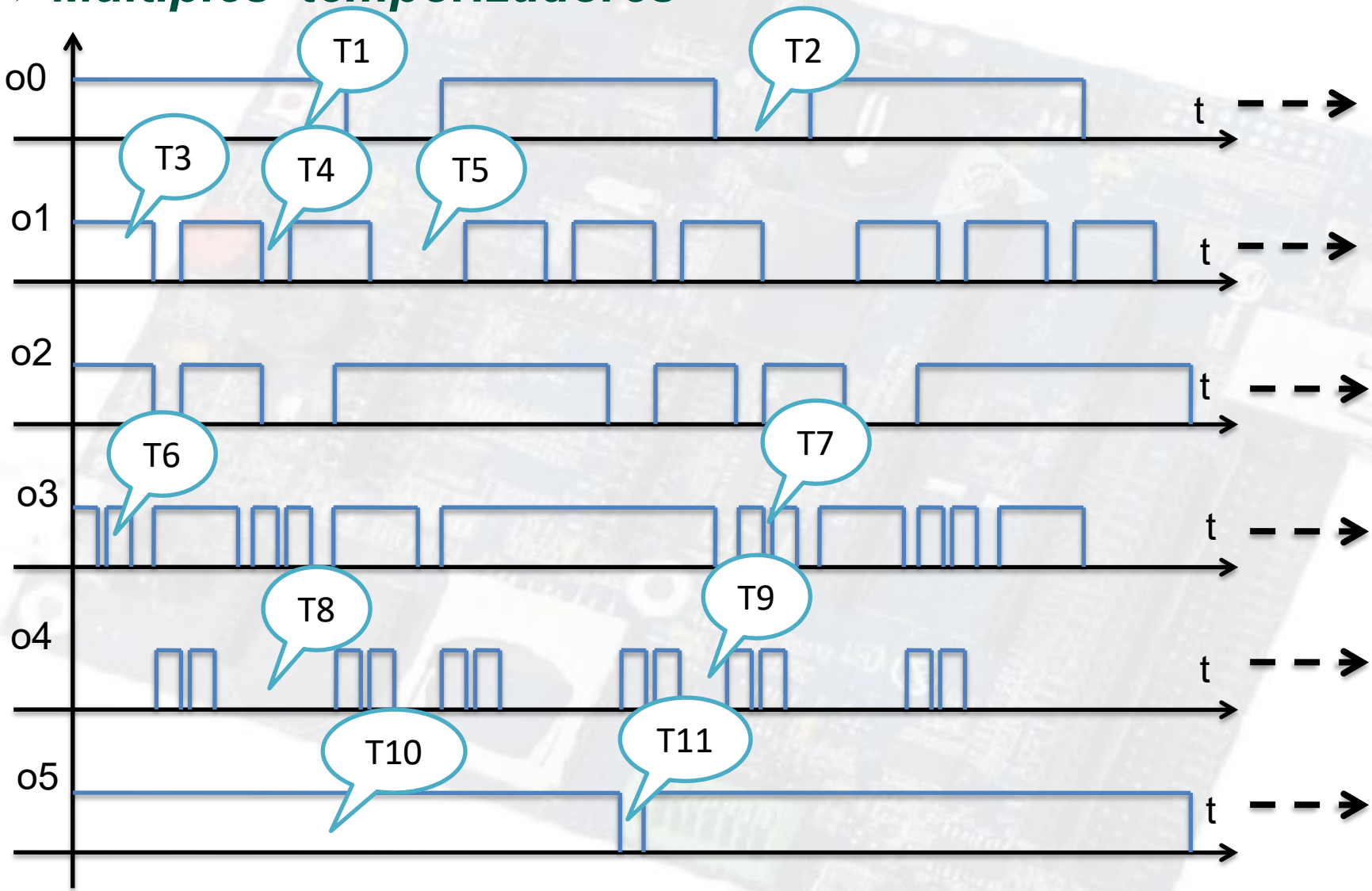
UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Bases de Tiempo Multiples

Ing. Marcelo Trujillo
Profesor Asociado

➢ *Multiples temporizadores*



➢ *Multiples temporizadores*

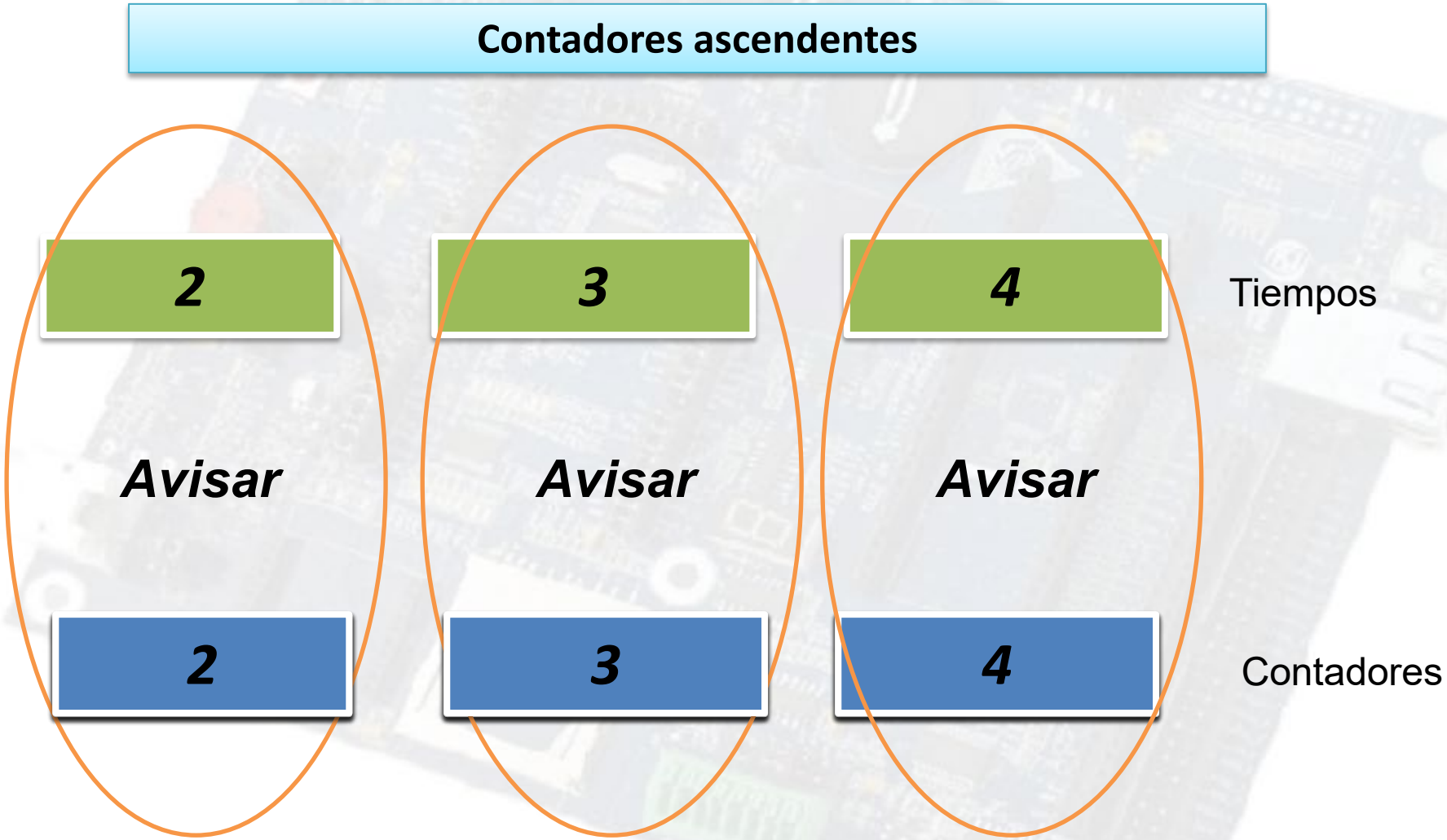
➢ *Las aplicaciones suelen tener*

- ✓Múltiples temporizadores (en nuestro ejemplo T1 a T 11)
- ✓En general con diferente orden de magnitud
- ✓Con valores de ms, s, m, h en el mismo ciclo de control

➢ *La solución será…..*

- ✓Base de tiempo única : Ticks
- ✓Utilización de variables como contadores de Ticks
- ✓Desarrollo de familia de funciones que nos hagan la tarea amigable

➢ *Utilización de variables como contadores de Ticks*



➢ *Utilización de variables como contadores de Ticks*

Si los contadores son descendentes el tope será único: Cero

➢ *Multiples temporizadores descendents*
Análisis de un conflicto

```
uint8_t Tmr_Run1, Tmr_Run2
```

Hay que avisar

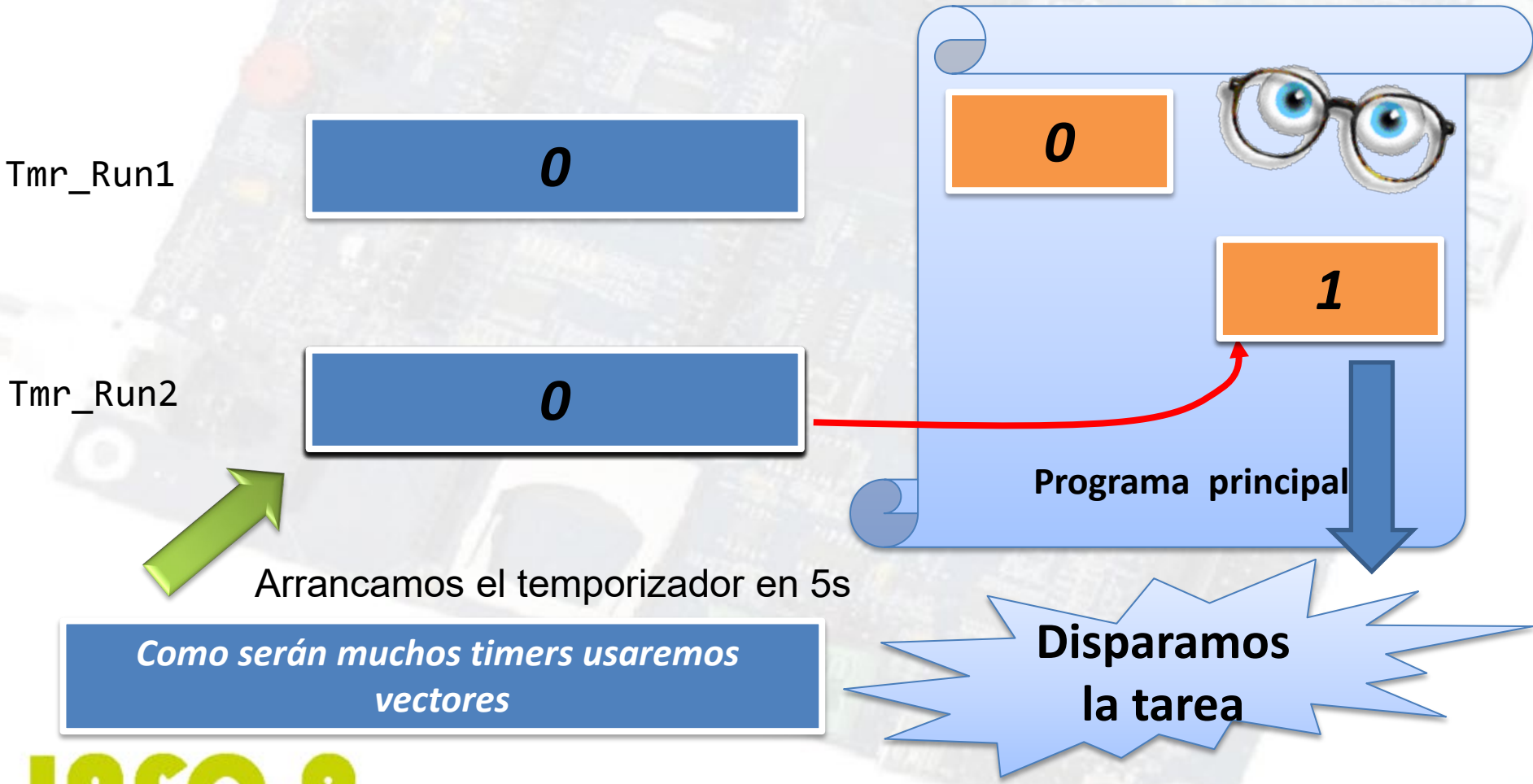
`Tmr_Run1` — **0** => Nunca arranco

`Tmr_Run2` — **0** => Terminó

Arrancamos el temporizador en 5s

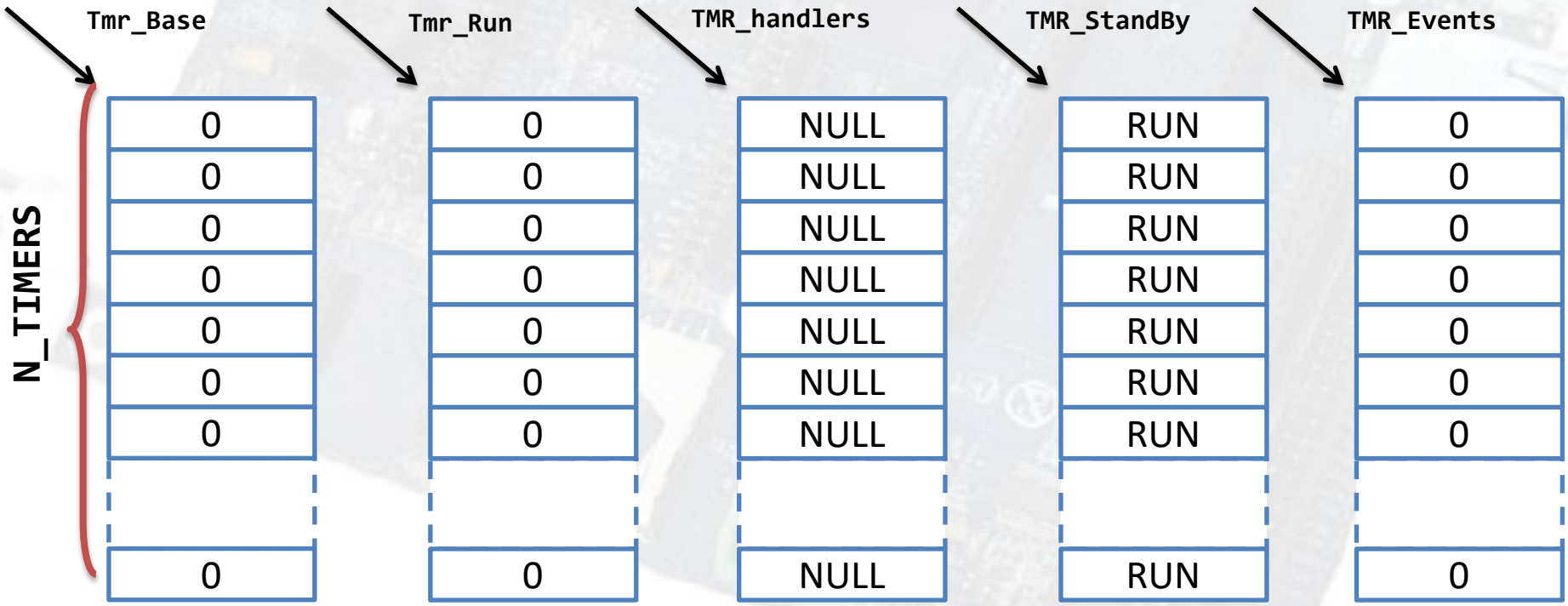
➢ *Multiples temporizadores descendents*

```
uint8_t Tmr_Run1, Tmr_Run2  TMR_Events0, TMR_Events1
```



➢ Buffers

```
volatile    uint32_t Tmr_Run[ N_TIMERS ];
volatile    uint8_t  TMR_Events[ N_TIMERS ];
void                 (* TMR_handlers [N_TIMERS]) (void);
volatile    uint8_t TMR_StandBy[ N_TIMERS ];
volatile    uint8_t Tmr_Base[ N_TIMERS ];
```



➢ *Funciones Primitivas* - TimerStart

```
typedef void (*Timer_Handler)(void);


/**
    \fn           void TimerStart( uint8_t event, timer_t t, void (*handler)(void) ,
    uint8_t )
    \brief        Inicia un timer
    \details      Inicia el timer llamando a la función apuntada handler con numero de
    \details      evento event durante el tiempo especificado por t según la base de
    \details      tiempo base
    \param [in]   event Numero de evento entre 0 y N_TIMERS
    \param [in]   t Tiempo del evento. Dependiente de la base de tiempos
    \param [in]   handler Callback del evento
    \param [in]   base DEC , SEG , MIN o HOR
    \return       void
*/
void TimerStart(uint8_t event, uint32_t time, Timer_Handler handler ,uint8_t base );
```

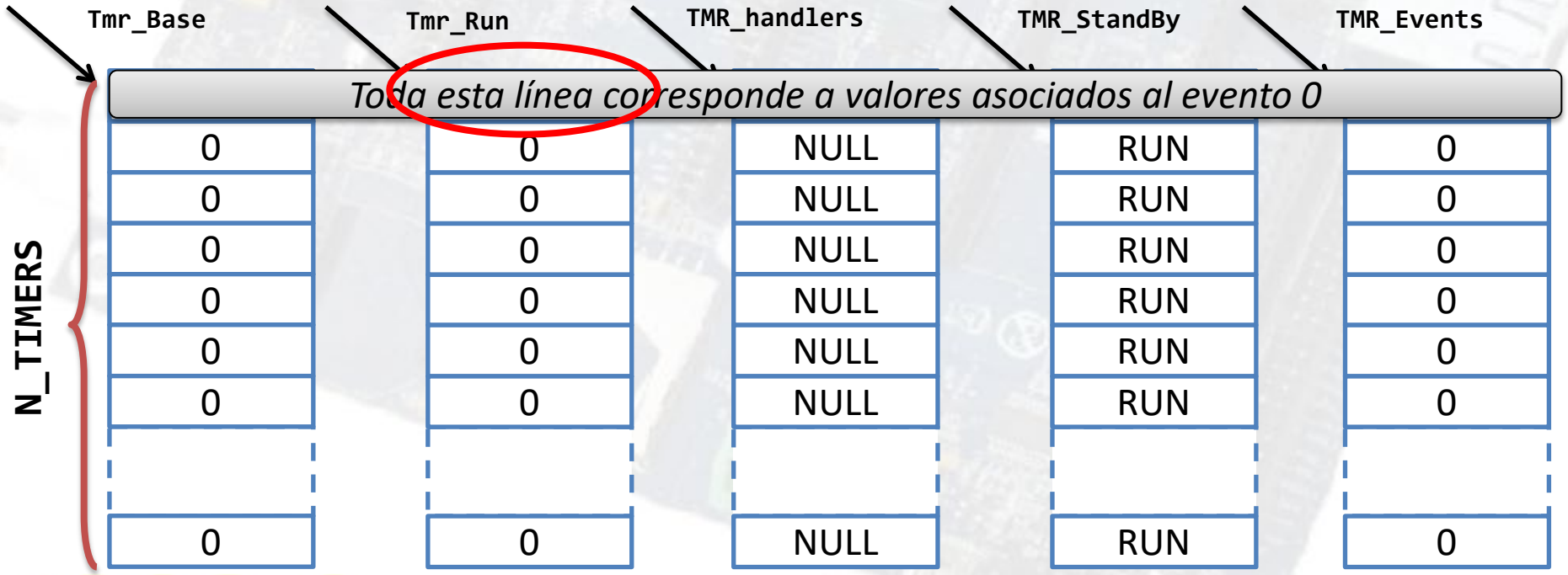
➢ *Analicemos con un ejemplo*

```
void EV_Alternar ( void );

TimerStart( 0, 10 , EV_Alternar , SEG );

void EV_Alternar ( void )
{
        //!< Aqui hacemos la tarea
}
```

Ticks = 2,5 ms

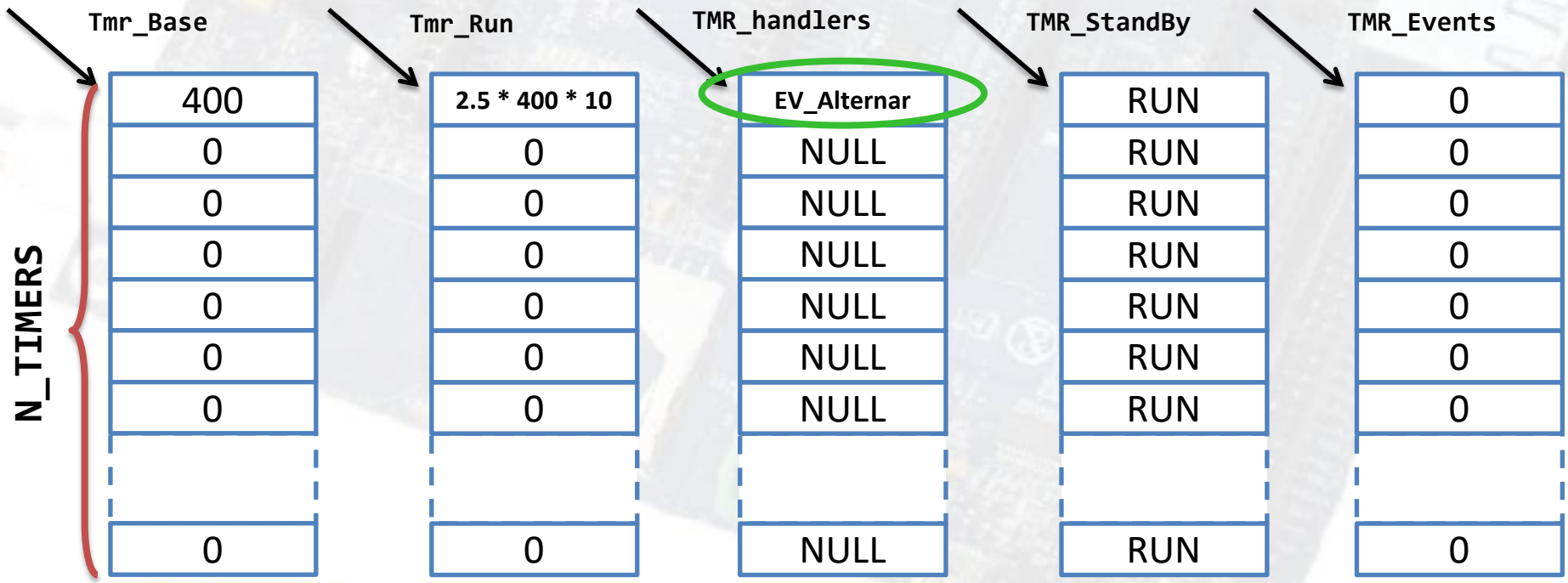
2,5 * 400 * 10

1 s ***10 s***



➢ *Analicemos con un ejemplo*

```
void EV_Alternar ( void );

TimerStart( 0, 10 , EV_Alternar, SEG );

void EV_Alternar ( void )
{
        //!< Aqui hacemos la tarea
}
```



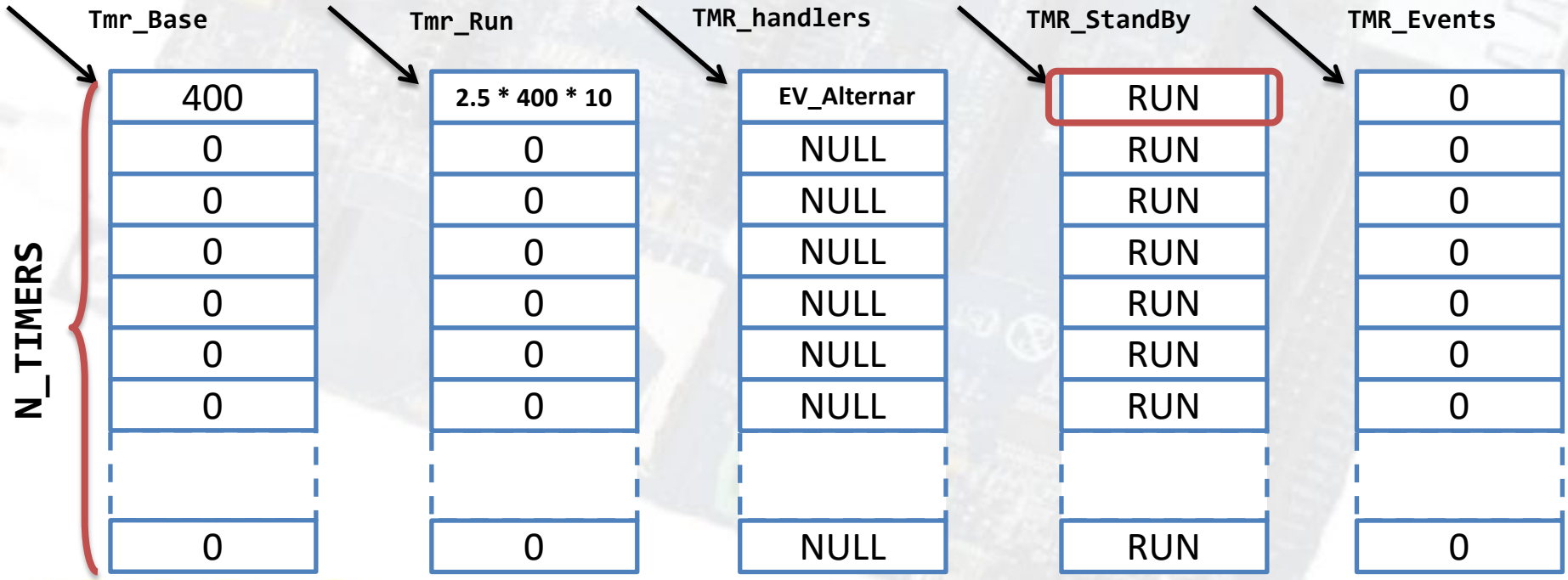
➢ *Analicemos con un ejemplo*

```
void EV_Alternar ( void );

TimerStart( 0, 10 , EV_Alternar , SEG );

void EV_Alternar ( void )
{
        //!< Aqui hacemos la tarea
}
```

StandByTimer (….)

1 (PAUSE)
Pone pausa

0 (RUN)
Temporiza



➢ *Analicemos con un ejemplo*

```
void EV_Alternar ( void );

TimerStart( 0, 10 , EV_Alternar , SEG );

void EV_Alternar ( void )
{
        //!< Aqui hacemos la tarea
}
```

Descuenta ticks → **AnalizarTimers ( )**

1
Finalización del tiempo

0
Timer detenido o corriendo

Analiza vencidos → **TimerEvent( )**

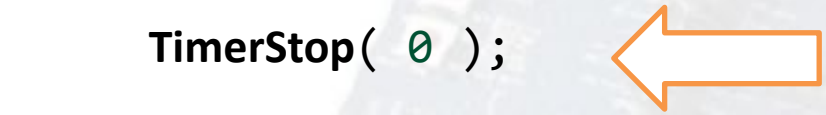
N_TIMERS	Tmr_Base	Tmr_Run	TMR_handlers	TMR_StandBy	TMR_Events
	400	2.5 * 400 * 10	EV_Alternar	RUN	0
	0	0	NULL	RUN	0
	0	0	NULL	RUN	0
	0	0	NULL	RUN	0
	0	0	NULL	RUN	0
	0	0	NULL	RUN	0
	0	0	NULL	RUN	0
	0	0	NULL	RUN	0

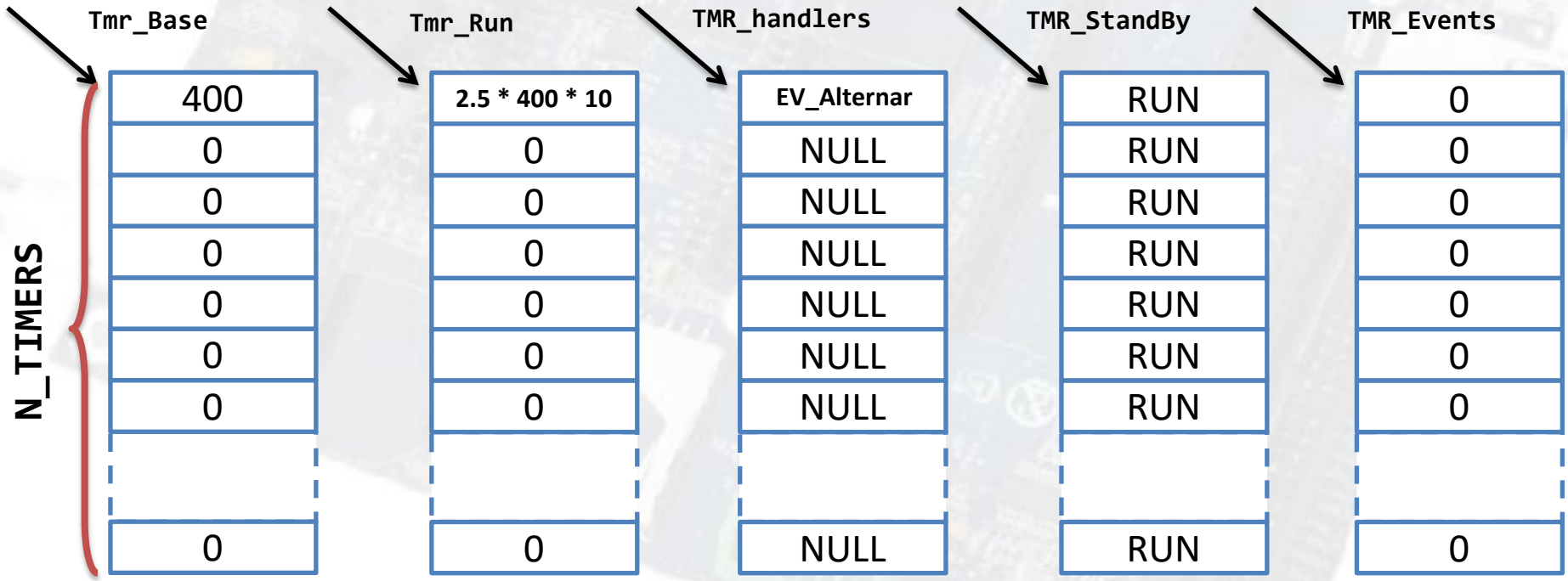
➢*Funciones* – TimerStop TimerClose

```
/**
    \fn void TimerStop( uint8_t e )
    \brief      Detiene un timer
    \details    Detiene el timer \a e
    \param [in] e Numero de evento entre 0 y N_TIMERS
    \return     void
*/
void TimerStop( uint8_t event );


/**
    \fn void TimerClose( void )
    \brief      Detiene los timers
    \details    Detiene todos los timers
    \return void
*/
void TimerClose( void );
```

➢ *Analicemos con un ejemplo*

```
TimerStop( 0 );
```

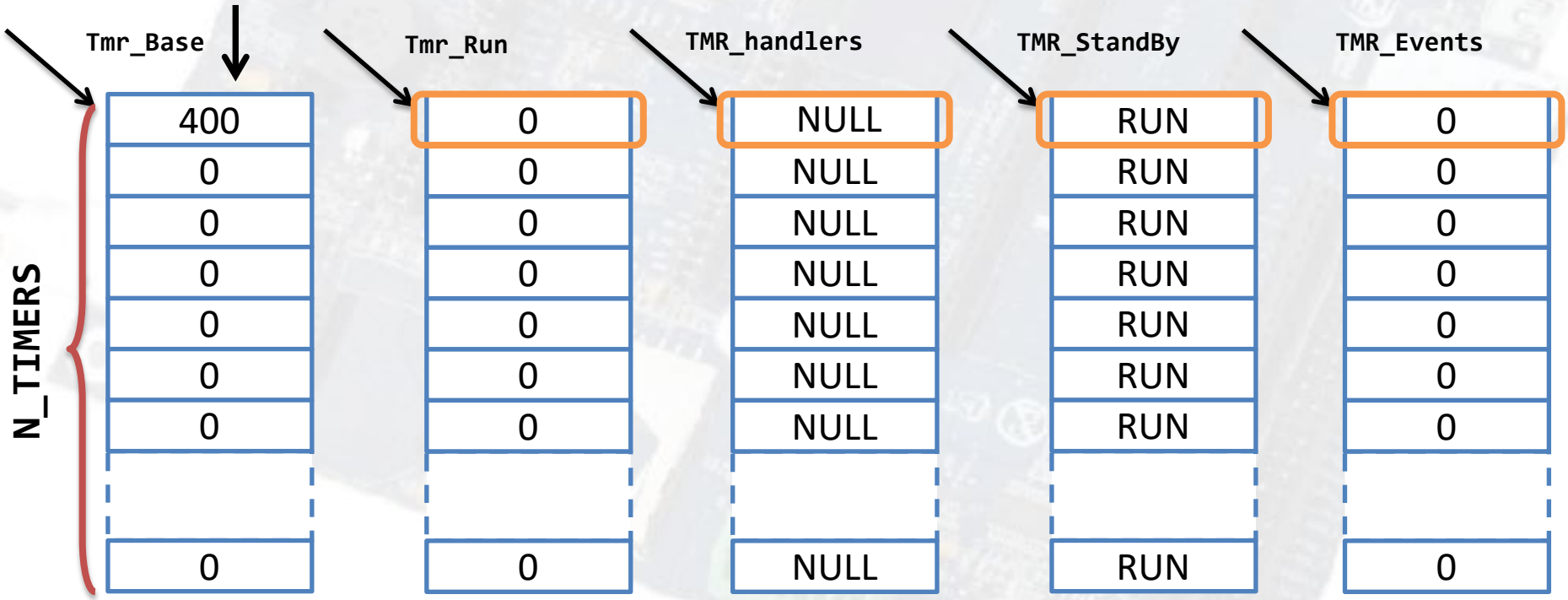




➢ *Analicemos con un ejemplo*

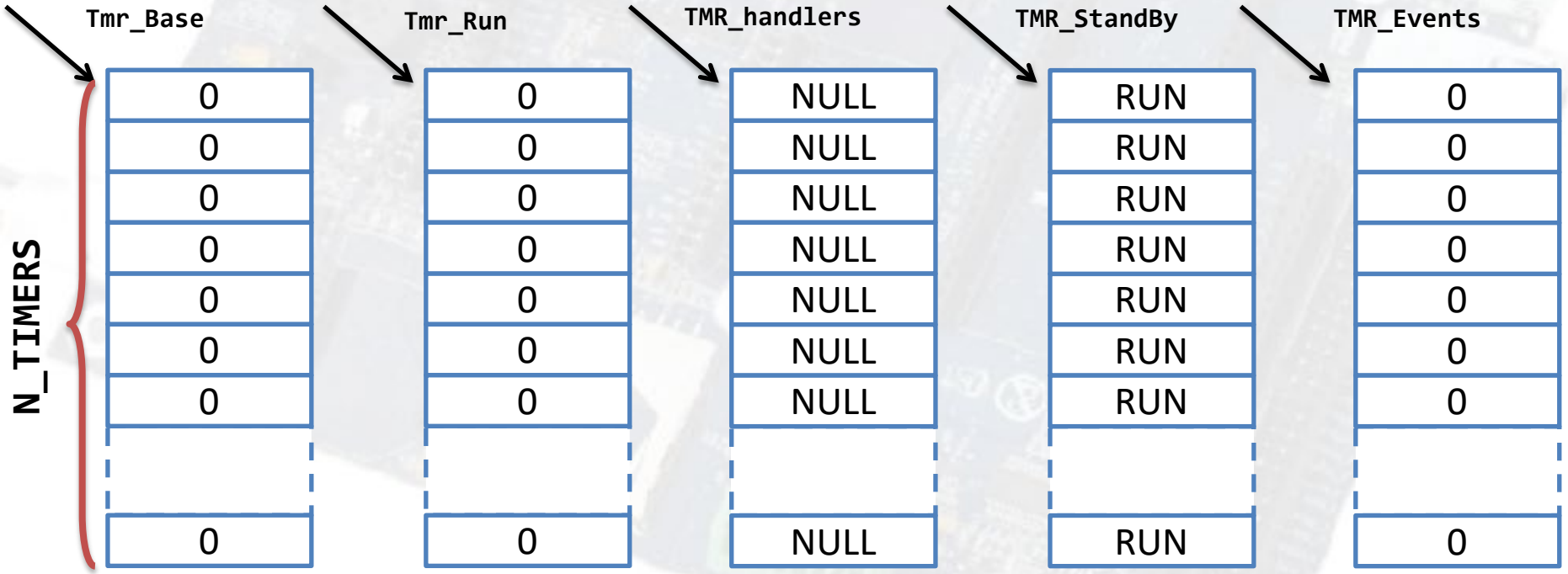
```
TimerStop( 0 );
```

Este valor no hace falta ponerlo en cero



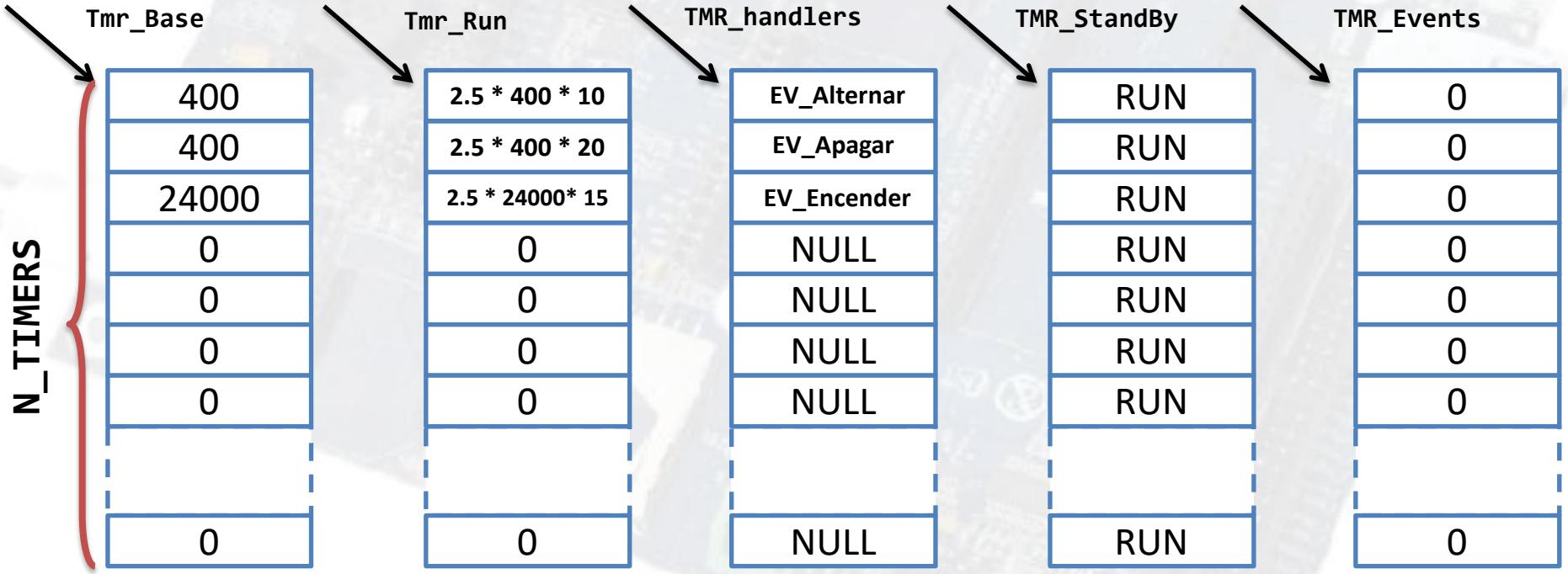
➢ *Analicemos con un ejemplo*

```
TimerStart( 0, 10 , EV_Alternar , SEG );
TimerStart( 0, 10 , EV_Apagar , SEG );
TimerStart( 0, 10 , EV_Encender , MIN);
```



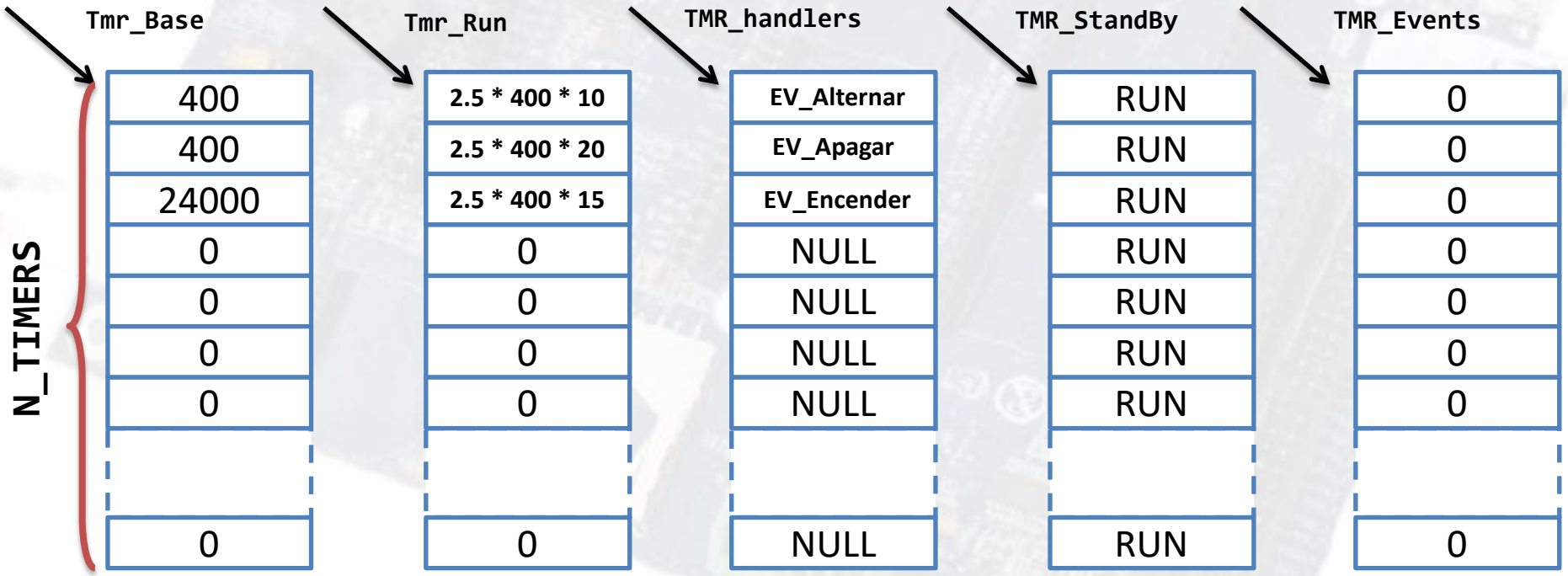
➢ *Analicemos con un ejemplo*

```
TimerStart( 0, 10 , EV_Alternar , SEG );
TimerStart( 1, 20, EV_Apagar , SEG );
TimerStart( 2, 15, EV_Encender , MIN);
```



➢ *Analicemos con un ejemplo*

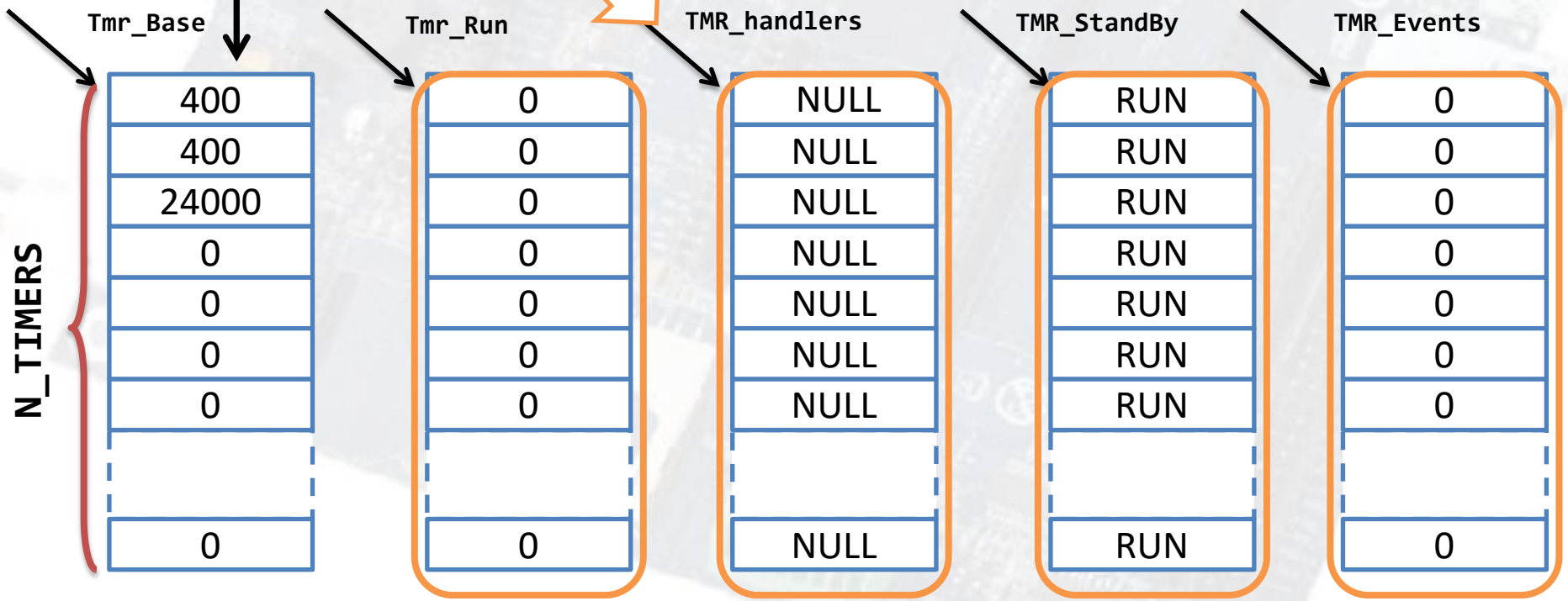
```
TimerClose( );
```



➢ *Analicemos con un ejemplo*

```
TimersClose( );
```

Este valor no hace falta ponerlo en cero



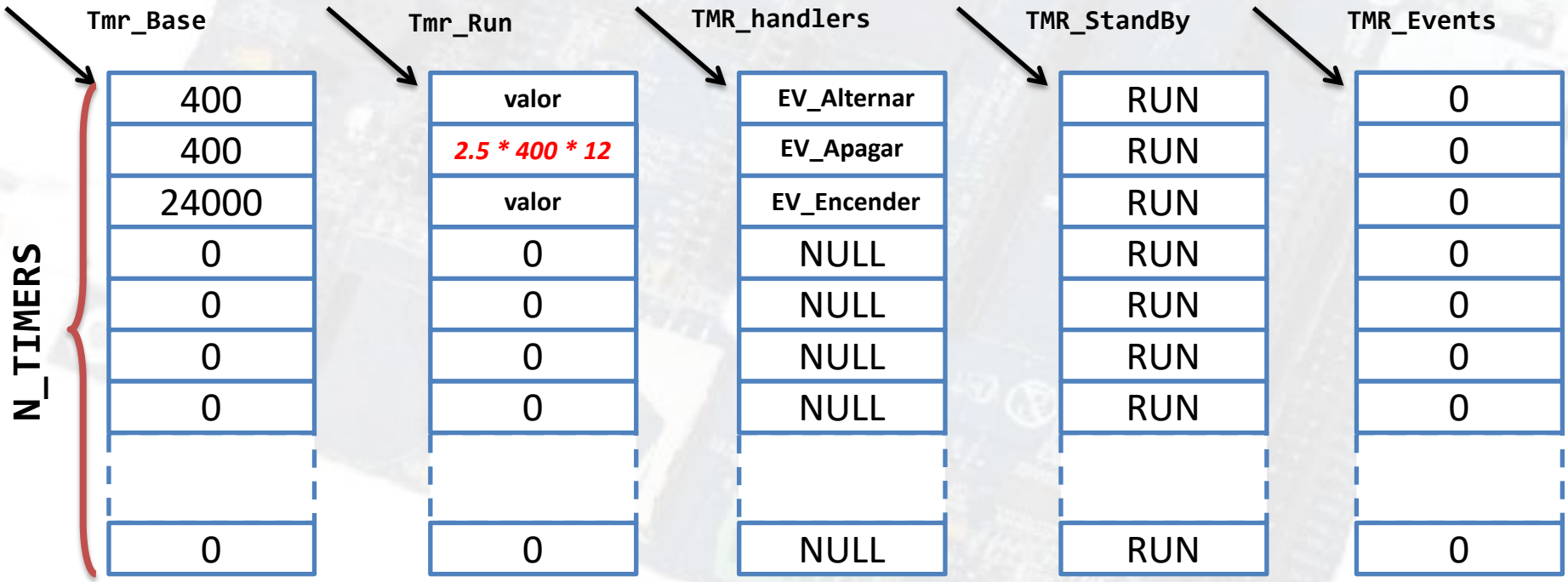
➢ *Funciones* – SetTimer y GetTimer

```
/**
    \fn void SetTimer( uint8_t event , uint32_t time )
    \brief      Inicia un timer
    \details    Reinicia el timer con el valor time (no lo resetea)
    \param [in] event Numero de evento entre 0 y N_TIMERS
    \param [in] time Tiempo del evento. Dependiente de la base de tiempos
    \return     void
*/
void SetTimer( uint8_t event , uint32_t time );

/**
    \fn GetTimer( uint8_t event )
    \brief      Toma el valor al vuelo del timer en cuestión
    \details    Lee el timer
    \param [in] event Numero de evento entre 0 y N_TIMERS
    \return     valor del timer
*/
uint32_t GetTimer( uint8_t event );
```

➢ *Analicemos con un ejemplo*

SetTimer(1 , 12);

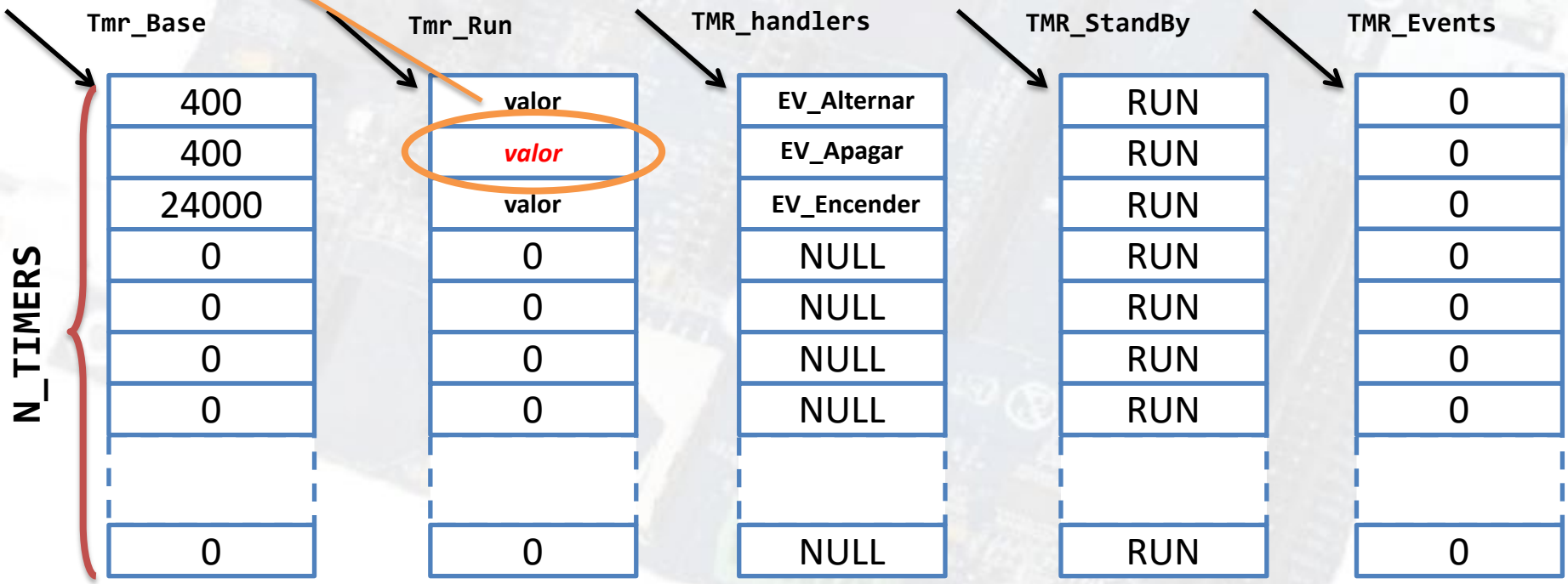
Cambia el tiempo estando en marcha



➢ *Analicemos con un ejemplo*

```
T = GetTimer( 1 );
```

Toma el valor actual del temporizador "al vuelo"



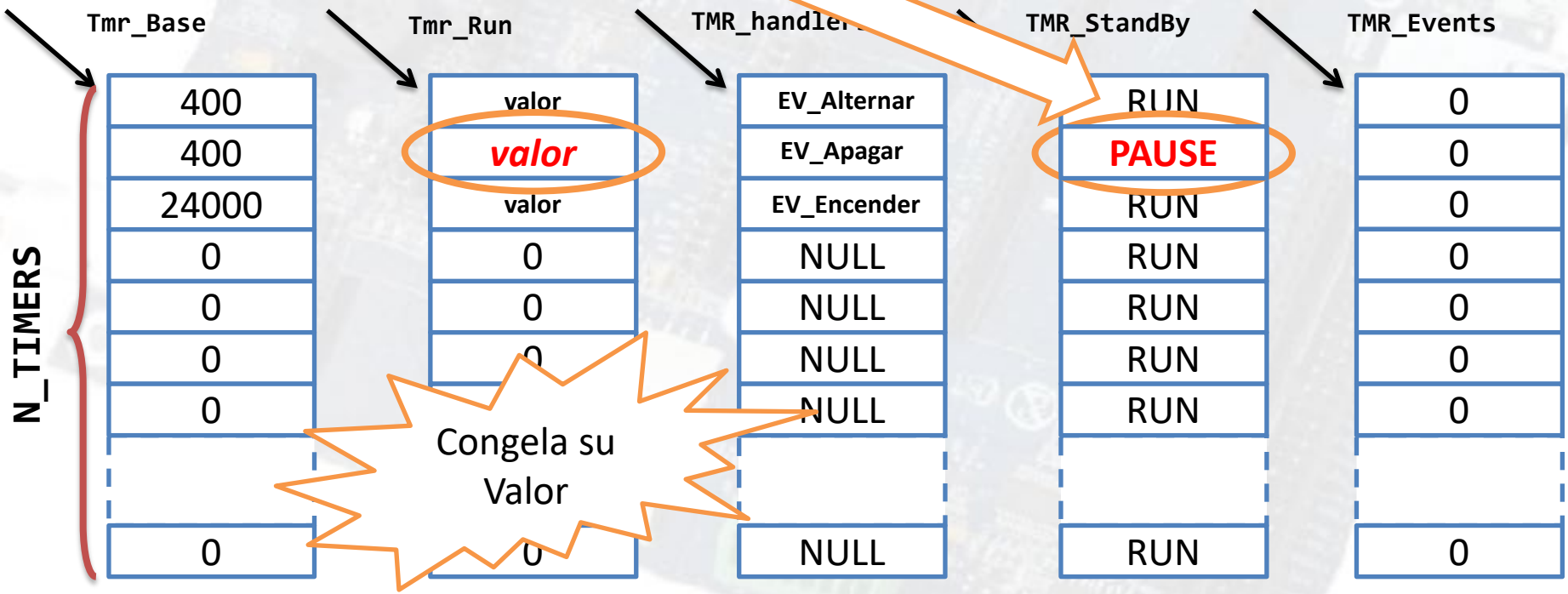
➢*Funciones* – StandByTimer

```
/**
    \fn StandByTimer( uint8_t event , uint8_t accion)
    \brief Detiene/Arranca el timer, NO lo resetea
    \details lo pone o lo saca de stand by
    \param [in] event Numero de evento entre 0 y N_TIMERS
    \param [in] accion RUN lo arranca, PAUSE lo pone en stand by
    \return     valor del timer
*/
uint32_t StandByTimer( uint8_t event , uint8_t accion)
```

➢ *Analicemos con un ejemplo*

Coloca una pausa
al Temporizador

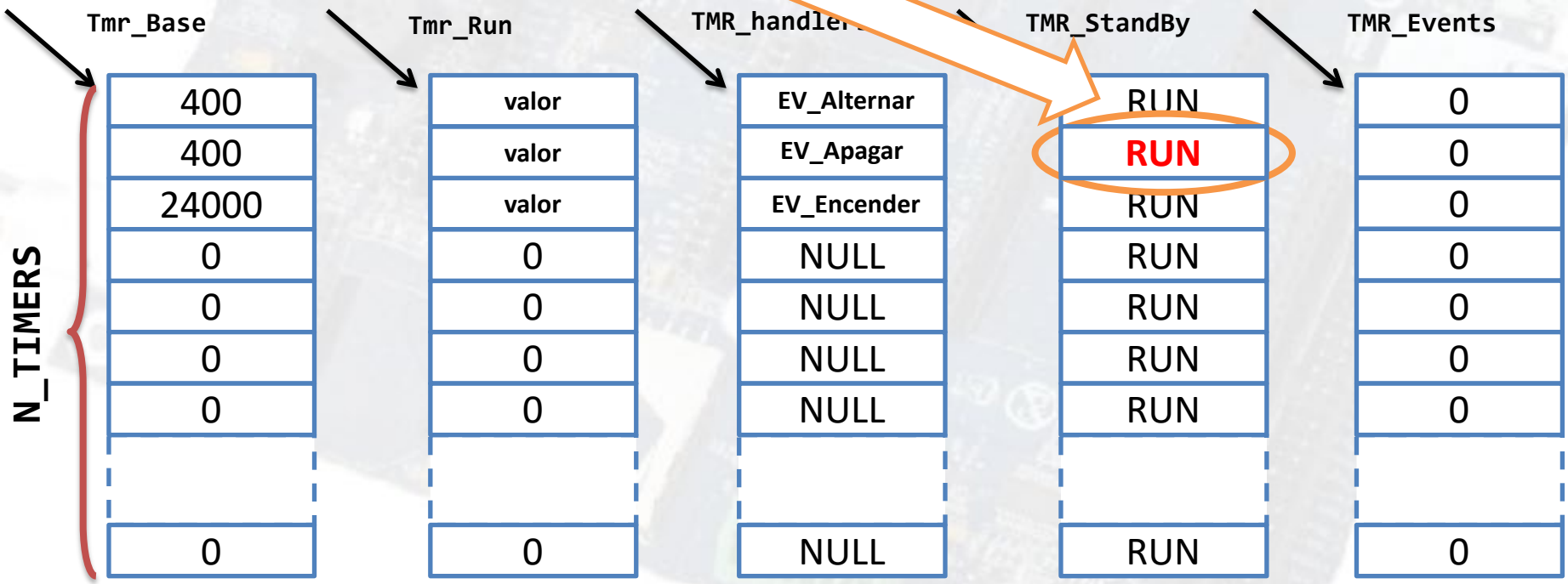
```
StandByTimer( 1 , PAUSE );
```



➢ *Analicemos con un ejemplo*

Arranca nuevamente

```
StandByTimer( 1 , RUN );
```

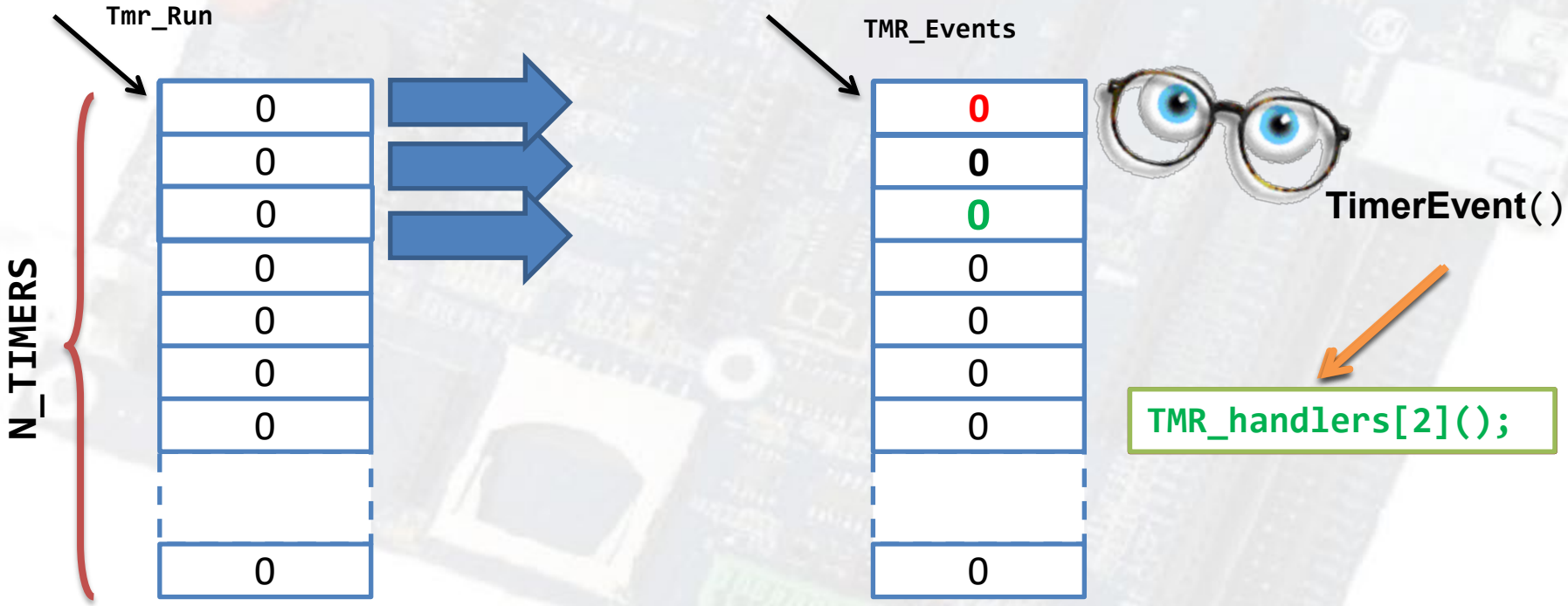


➢*Funciones Drivers* – AnalizarTimers y TimerEvent

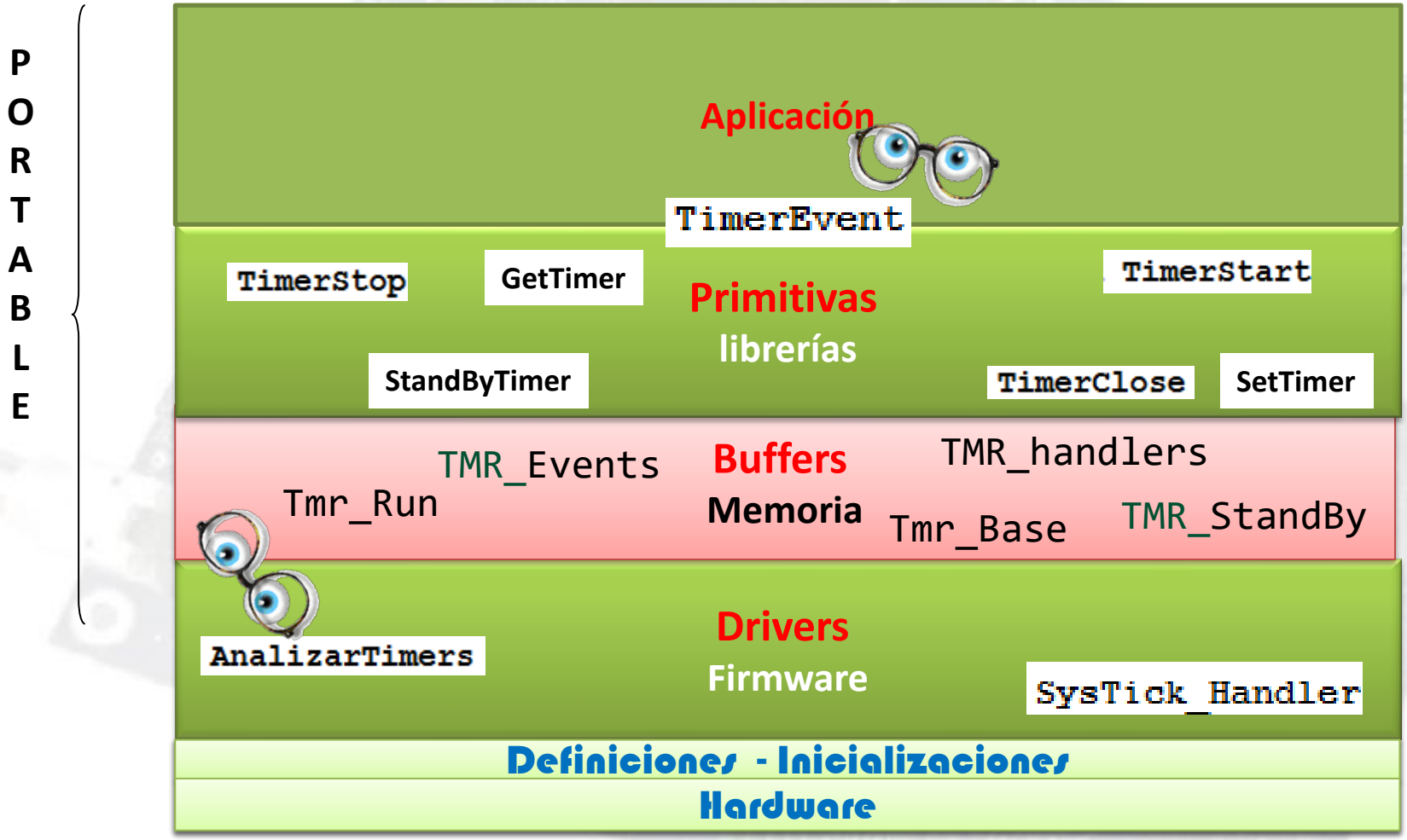
```
/**
    \fn void AnalizarTimers( void )
    \brief      Decremento periódico de los contadores
    \details    Decrementa los contadores de los timers en ejecución.
    \details    Debe ser llamada periódicamente con la base de tiempos
    \return     void
*/
void AnalizarTimers ( void )
/**
    \fn void TimerEvent( void )
    \brief      Chequeo de timers vencidos
    \details    Llama a los callbacks de los timers vencidos. Debe
    \details    llamarse desde el lazo principal del programa
    \return     void
*/
void TimerEvent(void)
```

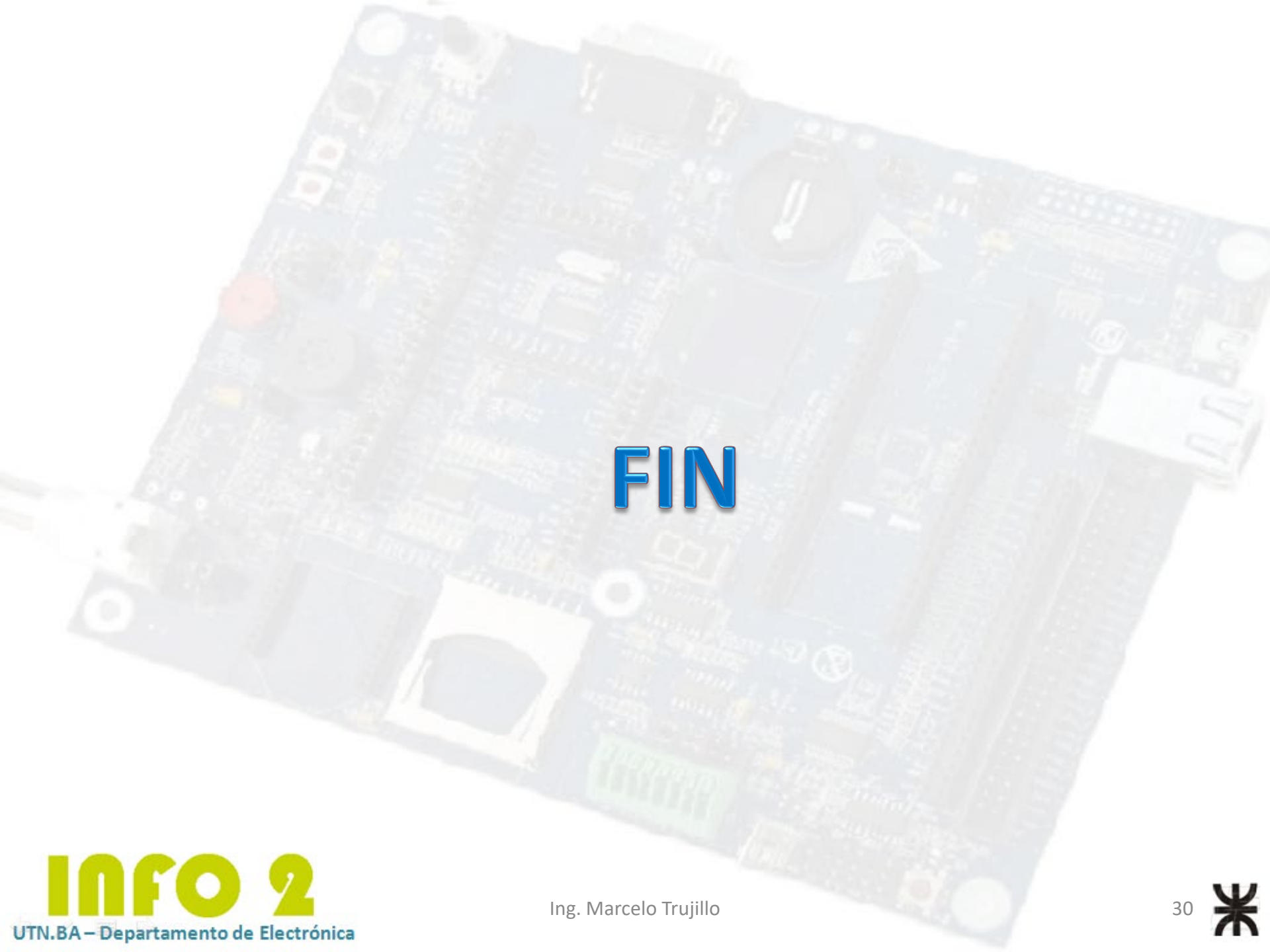
➢ *Analicemos con un ejemplo*

AnalizarTimers ( )



➢ Ubicación en las capas




FIN