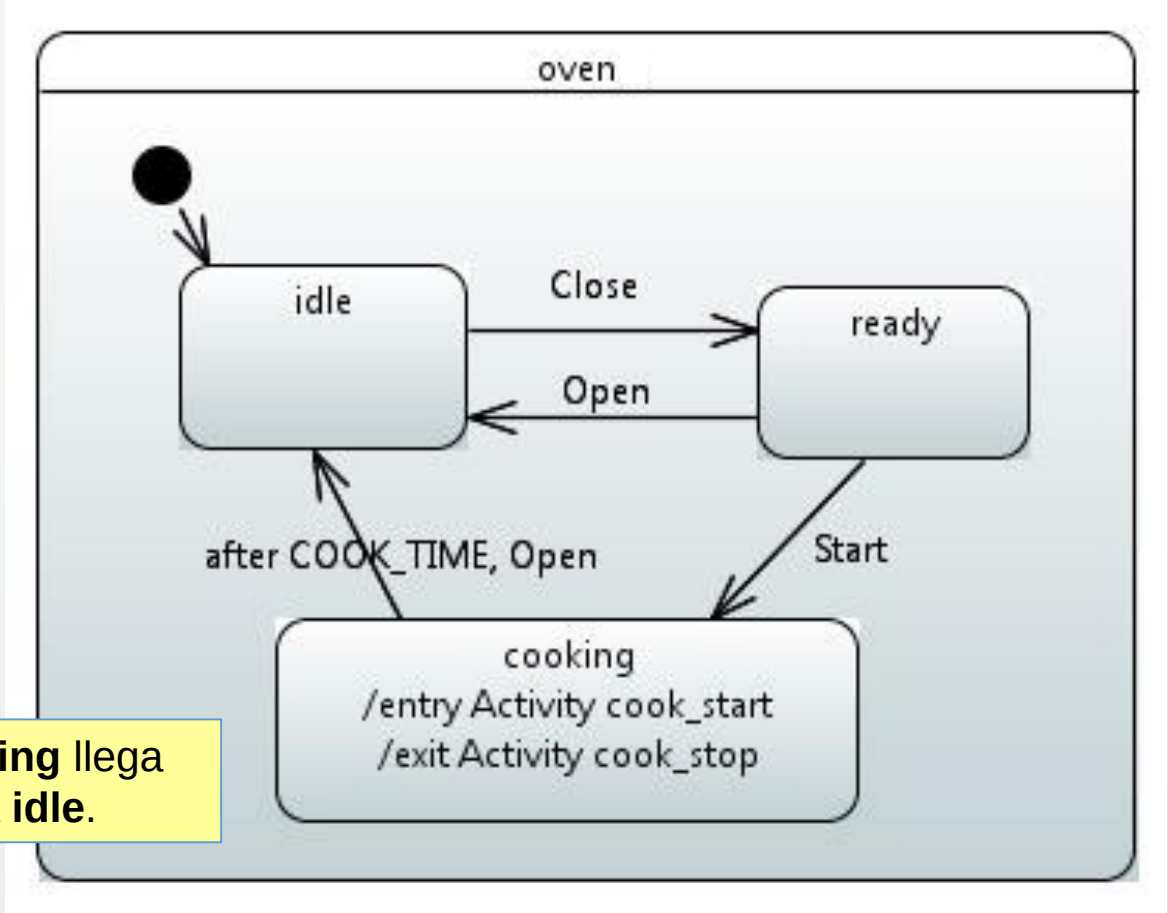
OV2: Análisis – Interacción

OV2: Si durante la cocción, el usuario abre la puerta, el horno apaga inmediatamente el emisor.



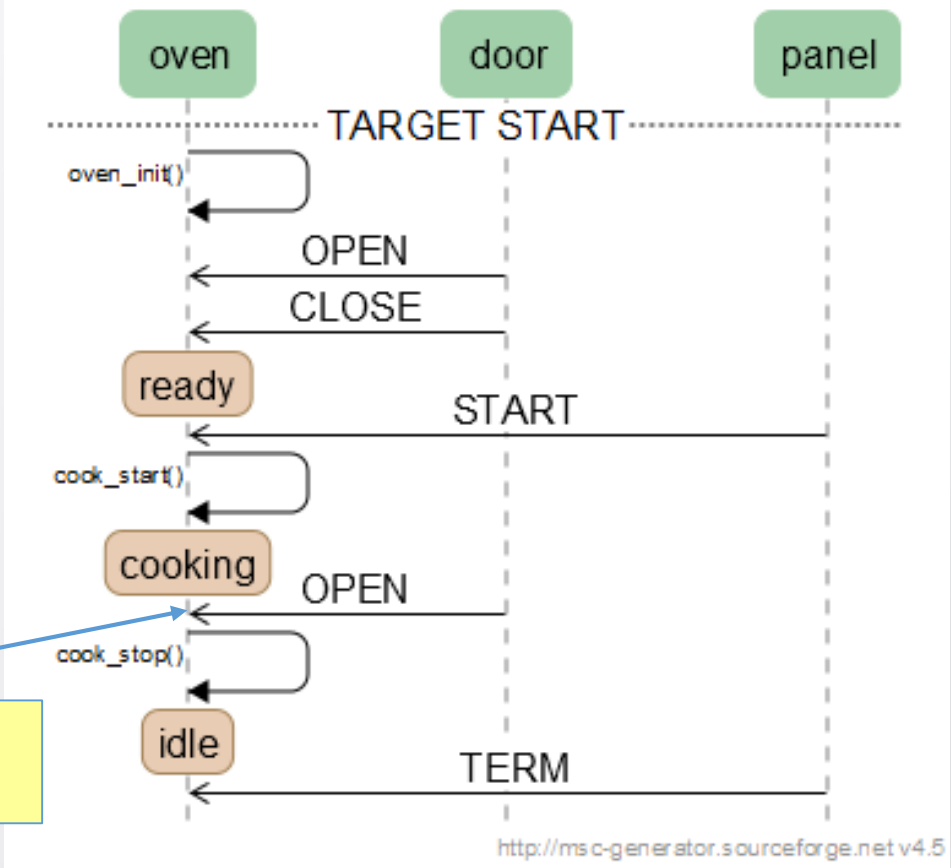
OV2: Análisis – Comportamiento

Ajuste del modelo y de la implementación



Si estando en **cooking** llega **Open**, transiciona a **idle**.

OV2: Validación



Si estando en **cooking** llega **Open**, transiciona a **idle**.

Oven demo: Análisis - Especificaciones

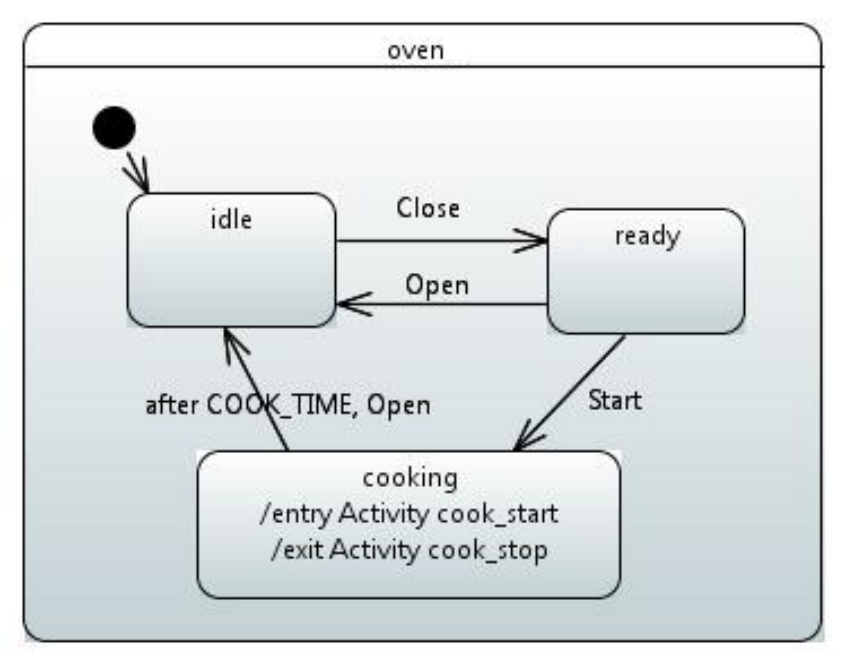
OV1: El usuario cierra la puerta y presiona Start, el horno enciende el emisor durante el intervalo de tiempo prefijado.

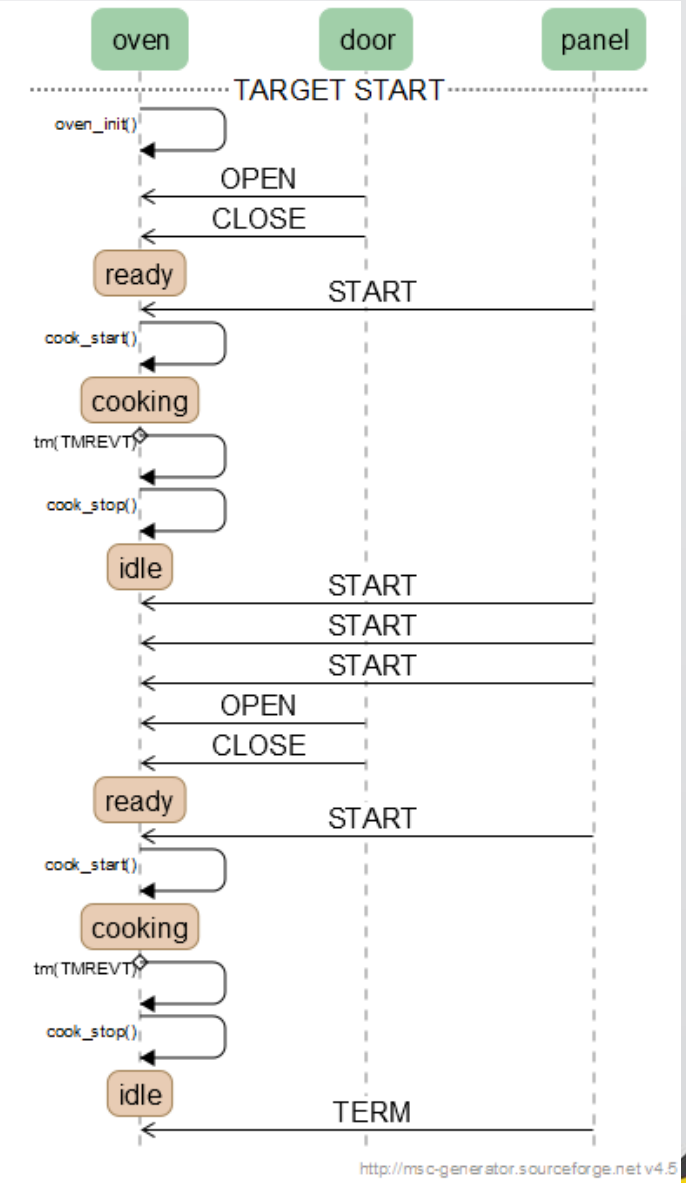
OV2: Si durante la cocción, el usuario abre la puerta, el horno apaga inmediatamente el emisor.

OV3: Al finalizar el ciclo, el usuario presiona Start, el horno enciende el emisor durante el intervalo de tiempo prefijado.

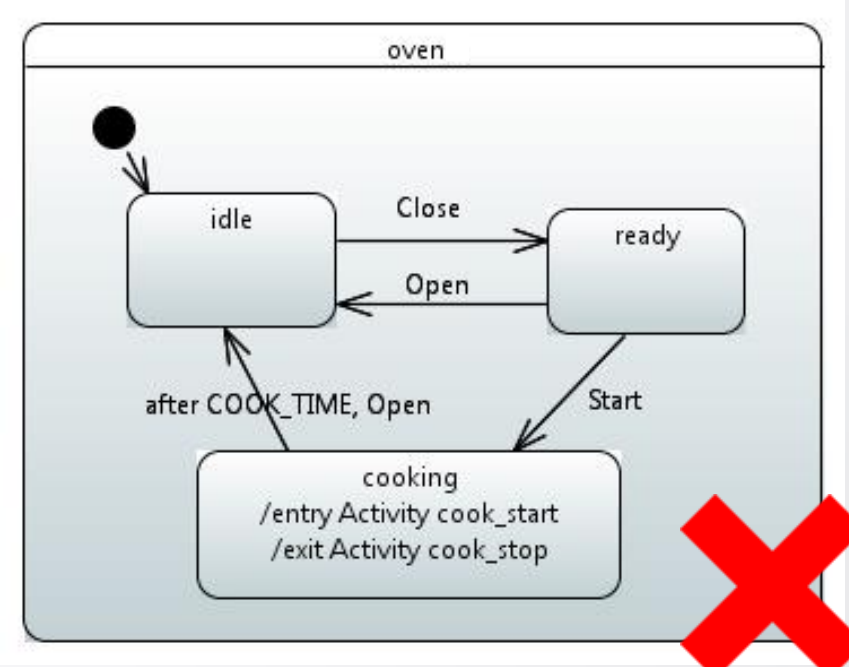
OV4: Si durante la cocción, el usuario abre la puerta y la vuelve a cerrar, el horno enciende el emisor durante el intervalo de tiempo prefijado.

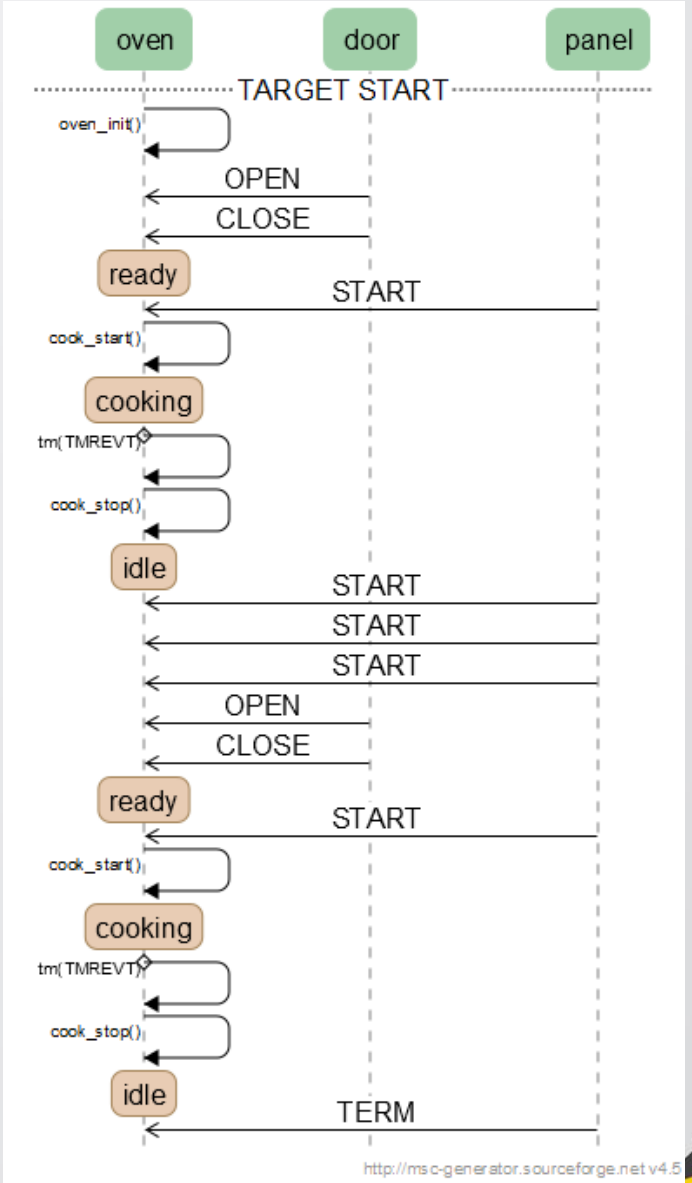
OV3: Ejecución / Evaluación





OV3: Ejecución / Evaluación

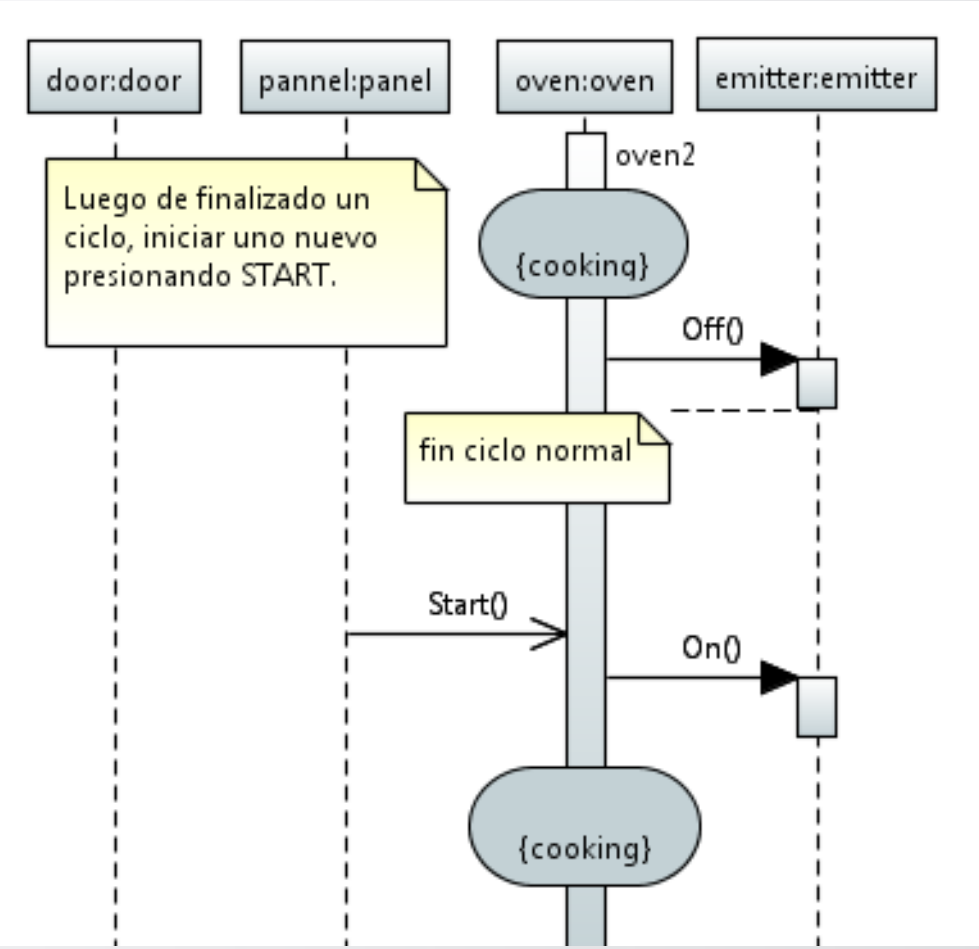




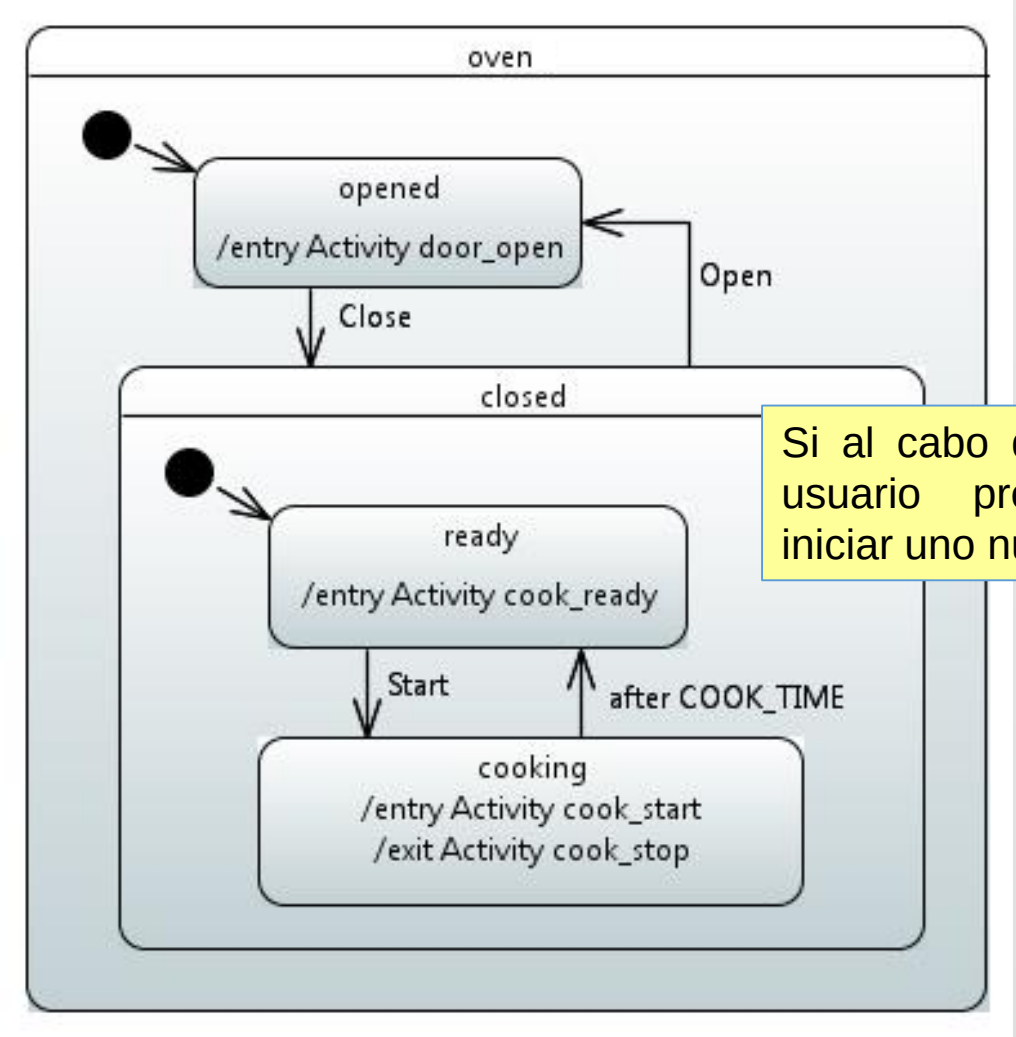
Si luego de un ciclo se quisiera iniciar uno nuevo, es necesario abrir y cerrar la puerta y presionar Start.

OV3: Análisis – Interacción

OV3: Al finalizar el ciclo, el usuario presiona Start, el horno enciende el emisor durante el intervalo de tiempo prefijado.



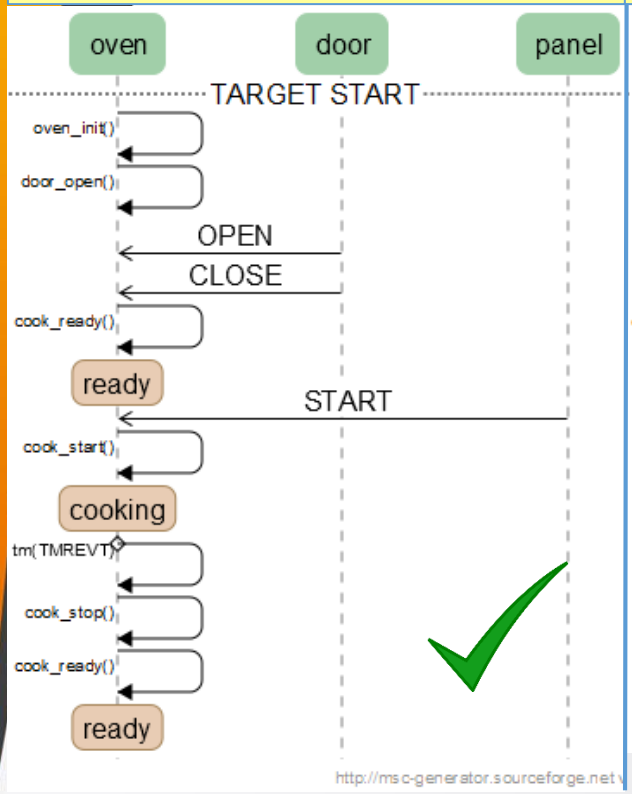
OV3: Análisis – Comportamiento



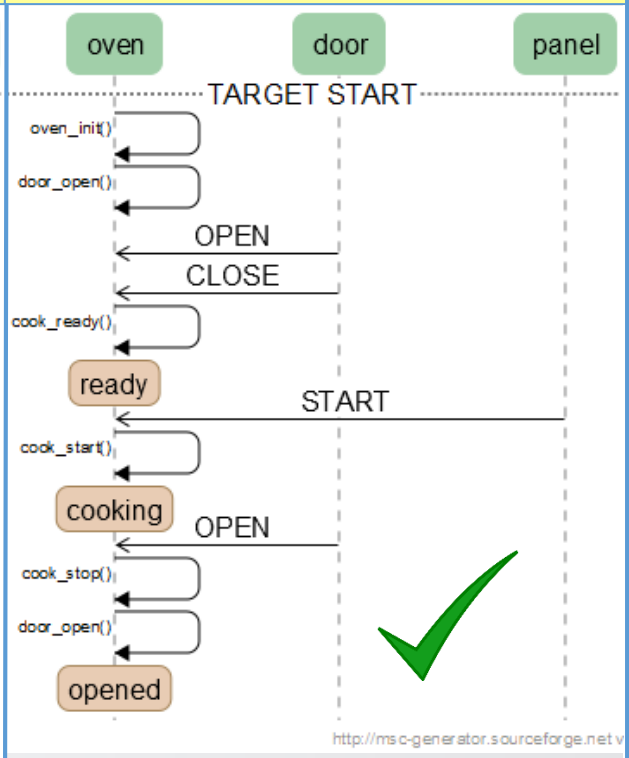
Si al cabo de un ciclo el usuario presiona Start, iniciar uno nuevo.

OV3: Validación

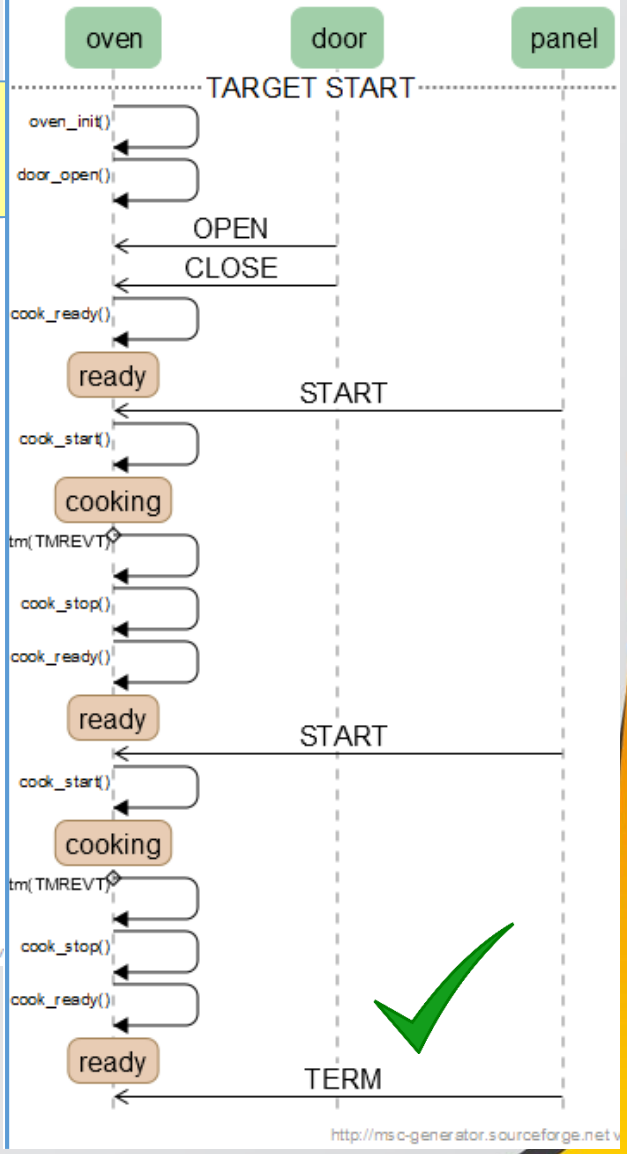
OV1: El usuario cierra la puerta y presiona Start, el horno enciende el emisor durante el intervalo de tiempo prefijado.



OV2: Si durante la cocción, el usuario abre la puerta, el horno apaga inmediatamente el emisor.



OV3: Al finalizar el ciclo, el usuario presiona Start, el horno enciende el emisor durante el intervalo de tiempo prefijado.



Oven demo:
Análisis - Especificaciones

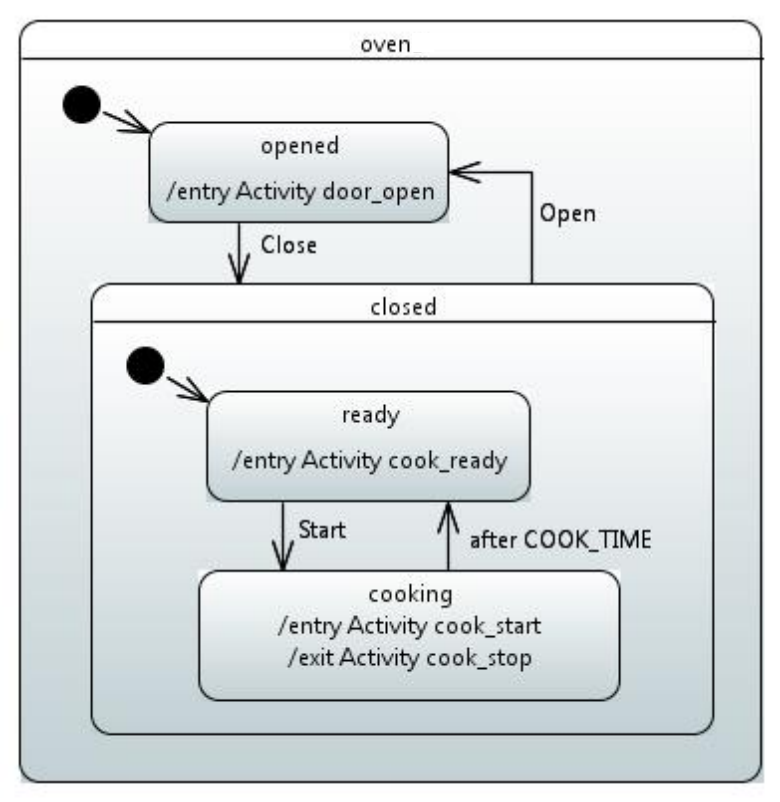
OV1: El usuario cierra la puerta y presiona Start, el horno enciende el emisor durante el intervalo de tiempo prefijado.

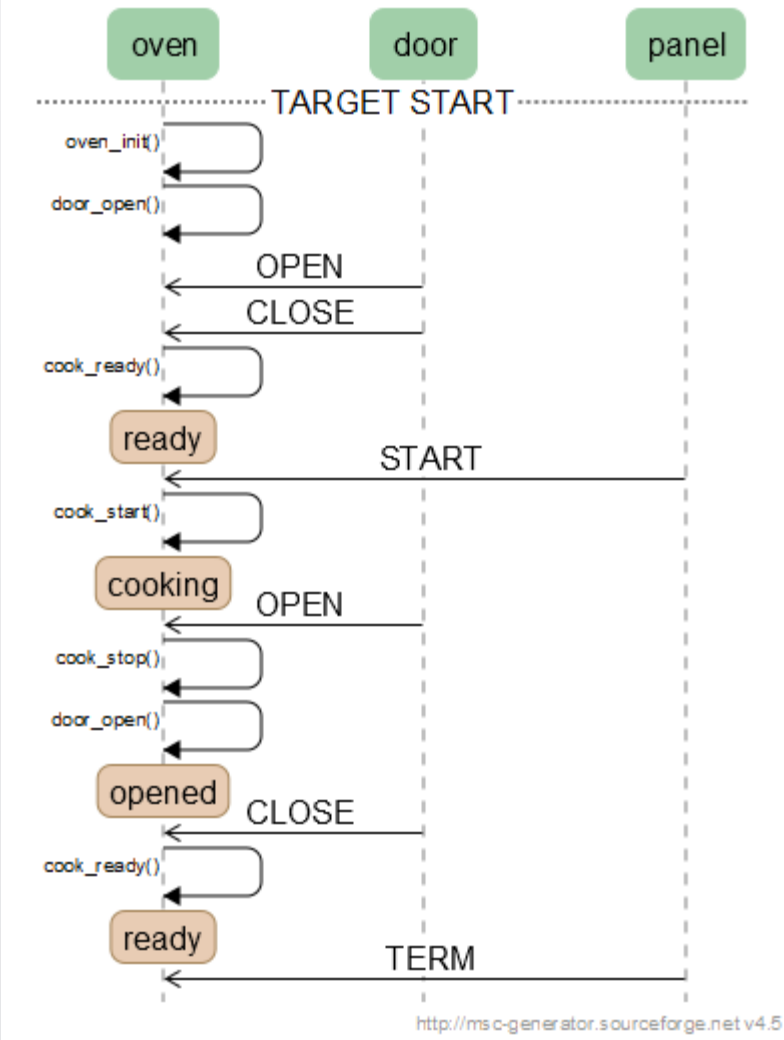
OV2: Si durante la cocción, el usuario abre la puerta, el horno apaga inmediatamente el emisor.

OV3: Al finalizar el ciclo, el usuario presiona Start, el horno enciende el emisor durante el intervalo de tiempo prefijado.

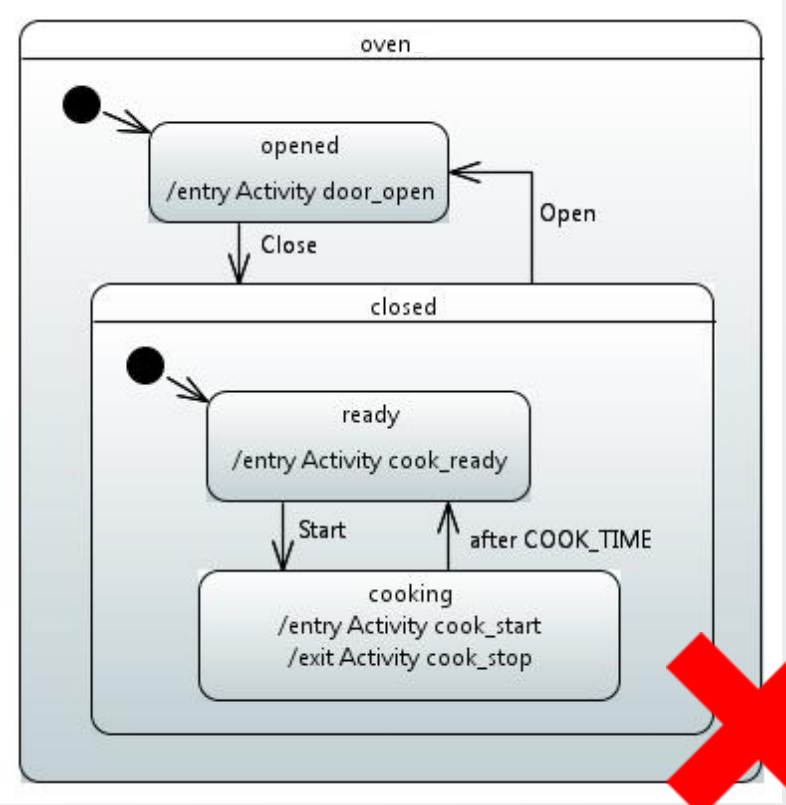
OV4: Si durante la cocción, el usuario abre la puerta y la vuelve a cerrar, el horno enciende el emisor durante el intervalo de tiempo prefijado.

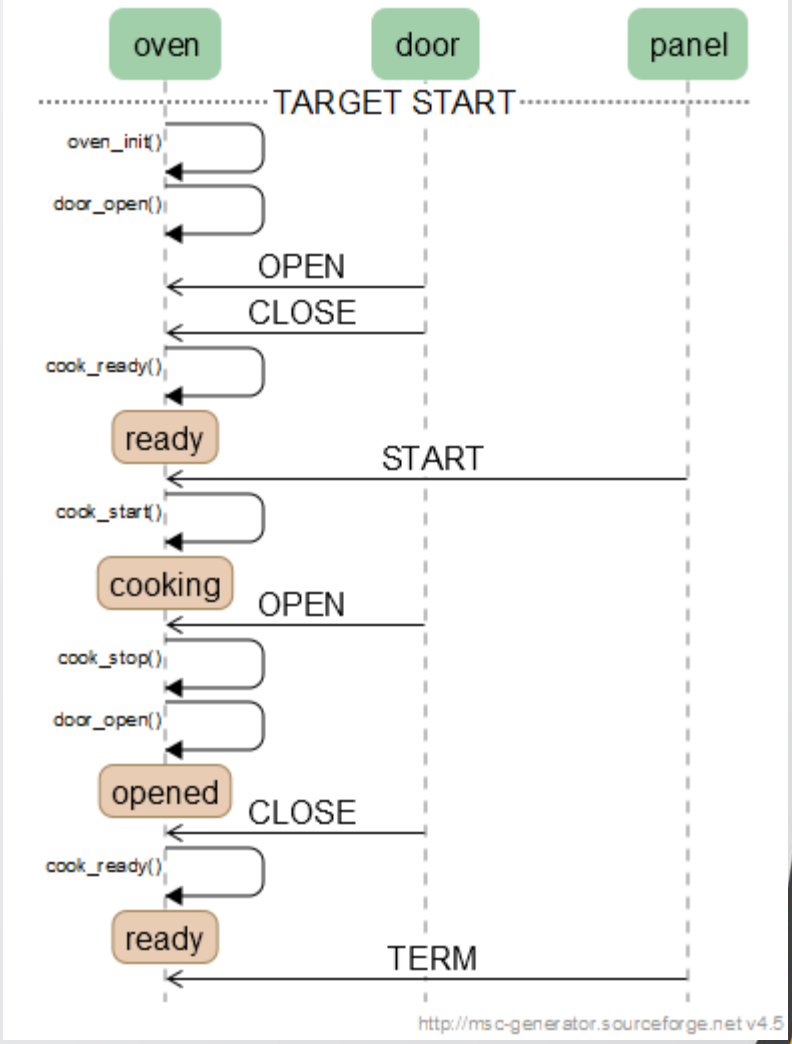
OV4: Ejecución / Evaluación





OV4: Ejecución / Evaluación

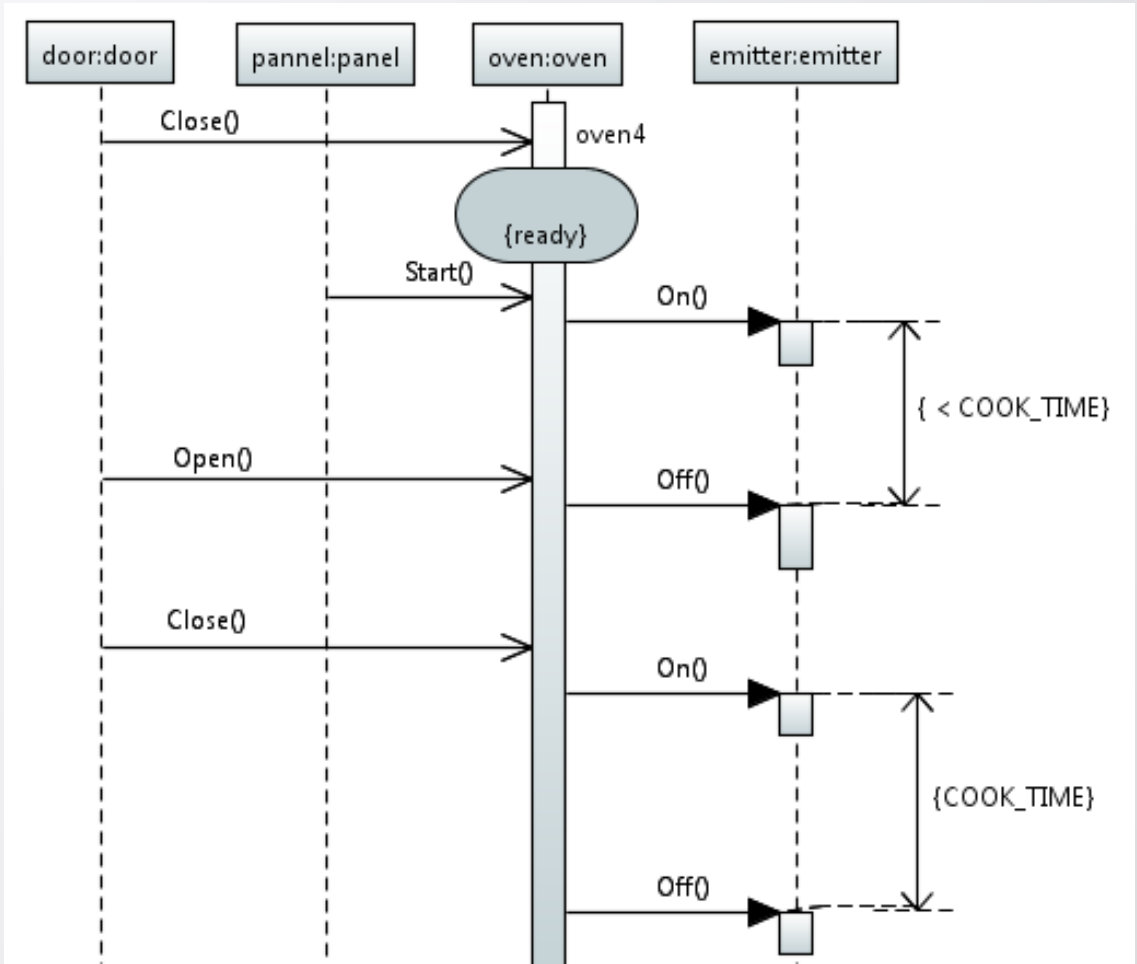




Queda en ready, no arranca solo. Es necesario presionar start para que inicie el ciclo.

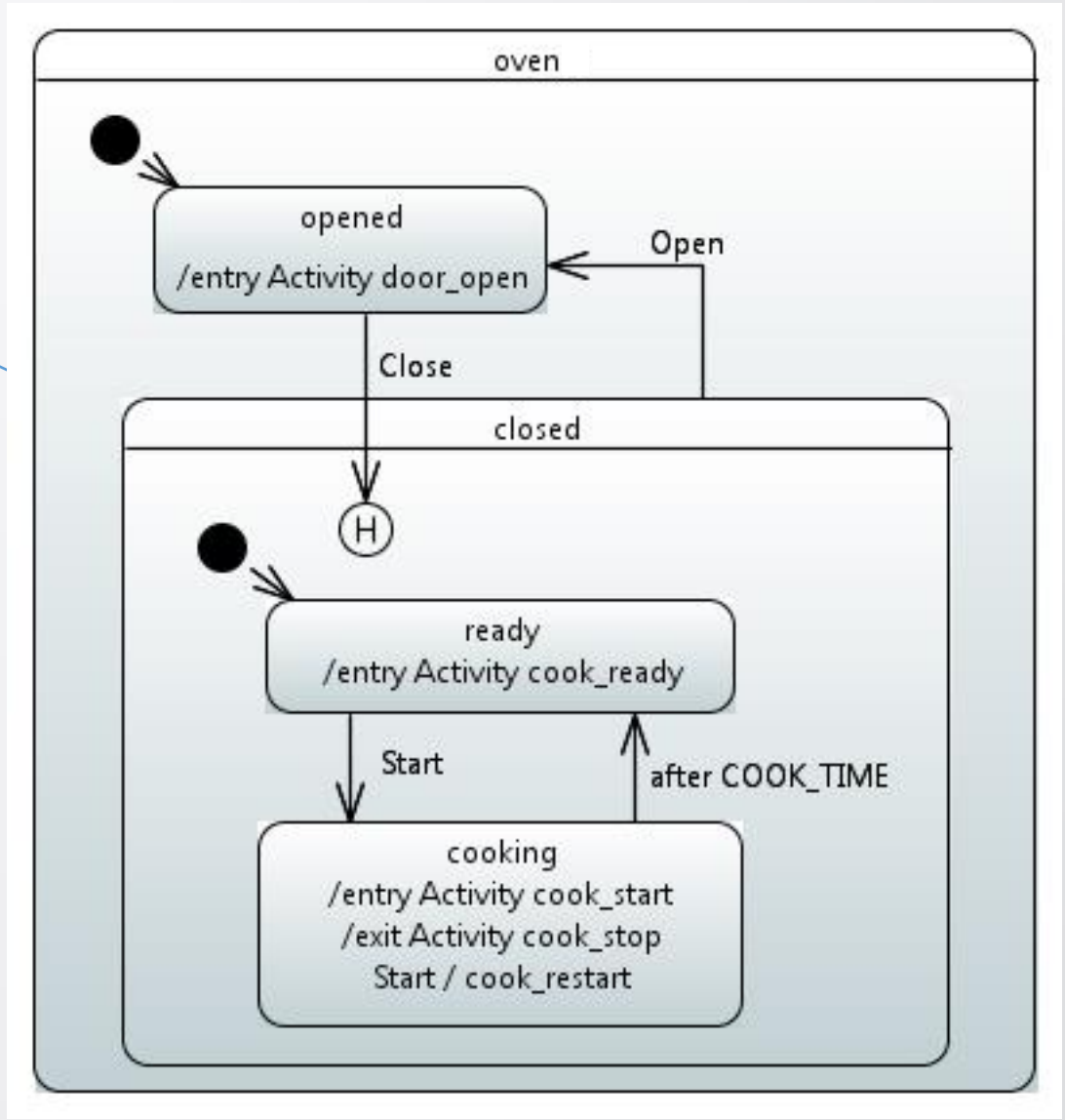
OV4: Análisis – Interacción

OV4: Si durante la cocción, el usuario abre la puerta y la vuelve a cerrar, el horno enciende el emisor durante el intervalo de tiempo prefijado.



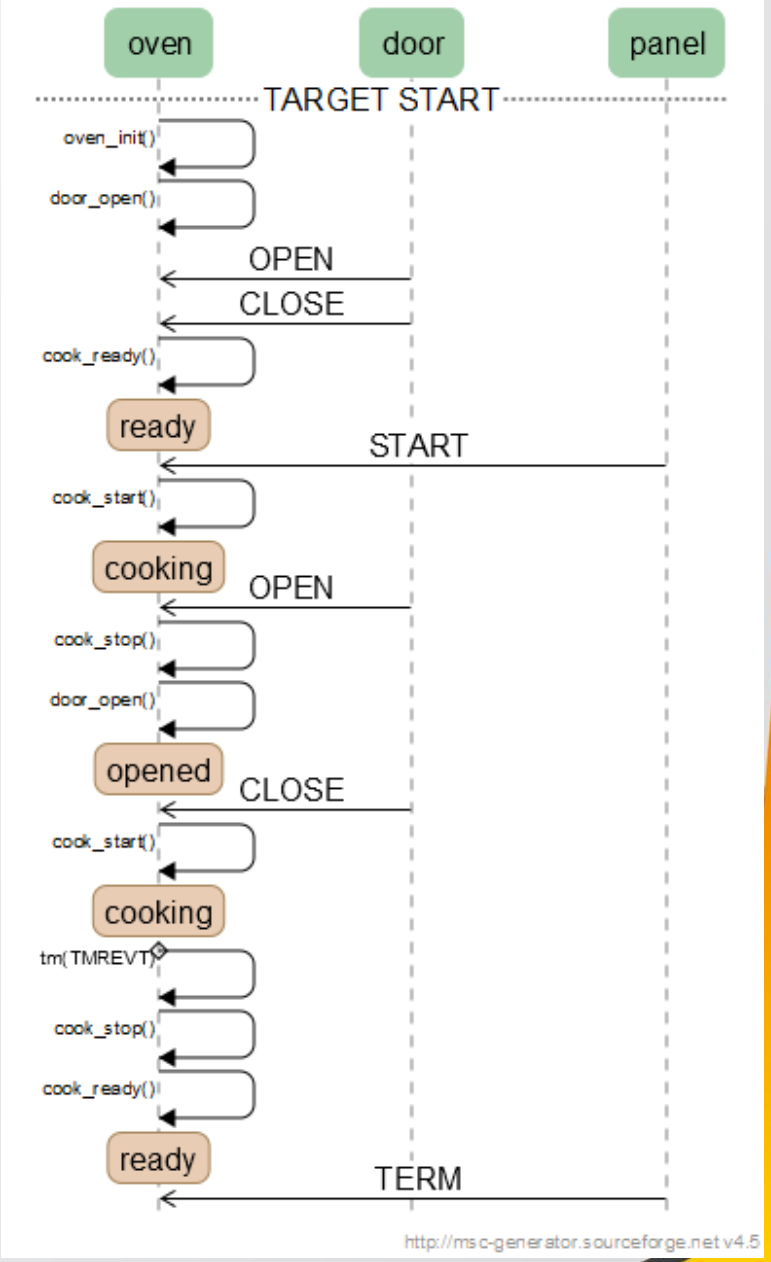
OV4: Análisis – Comportamiento

Si durante un ciclo la puerta se abre y vuelve a cerrar, reiniciar automáticamente.

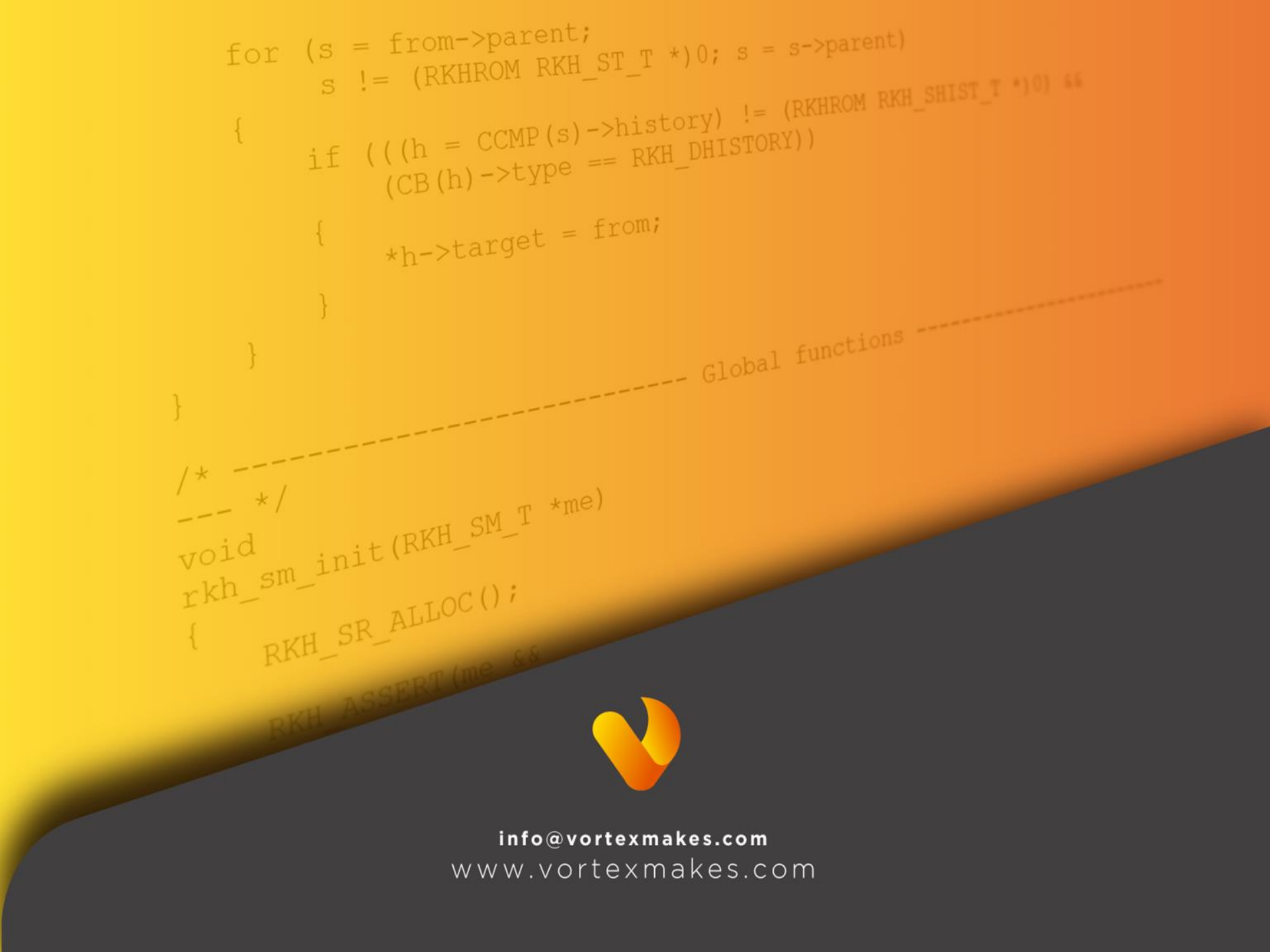


OV4: Validación

OV4: Si durante la cocción, el usuario abre la puerta y la vuelve a cerrar, el horno enciende el emisor durante el intervalo de tiempo prefijado.



?



info@vortexmakes.com
www.vortexmakes.com