OIL: OSEK Implementation Language

CPU		ATMEL_AT91SAM7S256 {	
	OS_TYPE	**OS;**	*1 instancia (obligatorio)*
	APPMODE_TYPE	**APPMODE;**	*Varias instancias (mínimo, 1)*
	TASK_TYPE	**TASK;**	*Varias instancias (mínimo, 1)*
	ISR_TYPE	**ISR;**	*Varias instancias*
	COUNTER_TYPE	**COUNTER;**	*Varias instancias*
	ALARM_TYPE	**ALARM;**	*Varias instancias*
	RESOURCE_TYPE	**RESOURCE;**	*Varias instancias*
	EVENT_TYPE	**EVENT;**	*Varias instancias*
		};	

Ejemplo:

```
CPU ATMEL_AT91SAM7S256
{
  OS LEJOS_OSEK
  {
    STATUS = EXTENDED;
    STARTUPHOOK = FALSE;
    ERRORHOOK = FALSE;
    SHUTDOWNHOOK = FALSE;
    PRETASKHOOK = FALSE;
    POSTTASKHOOK = FALSE;
    USEGETSERVICEID = FALSE;
    USEPARAMETERACCESS = FALSE;
    USERESSCHEDULER = FALSE;
  };

  /* Definition of application mode */
  APPMODE appmode1{};

  /* Definition of OSEK_Task_Background */
  TASK OSEK_Task_Background
  {
    AUTOSTART = TRUE
    {
      APPMODE = appmode1;
    };
    PRIORITY = 1; /* lowest priority */
    ACTIVATION = 1;
    SCHEDULE = FULL;
    STACKSIZE = 512;
  };
};
```

OS		**OSNAME {**	
	enum {STANDARD, EXTENDED}	**STATUS;**	*[= EXTENDED]*
	Boolean	**STARTUPHOOK;**	*[= FALSE]*
	Boolean	**ERRORHOOK;**	*[= FALSE]*
	Boolean	**SHUTDOWNHOOK;**	*[= FALSE]*
	Boolean	**PRETASKHOOK;**	*[= FALSE]*
	Boolean	**POSTTASKHOOK;**	*[= FALSE]*
	Boolean	**USEGETSERVICEID;**	*[= FALSE]*
	Boolean	**USEPARAMETERACCESS;**	*[= FALSE]*
	Boolean	**USERESSCHEDULER;**	*[= FALSE]*
		};	

- STATUS: Indica si el sistema admite tareas extendidas o sólo tareas estándar.
- STARTUPHOOK, ERRORHOOK, SHUTDOWNHOOK, PRETASKHOOK, POSTTASKHOOK: Indica si se utilizan rutinas «hook» tras la inicialización del sistema, tras la aparición de un error, en el proceso de apagado del sistema, antes de lanzar una nueva tarea o tras la ejecución de una tarea.
- USEGETSERVICEID, USEPARAMETERACCESS: Habilitan el acceso a las macros que permiten obtener el ID de servicio y con la información de contexto de la rutina «hook» de error.
- USERESSCHEDULER: Indica si el recurso RES_SCHEDULER se utiliza en el sistema.

Ejemplo:

```
OS sample_os {
      STATUS = EXTENDED;
      STARTUPHOOK = TRUE;
      ERRORHOOK = TRUE;
      SHUTDOWNHOOK = FALSE;
      PRETASKHOOK = FALSE;
      POSTTASKHOOK = FALSE;
      USEGETSERVICEID = TRUE;
      USEPARAMETERACCESS = TRUE;
      USERESSCHEDULER  = FALSE;
};
```

APPMODE		**APPMODENAME {**	
		};	

Ejemplo:

```
APPMODE appmode1{};
```

EVENT		**EVENTNAME {**	
	uint32 WITH_AUTO	**MASK;**	*[=AUTO]*
		};	

- MASK: Identificador interno del sistema para cada evento.

Ejemplos:

```
EVENT event1 {
      MASK = 7;
};

EVENT event2 {
      MASK = AUTO;
};
```

ISR		**ISRNAME {**	
	uint32 {1,2}	**CATEGORY;**	*[=NO_DEFAULT]*
	uint32	**PRIORITY;**	*[=NO_DEFAULT]*
	uint32	**ENTRY;**	*[=NO_DEFAULT]*
	RESOURCE_TYPE	**RESOURCE[];**	
		};	

- CATEGORY: Indica si la RTI es de tipo 1 (sin llamadas al SO ni replanificación) ó de tipo 2 (con llamadas al SO y con replanificación tras la ejecución de la RTI) .
- PRIORITY: Indica la prioridad de la RTI (0, es la menor; 15, la más alta).
- ENTRY: Número de Interrupción hardware asociada a la RTI.
- RESOURCE: Lista de recursos accesibles por la RTI.

Ejemplos:

```
ISR isr1 {
      CATEGORY = 1;
      PRIORITY = 8;
      ENTRY = 22;
};
```

TASK		**TASKNAME {**	
	boolean	**AUTOSTART;**	*[=NO_DEFAULT]*
	<SI AUTOSTART == TRUE >	APPMODE_TYPE	**APPMODE[];**
	uint32 {0..15}	**PRIORITY;**	*[=NO_DEFAULT]*
	uint32	**ACTIVATION;**	*[=NO_DEFAULT]*
	enum {NON, FULL}	**SCHEDULE;**	*[=NO_DEFAULT]*
	EVENT_TYPE	**EVENT[];**	
	RESOURCE_TYPE	**RESOURCE[];**	
	MESSAGE_TYPE	**MESSAGE[];**	
	uint32	**STACKSIZE;**	*[=1024]*
		};	

- AUTOSTART: Indica si la tarea se ejecuta (pasa a «lista») con la inicialización del sistema o no.
- PRIORITY: Indica la prioridad estática de la tarea. La prioridad más baja es 0, mientras que 15 es el nivel de prioridad más alto.
- ACTIVATION: Indica el número máximo de activaciones simultáneas de la tarea.
- SCHEDULE: Indica si la tarea es apropiativa o no.
- EVENT: Lista de eventos a los que puede responder la tarea extendida.
- RESOURCE: Lista de recursos que pueden ser accedidos por la tarea.
- MESSAGE: Lista de mensajes a los que puede acceder la tarea.

- STACKSIZE: Tamaño de la pila

Ejemplos:

```
TASK task1 {
      AUTOSTART = FALSE;
      PRIORITY = 10;
      ACTIVATION = 3;
      SCHEDULE = FULL;
      EVENT = event1;
      EVENT = event2;
      RESOURCE = resource1;
      STACKSIZE = 512;
};

TASK task2 {
      AUTOSTART = TRUE {
            APPMODE = appmode1;
      };
      PRIORITY = 5;
      ACTIVATION = 1;
      SCHEDULE = NON;
      EVENT = event1;
      STACKSIZE = 256;
};
```

COUNTER		**TASKNAME {**	
	uint32	**MINCYCLE;**	*[=NO_DEFAULT]*
	uint32	**MAXALLOWEDVALUE;**	*[=NO_DEFAULT]*
	uint32	**TICKSPERBASE;**	*[=NO_DEFAULT]*
		};	

- MINCYCLE: Mínimo valor permitido de ticks del contador para cualquier alarma que se conecte al contador.
- MAXALLOWEDVALUE: Máximo valor permitido para el contador.
- TICKSPERBASE: Número de ticks de contador requeridos para incrementar en una unidad el contador. (1 tick es 1 ms.)

Ejemplos:

```
COUNTER cnt1 {
      MINCYCLE = 1;
      MAXALLOWEDVALUE = 1000;
      TICKSPERBASE = 1;
};
```

ALARM			**TASKNAME {**		
	COUNTER_TYPE		**COUNTER;**		*[=NO_DEFAULT]*
	enum {ACTIVATETASK,SETEVENT,ALARMCALLBACK}		**ACTION;**		
	<SI ACTION == ACTIVATETASK >				
		TASK_TYPE		**TASK[];**	
	<SI ACTION==SETEVENT>				
		TASK_TYPE		**TASK[];**	
		EVENT_TYPE		**EVENT[];**	
	<SI ACTION==ALARMCALLBACK>				
		STRING		**ALARMCALLBACKNAME;**	
	boolean		**AUTOSTART;**		
	<SI AUTOSTART == TRUE >				
		uint32		**ALARMTIME;**	
		uint32		**CYCLETIME;**	
		APPMODE_TYPE		**APPMODE[];**	
			};		

- COUNTER: Contador al que está asociada la alarma.
- ACTION: Acción que se realizará cuando se alcance el valor indicado por el contador.
- ALARMCALLBACK: Define el nombre de la rutina «callback» que se llamará cuando el contador asociado llegue al valor límite. (Si ACTION==ALARMCALLBACK)
- AUTOSTART: Indica si la alarma se activa automáticamente al inicio del sistema.
- ALARMTIME: El instante temporal en el que la alarma se debe disparar la primera vez. (Si AUTOSTART==TRUE)
- CYCLETIME: El periodo de una alama cíclica. (Si AUTOSTART==TRUE)

Ejemplos:

```
ALARM alarm1 {
      COUNTER = 1000;
      ACTION = ACTIVATETASK {
            TASK = task1;
      };
      AUTOSTART = FALSE;
};

ALARM alarm2 {
      COUNTER = 1000;
      ACTION = SETEVENT {
            TASK = task2;
            EVENT = event1;
      };
      AUTOSTART = TRUE {
            ALARMTIME = 500;
            CYCLETIME = 1000;
            APPMODE = appmode1;
      };
};
```

RESOURCE		**RESOURCENAME {**		
	enum {STANDARD,LINKED,INTERNAL}	**RESOURCEPROPERTY;**		*[=NO_DEFAULT]*
	<SI ACTION==LINKED>			
		RESOURCE_TYPE	**LINKEDRESOURCE;**	
		};		

- RESOURCEPROPERTY: Puede tomar uno de los tres siguientes valores:
  - STANDARD: Un recurso normal que no está enlazado a ningún otro recurso y no es un recurso interno.
  - LINKED: Un recurso que está enlazado con otro recurso STANDARD o LINKED.
  - INTERNAL: Un recurso interno que no puede accederse por la aplicación.
- LINKEDRESOURCE: Indicación de a qué recurso está enlazado el recurso actual. (Si RESOURCEPROPERTY=LINKED)

Ejemplos:

```
RESOURCE resource1 {
      RESOURCEPROPERTY = STANDARD;
};

RESOURCE resource2 {
      RESOURCEPROPERTY = INTERNAL;
};

RESOURCE resource3 {
      RESOURCEPROPERTY = LINKED {
            LINKEDRESOURCE = resource1;
      };
};
```