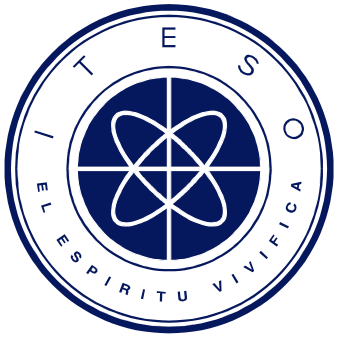
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación el 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

MAESTRÍA EN DISEÑO ELECTRÓNICO



REPORTE DE FORMACIÓN COMPLEMENTARIA EN ÁREA DE CONCENTRACIÓN EN SISTEMAS EMBEBIDOS Y TELECOMUNICACIONES

Trabajo recepcional que para obtener el grado de

MAESTRO EN DISEÑO ELECTRÓNICO

Presentan: Juan Esteban Bernardo Aldana

Asesor: Dr. José Luis Chávez Hurtado

San Pedro Tlaquepaque, Jalisco. mayo de 2018.

Contenido

Instituto Tecnológico y de Estudios Superiores de Occidente i

1. Resumen de los proyectos realizados 3

1.1. DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA OPERATIVO PARA SISTEMAS EMBEBIDOS 3
1.1.1 Introducción 3
1.1.1 Antecedentes 3
1.1.2 Solución Desarrollada 3
1.1.3 Análisis de Resultados 4
1.1.4 Conclusiones 4
1.2. TRANSMISIÓN DE DATOS EN EQUIVALENTE PASA-BAJAS CON MODULACIÓN 4-PSK 5
1.2.1 Introducción 5
1.2.1 Antecedentes 5
1.2.2 Solución Desarrollada 6
1.2.1 Análisis de Resultados 6
1.2.2 Conclusiones 7
1.3. K64 AUDIO PLAYER V2.0 Y AMPLIFICADOR DE AUDIO 7
1.3.1 Introducción 7
1.3.1 Antecedentes 7
1.3.2 Solución Desarrollada 8
1.3.1 Análisis de Resultados 8
1.3.2 Conclusiones 9

2. Conclusiones 9

Apéndices 11

A. REPORTE DE DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA OPERATIVO PARA SISTEMAS EMBEBIDOS 12
B. REPORTE DE TRANSMISIÓN DE DATOS EN EQUIVALENTE PASA-BAJAS CON MODULACIÓN 4-PSK 95
C. REPORTE DE K64 AUDIO PLAYER V2.0 Y AMPLIFICADOR DE AUDIO 115

Introducción

En el presente reporte se dan a conocer los proyectos que más aportaron en mi desarrollo profesional al estudiar la Maestría en Diseño Electrónico.

Al estar involucrado profesionalmente en la industria automotriz, he decidido enfocar mi preparación en el área de sistemas embebidos y telecomunicaciones. Esto me ha permitido actualizar mis conocimientos con nuevas herramientas y técnicas que contribuyen de manera significativa en mi desempeño diario.

Si bien todos los temas vistos en el transcurso de la maestría han sido un importante apoyo en mi carrera profesional, a continuación se enlistan los 3 proyectos que considero mayor impacto han tenido y que me han sido de mayor utilidad, ya sea en mis actividades y responsabilidades diarias así como apoyo en la resolución de temas complejos.

- Diseño de Sistemas Operativos en Ambientes Embebidos: Diseño e implementación de un sistema operativo para sistemas embebidos
- Introducción a las Telecomunicaciones Digitales y Analógicas: Transmisión de datos en equivalente pasa-bajas con modulación 4-PSK
- Taller de Diseño de Tarjetas de Circuito Impreso: K64 Audio Player V2.0 y Amplificador de audio

1. Resumen de los proyectos realizados

A continuación se presenta una vista general de los proyectos a reportar en el presente documento. Los reportes de los mismos serán incluidos al final de esta sección.

1.1. Diseño e implementación de un sistema operativo para sistemas embebidos

1.1.1 Introducción

De manera general, el proyecto tiene por objetivo desarrollar un sistema operativo que pueda ser utilizado en ambientes embebidos, portable y con tareas ejecutables de manera concurrente. Todo esto siguiendo los estándares de la industria automotriz como lo es AUTOSAR.

1.1.1 Antecedentes

La electrónica de consumo de nuestros días, demanda precisión, velocidad y eficiencia en la administración de recursos que las aplicaciones tienen a su disposición. Un elemento que contribuye de manera importante en el cumplimiento de estas demandas es el microcontrolador que forma parte de estas aplicaciones. A su vez, la implementación de sistemas operativos en un microcontrolador, contribuye enormemente, aportando la gestión de los recursos disponibles de manera eficiente y precisa. De ahí, la importancia de haber adquirido el conocimiento y técnicas existentes en el diseño de sistemas operativos más utilizados en la industria.

1.1.2 Solución Desarrollada

Para el desarrollo del proyecto, se utilizó el microcontrolador de la familia HCS12 de Freescale MC9S12XEP100. Se aprovecharon los recursos disponibles del MCU para que su reloj de operación fuera configurable. El scheduler fue diseñado de una manera que fuera portable en

diferentes plataformas, basado en el concepto de offset y enmascaramiento estudiados en clase. Para la administración de tareas se consideraron tres estados básicos: Ready, Suspended y Running. Se realizó un driver para el manejo de memoria y recursos de las tareas. Además se agregó un buffer para el manejo de prioridades de las mismas.

La implementación se desarrolló para soportar el mecanismo de cambio de contexto, así como interrupciones de categoría 2 para el soporte de eventos.

1.1.3 Análisis de Resultados

Se realizaron diferentes pruebas durante la etapa de desarrollo del proyecto de cada uno de los elementos que lo componen, con resultados positivos. Estos resultados y casos de prueba están documentados y se adjuntan en la sección de Apéndices junto al reporte del mismo proyecto.

1.1.4 Conclusiones

Aunque el proyecto fue relativamente sencillo de implementar, esto debido a que se desarrolló por etapas y conforme los temas se fueron estudiando, me parece que fue de gran utilidad conocer a detalle cada uno de los elementos que comprenden un sistema operativo para sistemas embebidos, ya que al hacer un comparativo con los sistemas utilizados en la industria automotriz, encuentro un gran parecido, lo que facilita la comprensión, implementación y análisis de nuevas funcionalidades siguiendo los estándares industriales.

1.2. Transmisión de datos en equivalente pasa-bajas con modulación 4-PSK

1.2.1 Introducción

El objetivo principal de este proyecto es realizar en la plataforma de simulación Matlab, un proceso de modulación digital pasa-baja 4-PSK representado por su equivalente pasa-bajas, considerando la adición de ruido en la transmisión, así como ser capaz de calcular la taza de error y mostrar el diagrama de estados tanto en el transmisor como en el receptor.

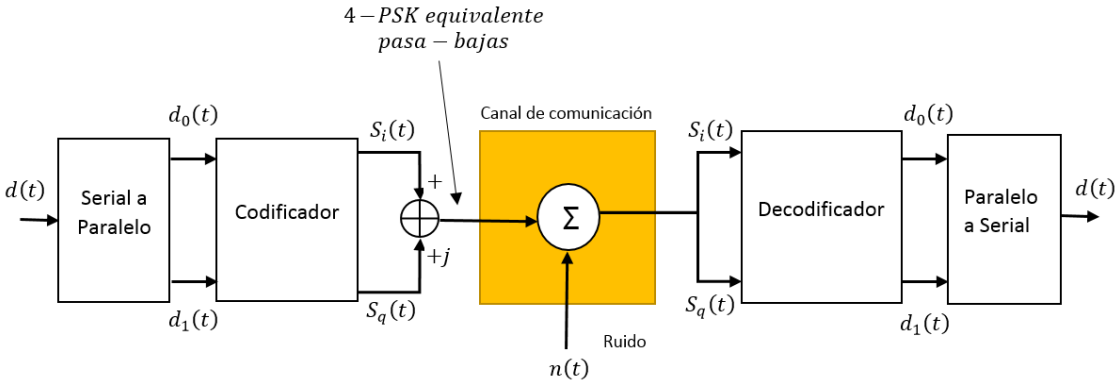


Figura 1-1 Diagrama a bloques del modulador digital pasa-baja 4-PSK

1.2.1 Antecedentes

La modulación es el proceso en el cual la información contenida en una señal (moduladora), es transferida a otra señal (portadora), cuyas características son más apropiadas para el medio de transmisión a utilizar. En la modulación 4-PSK o QPSK, se hace variar la fase de la señal portadora mediante la aplicación de una señal digital que contiene la información que se desea transmitir. Entre las ventajas que se obtienen con una comunicación de este tipo es que es posible duplicar la velocidad de transmisión sin requerir un mayor ancho de banda y sin incrementar significativamente la taza de error en la transmisión. Esto la hace ideal para muchas aplicaciones que tenemos hoy en día como la transmisión digital de video y protocolos de comunicación inalámbrica como ZigBee y Bluetooth.

1.2.2 Solución Desarrollada

En este proyecto se generó una señal digital compuesta de 1024 valores digitales binarios y producidos de manera aleatoria, con una velocidad de datos de 1000bps. De esta señal digital, se formaron pares secuenciales de bits para producir las 4 fases de la portadora mediante un codificador de fases. Una vez codificada la señal, se transmitió a través del canal, al cual se le adicionó una señal de ruido con distribución gaussiana normalizada y de amplitud variable.

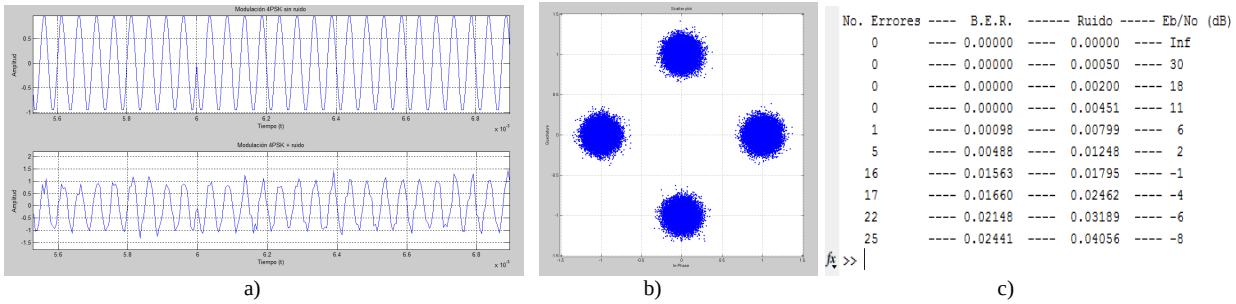
La señal digital recibida, se pasó por un filtro pasa-bajas para eliminar el ruido añadido en el canal de transmisión. Se aplican las propiedades ortogonales de la función senoidal y cosenoidal para obtener la señal en fase y cuadratura.

Una vez que la señal ha sido demodulada, se realiza un muestreo de 20 elementos a partir del punto medio de cada bit, para determinar si el mismo corresponde a un 0, 1 o -1, y posteriormente generar los pares de bits de acuerdo a su fase.

Finalmente se reordenan los pares de bits en una serie de 1024 bits, tal como se transmitieron originalmente y se realiza el cálculo de la taza de bits erróneos recibidos, así como su diagrama de fases según el nivel de ruido añadido en el canal de comunicación.

1.2.1 Análisis de Resultados

En los resultados, se observa de manera clara la influencia que el ruido del canal de transmisión tiene sobre la transmisión 4-PSK. También se observa que al filtrar la señal en el receptor para eliminar este ruido, no fue suficiente, y se tuvieron que buscar niveles óptimos en el comparador para codificar correctamente la señal recibida.



a) b) c)

Figura 1-2 a) Comparativo de la modulación sin ruido vs modulación con ruido.
b) Diagrama de fases con un nivel de ruido al 50% de la amplitud de la señal.
c) Taza de Error obtenida (BER) vs Eb/No expresada en dBs.

1.2.2 Conclusiones

Al resolver este proyecto, pude observar claramente los aspectos a cuidar durante la transmisión y recepción con la modulación 4-PSK y en general con cualquier otro tipo de modulación. La importancia en la potencia de transmisión, el canal y los efectos del ruido sobre la señal transmitida, la elección correcta de los filtros utilizados y sus coeficientes en la recepción, así como la selección adecuada del codificador para poder reconstruir la señal original.

Este ha sido desde mi punto de vista, un ejercicio muy bueno para entender los elementos básicos en las comunicaciones digitales y analógicas.

1.3. K64 Audio Player V2.0 y amplificador de audio

1.3.1 Introducción

En este curso, se realizaron dos proyectos con la finalidad de desarrollar las competencias necesarias, para el desarrollo profesional de un sistema electrónico. Estos proyectos son: una tarjeta basada en microcontrolador de 32 bits estilo Arduino con códec de audio (K64 Audio Player V2.0) y un amplificador de audio basado en el circuito TPA3125D2.

1.3.1 Antecedentes

La miniaturización de los componentes electrónicos, la necesidad de comunicar a velocidades cada vez más altas, las propiedades de los materiales utilizados hoy en día, así como el cuidado del medio ambiente, requieren de un desarrollo más eficiente y retador en las tarjetas de circuito impreso. La orientación, ubicación y la correcta interconexión de los componentes en el circuito impreso, ya no es suficiente al momento de realizar nuestro diseño. Aspectos como la velocidad de transmisión de datos y distancia entre componentes, aislamiento electromagnético, tipos de fuente de alimentación utilizados, interferencia electromagnética, buses de comunicación, tipo de empaquetado de componentes, acoplamiento de impedancias,

restricciones de manufactura y de ensamble, son algunos elementos a considerar en los circuitos impresos que requiere la industria de nuestros días.

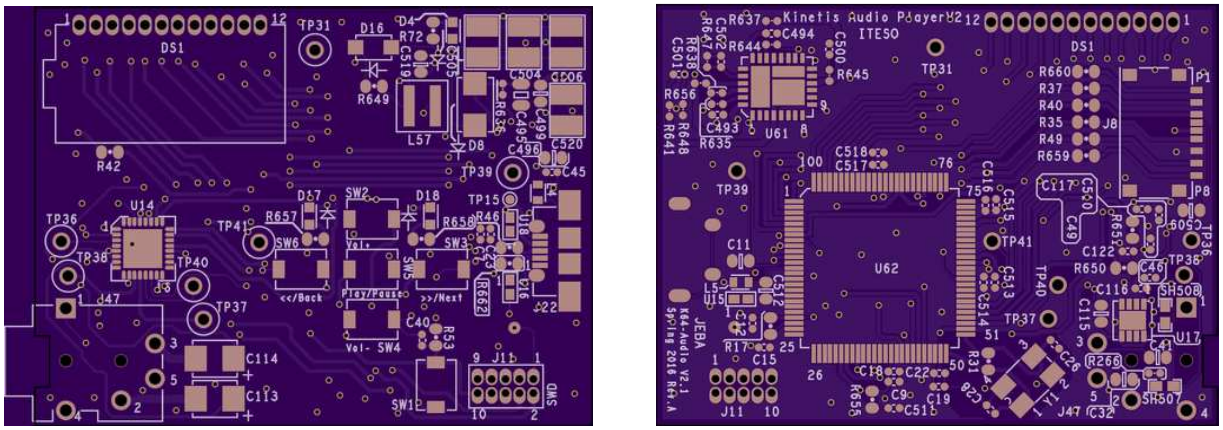
1.3.2 Solución Desarrollada

Para el desarrollo del PCB K64-Audio, se siguieron los requerimientos de diseño establecidos por el profesor durante el curso. Dichos requerimientos incluían aspectos mecánicos, identificación, fabricación, esquemático, eléctricos, colocación, ruteado y puntos de prueba entre otros. Con el desarrollo de este proyecto se trabajó en fortalecer las habilidades en el uso de la herramienta CAD, restricciones de diseño, consideraciones de ruteo, colocación, de ensamble y manufactura y documentos de fabricación.

Para desarrollar el amplificador de audio, se utilizó el circuito TPA3125D2, con el cual se trabajaron principalmente las competencias para el desarrollo de librerías, diseño y captura de esquemáticos. Este proyecto se fabricó con las herramientas existentes en el ITESO, por tal motivo se tuvieron restricciones específicas como no incluir planos de tierra y fuente de alimentación, dadas las características de la herramienta de fabricación.

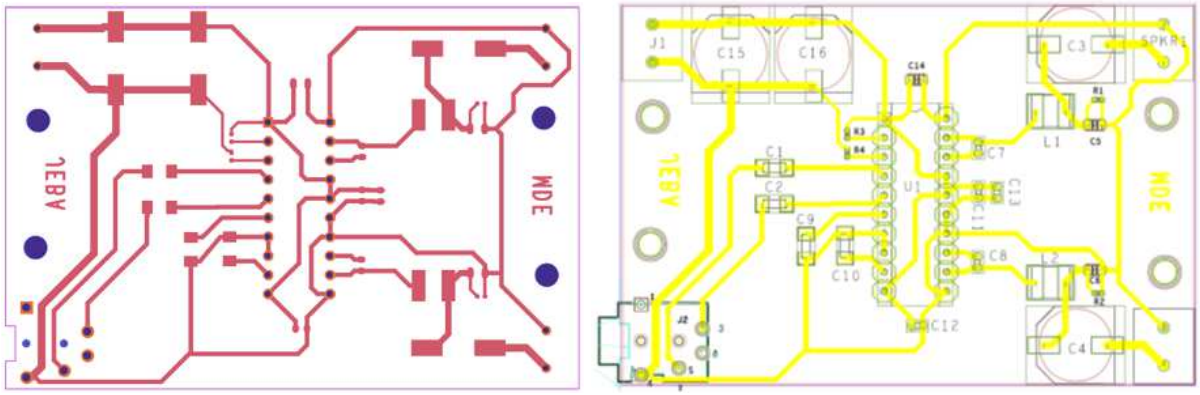
1.3.1 Análisis de Resultados

Los resultados de este proyecto están documentados con mayor detalle en la sección de Apéndices de este reporte.



TOP BOTTOM

Figure 2-1 K64 Audio Board



Archivos Gerber Ensamble Silkscreen

Figura 2-2 Amplificador de Audio diseño final

1.3.2 Conclusiones

Con el desarrollo de este proyecto, alcancé los conocimientos y habilidades necesarias para realizar un diseño profesional para el desarrollo de prototipos o producto final, cumpliendo con los estándares y calidad que la industria electrónica demanda. En general cumplió mis expectativas.

2. Conclusiones

Si bien los proyectos presentados en este reporte influyeron significativamente en mi carrera profesional, considero importante mencionar que los demás cursos de la maestría contribuyeron y facilitaron el aprendizaje y desarrollo de los proyectos aquí presentados.

Los temas estudiados, la experiencia transmitida de los profesores y alumnos, así como las metodologías utilizadas en el desarrollo de proyectos durante el estudio de la maestría en diseño electrónico, me brindan la confianza para seguir creciendo profesionalmente y superar cualquier reto, sin importar los obstáculos se presenten o el tipo de industria en la que me involucre, estos elementos siempre contribuirán de una manera importante en mi carrera para alcanzar las metas que me proponga.

Apéndices

A. REPORTE DE DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA OPERATIVO PARA SISTEMAS EMBEBIDOS

Operating Systems Design for Embedded Environments

Project:	**Implementación de un Sistema operativo para Sistemas embebidos**
Team:	**Team 4, Sergio, Miguel & Esteban**
Date:	**10/02/2014**
Comments:	**Released**

Table of Contents

Operating Systems Design for Embedded Environments 13

1. OS Tick SW Conceptual design 16

2 Function Description and Dynamic Behavior 16

2.1 FUNCTION *VOID MCU_INIT (VOID)* 16
2.2 FUNCTION *VOID MCU_INITCLOCK (VOID)* 17
2.3 FUNCTION *VOID MCU_DISTRIBUTEPLLCLOCK (VOID)* 17
2.4 FUNCTION *VOID GPT_INIT (VOID)* 17
2.5 FUNCTION *VOID GPT_STARTTIMER(GPT_CHANNELTYPE CHANNEL, GPT_VALUETYPE VALUE)* 17

2. Scheduler SW Conceptual design 19

3 Function Description and Dynamic Behavior 20

3.1 FUNCTION *VOID SCHM_INIT (SCHM_TASKCONFIGTYPE *SCHM_CONFIG)* 20
3.2 FUNCTION *VOID SCHM_DEINIT (VOID)* 20
3.3 FUNCTION *VOID SCHM_START (VOID)* 21
3.4 FUNCTION *VOID SCHM_OSTICK (VOID)* 21
3.5 FUNCTION *VOID SCHM_BACKGROUND(VOID)* 21
3.6 SCHEDULER TASK CALLBACK FUNCTIONS 21

3. OS Task Manager Conceptual design 22

4 Function Description and Dynamic Behavior 25

4.1 FUNCTION *STATUS_TYPE ACTIVATETASK(TASKTYPE TASKID)* 25
4.2 FUNCTION *STATUS_TYPE TERMINATETASK (VOID)* 25
4.3 FUNCTION *STATUS_TYPE GETTASKID(TASKREFTYPE TASKIDREF)* 25
4.4 FUNCTION *STATUS_TYPE GETTASKSTATE(TASKTYPE TASKID, TASKSTATEREFTYPE STATEREF)* 25
4.5 FUNCTION *VOID DISPATCHER(VOID)* 26
4.6 SCHEDULER TASK CALLBACK FUNCTIONS 26

4. OS Context switch Conceptual design 27

5 Function Description and Dynamic Behavior 29

5.1 FUNCTION *VOID DISABLEALLINTERRUPTS (VOID)* 29
5.2 FUNCTION *VOID ENABLEALLINTERRUPTS (VOID)* 29
5.3 FUNCTION *MEM_RETURNTYPE MEM_ALLOC(MEM_SIZETYPE)* 30
5.4 FUNCTION *VOID TASK_200MS (VOID)* 30
5.5 FUNCTION *VOID TASK_100MS (VOID)* 30
5.6 CONTEXT SWITCH ASSEMBLER 30
5.7 SCHEDULER TASK CALLBACK FUNCTIONS 31

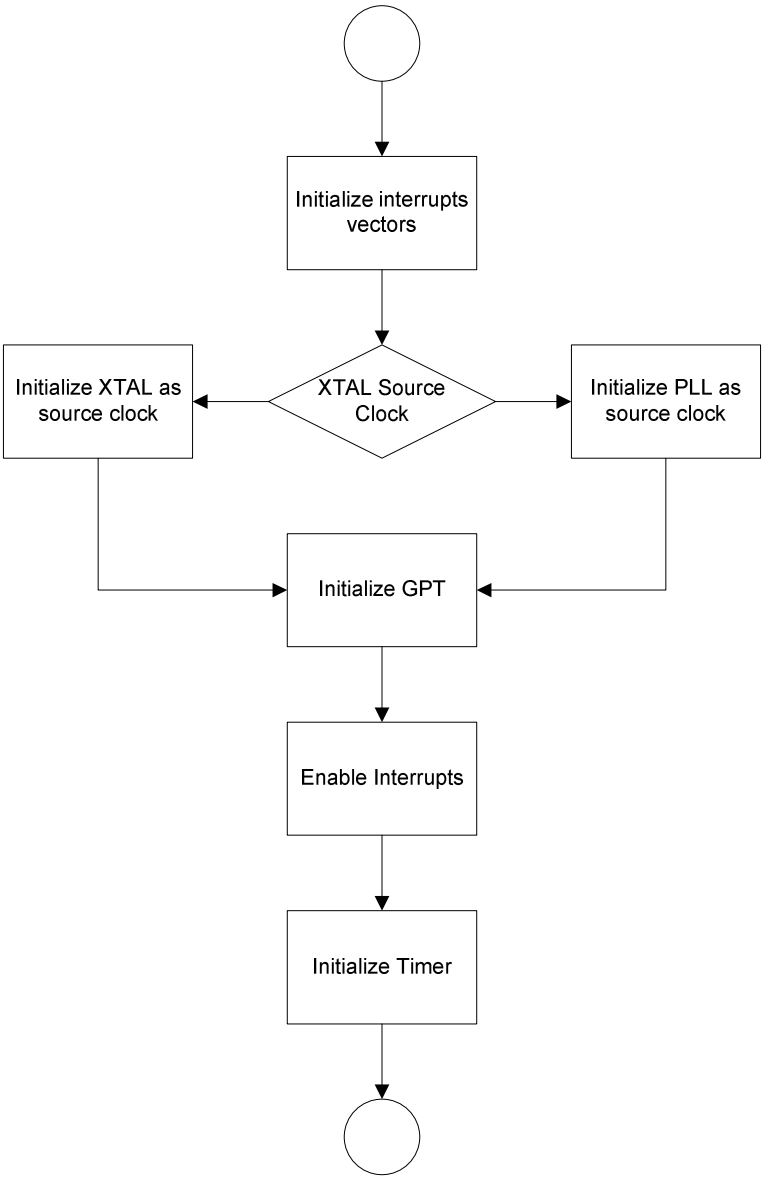
1. OS Tick Module Test Cases 31

2. Context switch Test Cases 34

3. Integration Test Cases 44

4. Scheduler Test Cases 54
5. Integration Test Cases 68
6. Task Manager Test Cases 74
7. Integration Test Cases 85

1. OS Tick SW Conceptual design



2 Function Description and Dynamic Behavior

2.1 Function *void Mcu_Init (void)*

Description	*Void Mcu_Init(void);*
Return Value	*None*
Precondition	*None*
Post condition	*None*

Error Conditions	

Dynamic Behavior
Mcu_Init API initializes the PLL or XTAL frequency configuration.

2.2 Function *void Mcu_InitClock (void)*

Description	*Void Mcu_InitClock(void);*
Return Value	*None*
Precondition	*None*
Post condition	*None*
Error Conditions	

Dynamic Behavior
Mcu_InitClock API enables XTAL Clock as the main clock source.

2.3 Function *void Mcu_DistributePllClock (void)*

Description	*Void Mcu_DistributePllClock(void);*
Return Value	*None*
Precondition	*None*
Post condition	*None*
Error Conditions	

Dynamic Behavior
Mcu_DistributePllClock API enables PLL as the main clock source.

2.4 Function *void Gpt_Init (void)*

Description	*Void Gpt_Init(const Gpt_ConfigType* ConfigPtr);*
Return Value	*None*
Parameters	*ConfigPtr : pointer to the initial configuration structure*
Post condition	*None*
Error Conditions	

Dynamic Behavior
Gpt_Init API initializes PIT according *ConfigPtr* configuration.

2.5 Function *void Gpt_StartTimer(Gpt_ChannelType Channel, Gpt_ValueType Value)*

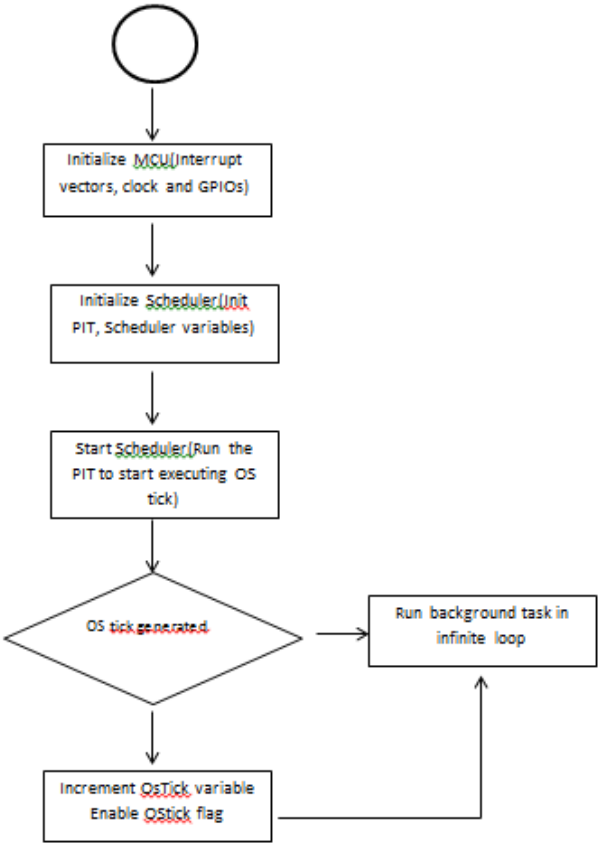
Description	*void Gpt_StartTimer(Gpt_ChannelType Channel, Gpt_ValueType Value);*
Return Value	*None*
Parameters	*Channel: pointer to the initial configuration structure*
Post condition	*None*

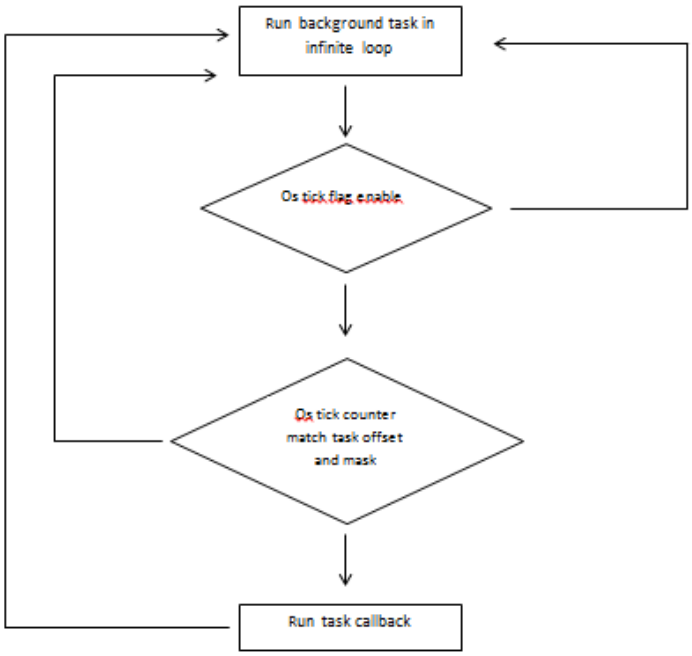
Error Conditions	

Dynamic Behavior

Starst/Run the GPT timer desired in the parameters

2. Scheduler SW Conceptual design





3 Function Description and Dynamic Behavior

3.1 Function *void SchM_Init (SchM_TaskConfigType *SchM_Config)*

Description	*Void SchM_Init(SchM_TaskConfigType *SchM_Config);*	
Return Value	*None*	
Precondition	*None*	
Parameters	*SchM_Config*	*Pointer to the module's configuration structure*
Error Conditions		

Dynamic Behavior
SchM_Init function shall allocate and initialize the resources to be used by the Scheduler Module.

3.2 Function *void SchM_Deinit (void)*

Description	*Void SchM_Deinit(void);*
Return Value	*None*
Precondition	*None*
Post condition	*None*
Error Conditions	

Dynamic Behavior
SchM_Deinit function shall finalize and free the resources used by the Scheduler Module.

3.3 Function *void SchM_Start (void)*

Description	*Void SchM_Start(void);*
Return Value	*None*
Precondition	*None*
Post condition	*None*
Error Conditions	

Dynamic Behavior
SchM_Start API shall start the scheduler.

3.4 Function *void SchM_OsTick (void)*

Description	*Void SchM_OsTick(void);*
Return Value	*None*
Precondition	*None*
Post condition	*None*
Error Conditions	

Dynamic Behavior
SchM_OsTick is a callback function for the base timer Tick.

3.5 Function *void SchM_Background(void)*

Description	*Void SchM_Background(void);*
Return Value	*None*
Parameters	*None*
Post condition	*None*
Error Conditions	

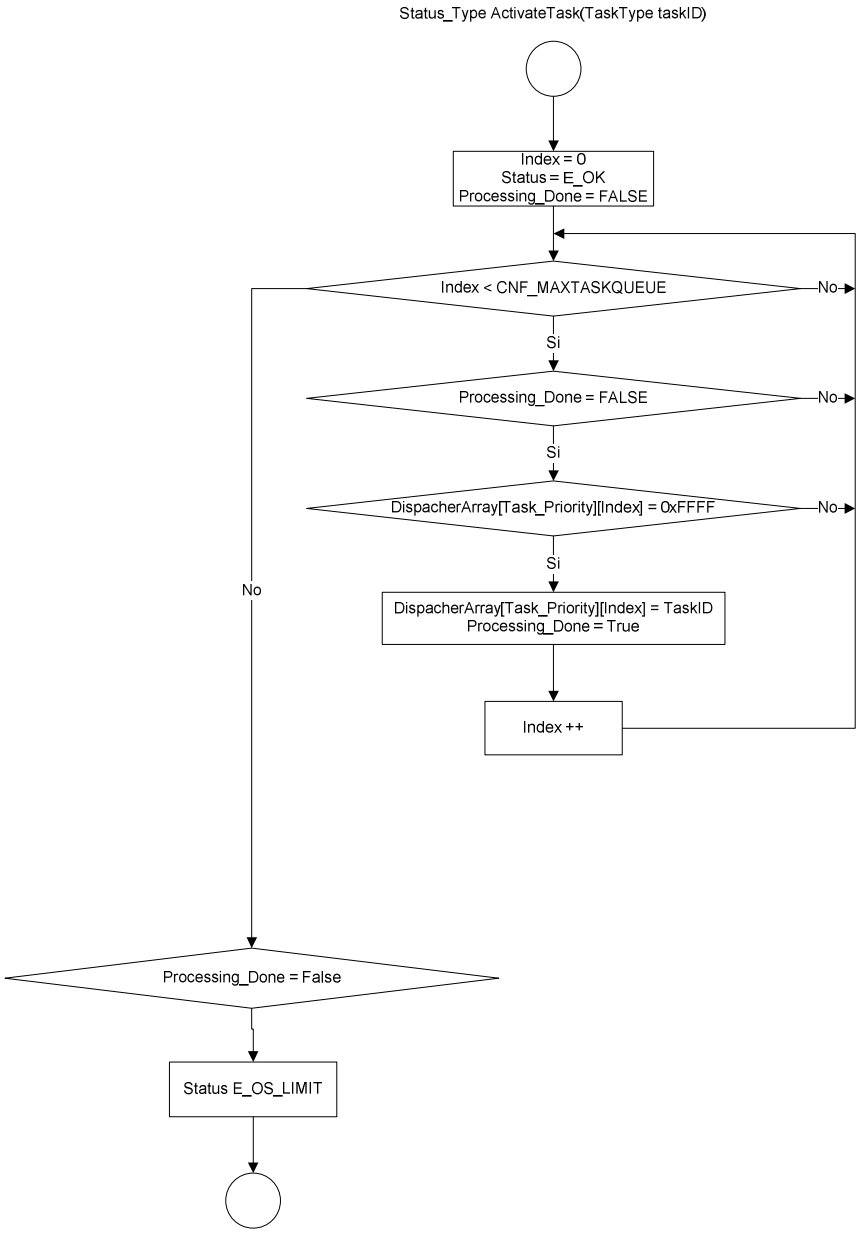
Dynamic Behavior
SchM_Background function runs when no tasks are scheduled.

3.6 Scheduler task callback functions

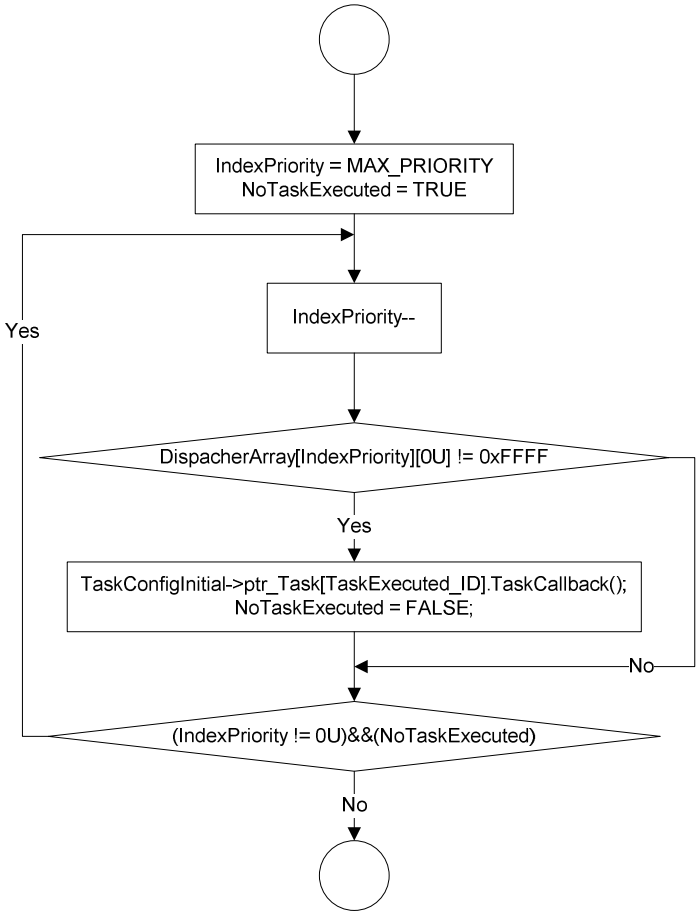
Task	Description & Period
(void) SchM_Tsk_1ms (void);	*Callback function for 1ms*
(void) SchM_Tsk_4ms (void);	*Callback function for 4ms*
(void) SchM_Tsk_8ms (void);	*Callback function for 8ms*
(void) SchM_Tsk_16ms (void);	*Callback function for 16ms*
(void) SchM_Tsk_32ms (void);	*Callback function for 32ms*
(void) SchM_Tsk_64ms (void);	*Callback function for 64ms*

Dynamic Behavior
Callback functions for periods requested.

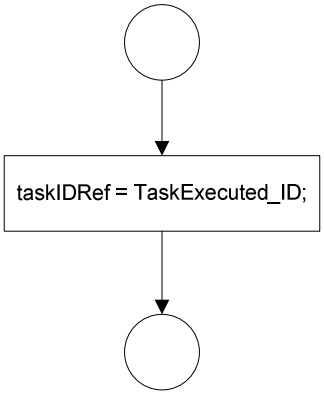
3. OS Task Manager Conceptual design



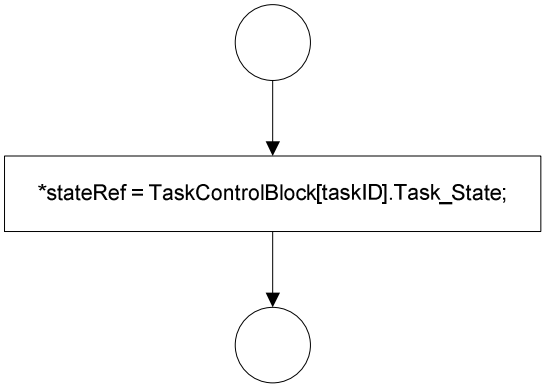
void Dispatcher(void)



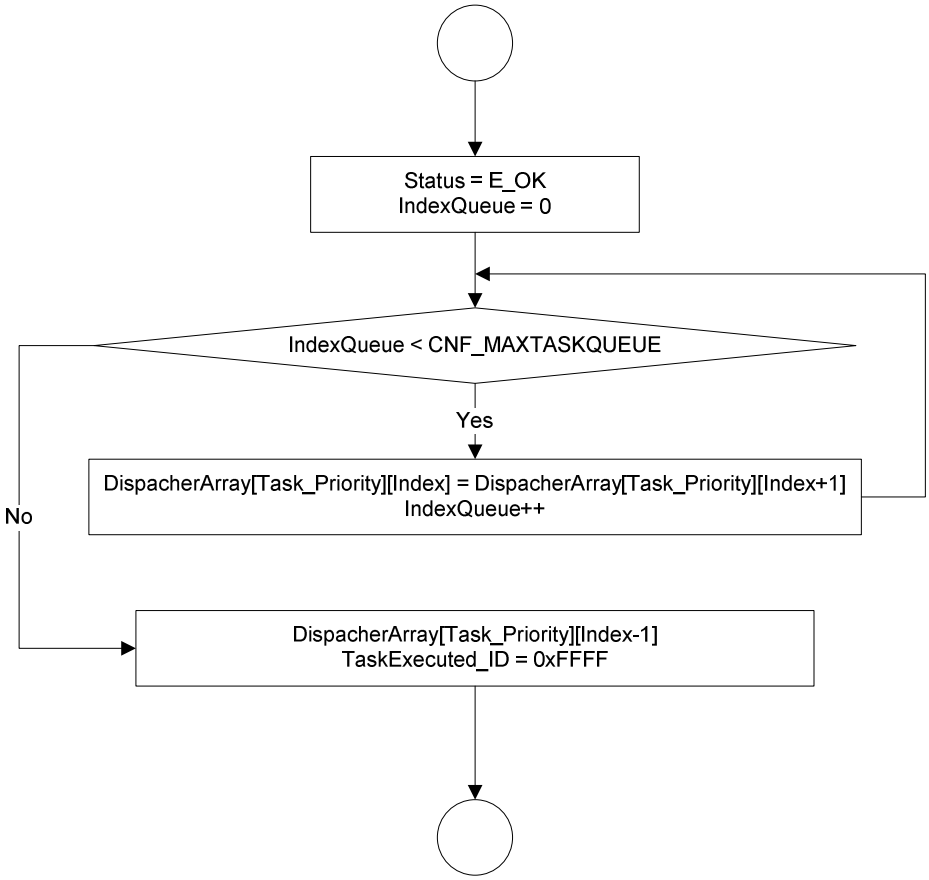
Status_Type GetTaskID(TaskRefType taskIDRef)



Status_Type GetTaskState(TaskType taskID, TaskStateRefType stateRef)



Status_Type TerminateTask (void)



4 Function Description and Dynamic Behavior

4.1 Function *Status_Type ActivateTask(TaskType taskID)*

Description	*Status_Type ActivateTask(TaskType taskID)*	
Return Value	*E_OS_ID, E_OS_LIMIT, E_OK*	
Precondition	*None*	
Parameters	*TaskType*	
Error Conditions		

Dynamic Behavior
Set a task in Ready inside the Dispatcher Array.

4.2 Function *Status_Type TerminateTask (void)*

Description	*Status_Type TerminateTask (void)*
Return Value	*E_OS_RESOURCE, E_OS_CALLEVEL, E_OK*
Precondition	*None*
Post condition	*None*
Error Conditions	

Dynamic Behavior
Reference to a variable of type TaskType that contains the TaskID that is currently running. If no task is running, the variable is set to INVALID_TASK.

4.3 Function *Status_Type GetTaskID(TaskRefType taskIDRef)*

Description	*Status_Type GetTaskID(TaskRefType taskIDRef)*
Return Value	*E_OK*
Precondition	*None*
Post condition	*None*
Error Conditions	

Dynamic Behavior
Return the ID of the Task Running, in case no task is running return 0xFFFF.

4.4 Function *Status_Type GetTaskState(TaskType taskID, TaskStateRefType stateRef)*

Description	*Status_Type GetTaskState(TaskType taskID, TaskStateRefType stateRef)*
Return Value	*E_OK, E_OS_ID*
Precondition	*None*
Post condition	*None*
Error Conditions	

Dynamic Behavior
Returns the State of the ID specified in the parameter tasked

4.5 Function *void Dispatcher(void)*

Description	*void Dispatcher(void)*
Return Value	*None*
Parameters	*None*
Post condition	*None*
Error Conditions	

Dynamic Behavior
Run the hightest priority task in ready.

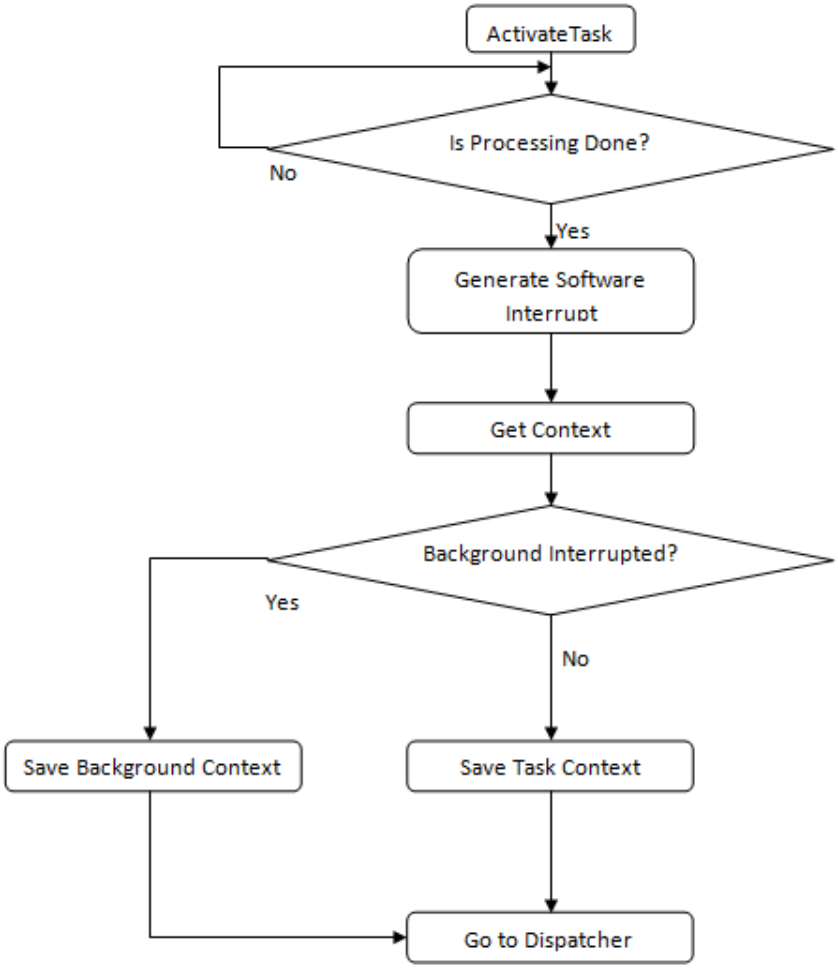
4.6 Scheduler task callback functions

Task	Description & Period
(void) SchM_Tsk_1ms (void);	*Callback function for 1ms*
(void) SchM_Tsk_4ms (void);	*Callback function for 4ms*
(void) SchM_Tsk_8ms (void);	*Callback function for 8ms*
(void) SchM_Tsk_16ms (void);	*Callback function for 16ms*
(void) SchM_Tsk_32ms (void);	*Callback function for 32ms*
(void) SchM_Tsk_64ms (void);	*Callback function for 64ms*

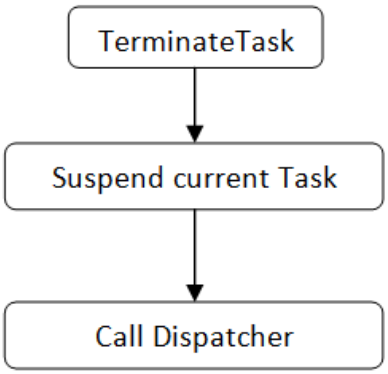
Dynamic Behavior
Callback functions for periods requested.

4. OS Context switch Conceptual design

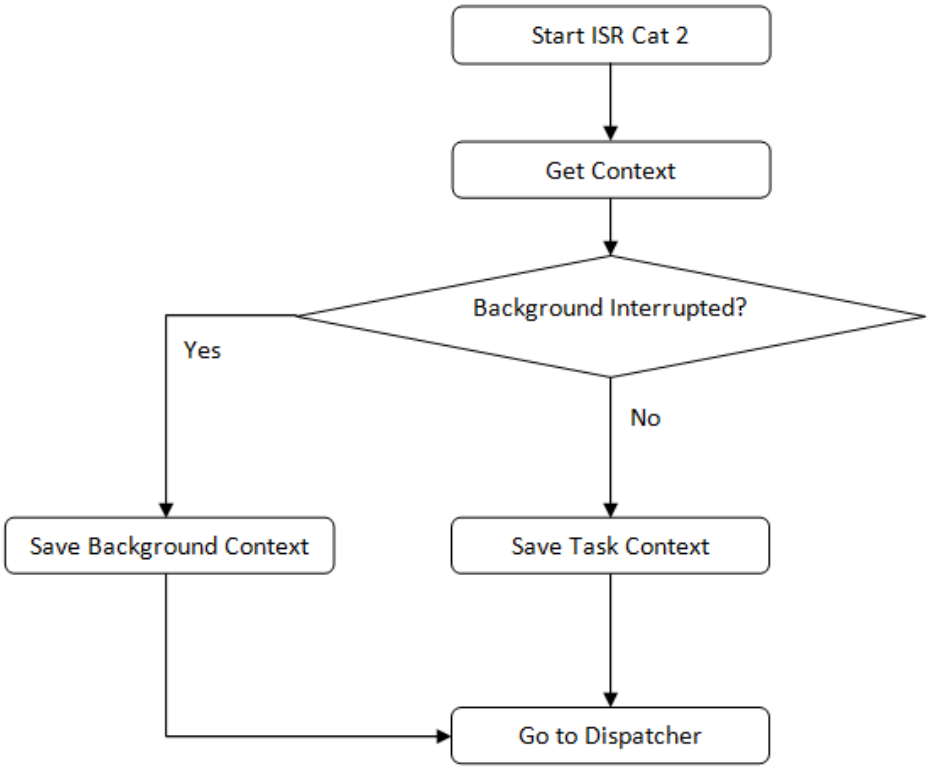
Dispatcher called after an ActivateTask routine



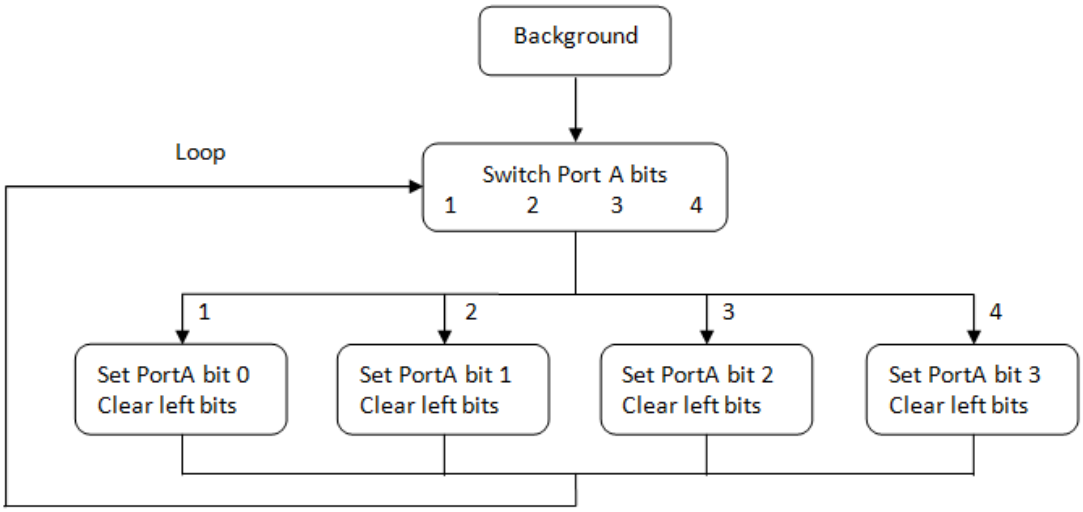
Dispatcher called after TerminateTask



Switch Context in an ISR category 2 and Dispatcher call at the end



Rotabit



5 Function Description and Dynamic Behavior

5.1 Function *void DisableAllInterrupts (void)*

Description	*void DisableAllInterrupts (void)*
Return Value	*Void*
Precondition	*Within the critical section, no API service calls are allowed.* *How the system interrupts are disabled will differ between implementations and between microcontrollers*
Parameters	*None*
Post Conditions	*System service shall support nested calls in order to avoid any enabling interrupt when not yet required.*

Dynamic Behavior

API service saves the current state of all interrupts, disables all interrupts that are enabled, and identifies the beginning of a critical section.

5.2 Function *void EnableAllInterrupts (void)*

Description	*void EnableAllInterrupts (void)*
Return Value	*void*
Precondition	*Within the critical section, no API service calls are allowed. If DisableAllInterrupts () was not previously called, the action taken is undefined by the specification.*
Post condition	*System service shall support nested calls in order to avoid any enabling interrupt when not yet required.*
Error Conditions	

Dynamic Behavior

API service enables all interrupts that were enabled prior to the previous call to DisableAllInterrupts and identifies the end of a critical section.

5.3 Function *Mem_ReturnType Mem_Alloc(Mem_SizeType)*

Description	*Mem_ReturnType Mem_Alloc(Mem_SizeType)*
Return Value	*Mem_ReturnType - Returns a pointer to the allocated memory, or NULL if the request fails*
Precondition	*None*
Parameters	*Mem_SizeType - Size of memory in bytes to allocate*
Error Conditions	

Dynamic Behavior
Dynamic memory allocation function

5.4 Function *void Task_200ms (void)*

Description	*void Task_200ms (void)*
Return Value	*void*
Precondition	*None*
Post condition	*None*

Dynamic Behavior
Timer event periodic task callback function 200ms.

5.5 Function *void Task_100ms (void)*

Description	*void Task_100ms (void)*
Return Value	*None*
Parameters	*None*
Post condition	*None*
Error Conditions	

Dynamic Behavior
Timer event periodic task callback function 100ms.

5.6 Context Switch Assembler

```
__asm
  {
      PULD                              ; (CCR) Pull stack into the CPU Register D
      STD   CCR_ContextSaving_u16       ; Store the CPU Register D value in fixed memory
      PULD                              ; (D || BA) Pull the stack into the CPU Register D
      STD   D_ContextSaving_u16         ; Store the CPU Register D value in fixed memory
      PULD                              ; (IX) Pull the stack into the CPU Register D
      STD   IY_ContextSaving_u16        ; Store the CPU Register D value in fixed memory
      PULD                              ; (IY) Pull the stack into the CPU Register D
      STD   IX_ContextSaving_u16        ; Store the CPU Register D value in fixed memory
      PULD                              ; (PC) Pull the stack into the CPU Register D
      STD   PC_ContextSaving_u16        ; Store the CPU Register D value in fixed memory
      PULA                              ; (P_PAGE) Pull the stack into the CPU Register D
      STAA   PPAGE_ContextSaving_u8 ; Store the CPU Register A value in fixed memory
```

```
    STS     SP_ContextSaving_u16            ; (SP) Store Stack Pointer in fixed memory
}
```

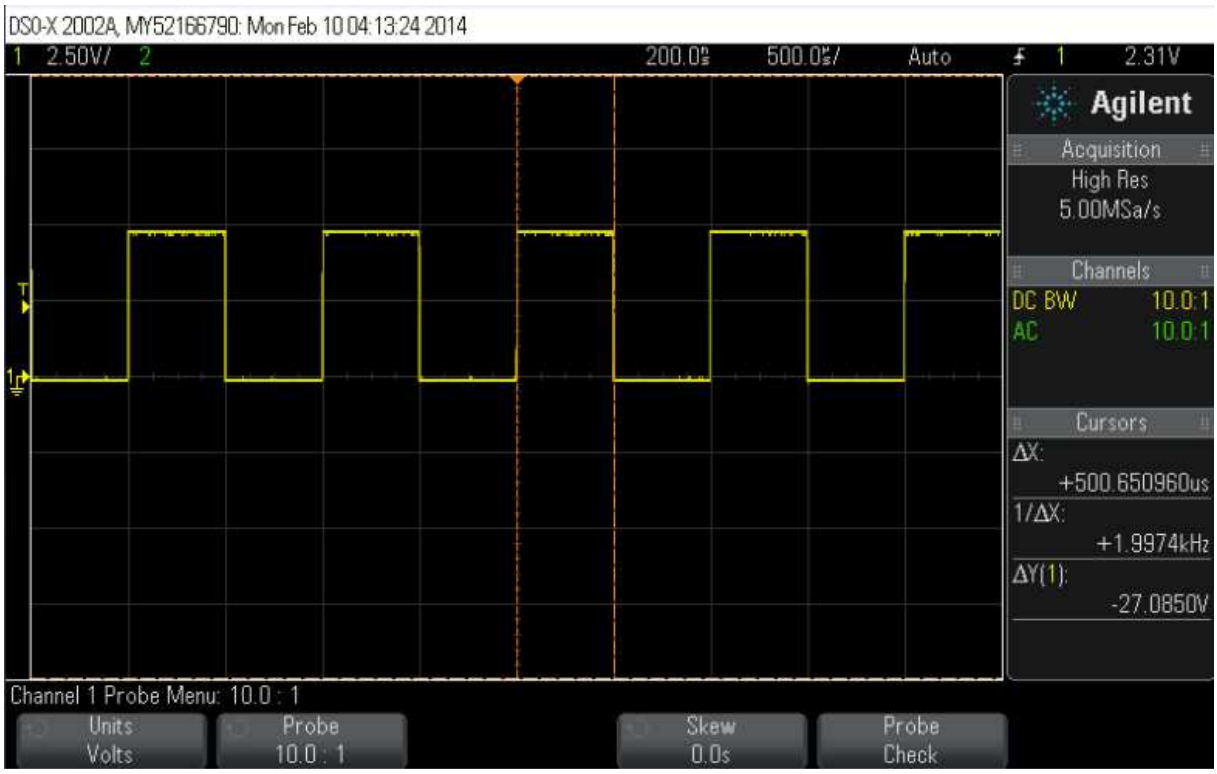
5.7 Scheduler task callback functions

Task	Description & Period
(void) SchM_Tsk_1ms (void);	*Callback function for 1ms*
(void) SchM_Tsk_4ms (void);	*Callback function for 4ms*
(void) SchM_Tsk_8ms (void);	*Callback function for 8ms*
(void) SchM_Tsk_16ms (void);	*Callback function for 16ms*
(void) SchM_Tsk_32ms (void);	*Callback function for 32ms*
(void) SchM_Tsk_64ms (void);	*Callback function for 64ms*
void Task_200ms (void)	*Callback function for 200ms*
void Task_100ms (void)	*Callback function for 100ms*

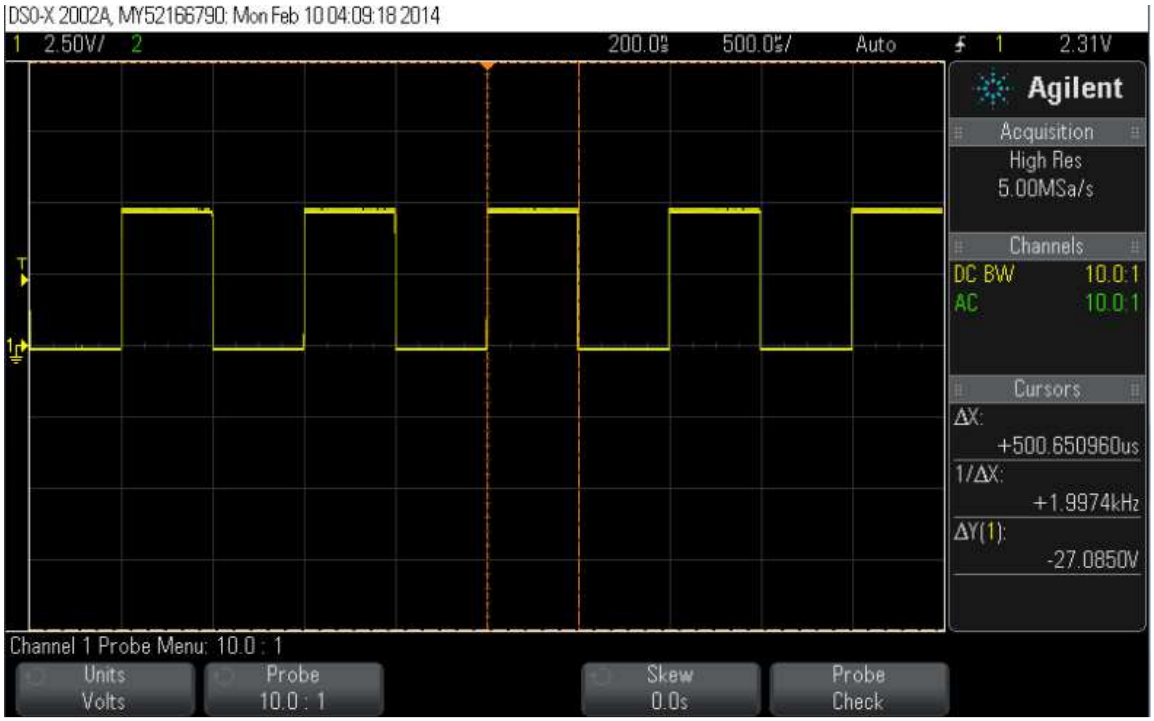
Dynamic Behavior
Callback functions for periods requested.

1. OS Tick Module Test Cases

Test Case	**ID**	**Status**
	0.1	
Requirements covered		
0.1,0.2,0.3,0.4,0.5,0.6,0.7		
Test Procedure		
Set up the frequency of the output PA0 of the ECU to 500 Hz using the XTAL		
Expected Results		
Using an oscilloscope verify that the frequency of the ECU is set to 500 Hz.		
Actual Results		**Test Results**
The frequency of the output PA0 of the ECU is equal to 500 Hz.		PASS
Comments		



Test Case	ID	Status
	0.2	
Requirements covered		
0.1,0.2,0.3,0.4,0.5,0.6,0.7		
Test Procedure		
Set up the frequency of the output PA0 of the ECU to 500 Hz using the Internal Oscillator of the ECU		
Expected Results		
Using an oscilloscope verify that the frequency of the ECU is set to 500 Hz.		
Actual Results		**Test Results**
The frequency of the output PA0 of the ECU is equal to 500 Hz.		PASS
Comments		



Test Case	**ID**	**Status**
	0.3	
Requirements covered		
0.1,0.2,0.3,0.4,0.5,0.6,0.7		
Test Procedure		
Stop tick of the output PA4 and verify with the oscilloscope that the output is turned off		
Expected Results		
The voltage of the output PA4 is turned off.		
Actual Results		**Test Results**
The voltage of the output PA4 is turned off.		PASS
Comments		

Test Case	ID	Status
	0.4	
Requirements covered		
0.1,0.2,0.3,0.4,0.5,0.6,0.7		
Test Procedure		
Disable notification of the output PA3 and verify with the oscilloscope that the output is turned off		
Expected Results		
The voltage of the output PA3 is turned off.		
Actual Results		**Test Results**
The voltage of the output PA3 is turned off.		PASS
Comments		

2. Context switch Test Cases

Test Case	ID	Status
File Structure	2.0	Done
Requirements covered		
3.0		
Test Procedure		
The void DisableAllInterrupts (void) API service saves the current state of all interrupts, disables all interrupts that are enabled, and identifies the beginning of a critical section. - Within the critical section, no API service calls are allowed. - How the system interrupts are disabled will differ between implementations and between microcontrollers		
Expected Results		
Interrupts Disabled		
Actual Results		**Test Results**
Interrupts Disabled during critical code execution. See Figure 1, Figure 2		PASS
Comments		

```
void DisableAllInterrupts(void){

 if(gInterrupt_nested_flag==0) {
    DisableInterrupts;
 }
 gInterrupt_nested_flag++;
}

/*********************************
*   Function: EnableAllInterrupts
*
*   Description: To enable MCU ir
*                function.
*
*   Caveats: Non Reentrant
*********************************
void EnableAllInterrupts(void){

 gInterrupt_nested_flag--;

 if(gInterrupt_nested_flag==0){
    EnableInterrupts;
 }
}
```

Figure 1

```
void Dispatcher(void)
{
    u16 IndexPriority = MAX_PRIORITY;
    u16 NoTaskExecuted = TRUE;
    u16 DispatcherDone = FALSE;

    while(!DispatcherDone)
    {
        do
        {
            IndexPriority--;
            if(DispacherArray[IndexPriority][0U] != 0xFFFF)
            {
                TaskExecuted_ID = DispacherArray[IndexPriority][0U];
                if(TaskControlBlock[TaskExecuted_ID].Task_Interrupted == TASK_PREEMPTED)
                {
                    DisableAllInterrupts();
                    /* Return Context to the Interrupted Task */
                    TaskControlBlock[TaskExecuted_ID].Task_Interrupted = TASK_NOPREEMPTED;
                    PPAGE_ContextRestore_u8  =  TaskControlBlock[TaskExecuted_ID].Task_ContextSave.PPAGE_TaskContext_u16;
                    PC_ContextRestore_u16    =  TaskControlBlock[TaskExecuted_ID].Task_ContextSave.PC_TaskContext_u16;
                    IY_ContextRestore_u16    =  TaskControlBlock[TaskExecuted_ID].Task_ContextSave.X_TaskContext_u16;
                    IX_ContextRestore_u16    =  TaskControlBlock[TaskExecuted_ID].Task_ContextSave.Y_TaskContext_u16;
                    D_ContextRestore_u16     =  TaskControlBlock[TaskExecuted_ID].Task_ContextSave.D_TaskContext_u16;
                    CCR_ContextRestore_u16   =  TaskControlBlock[TaskExecuted_ID].Task_ContextSave.CCR_TaskContext_u16;
                    SP_ContextRestore_u16    =  TaskControlBlock[TaskExecuted_ID].Task_ContextSave.SP_TaskContext_u16;
                    __asm
                    {
                        LDS    SP_ContextRestore_u16          ; (SP) Load from fixed memory location to SP
                        LDAA   PPAGE_ContextRestore_u8        ; (P_PAGE) Load from a fixed memory location to Register A
                        PSHA                                  ; Push the CPU Register A value into the Stack
                        LDD    PC_ContextRestore_u16          ; (PC) Load from a fixed memory location to Register D
                        PSHD                                  ; Push the CPU Register D value into the Stack
                        LDD    IY_ContextRestore_u16          ; (IY) Load from a fixed memory location to Register D
                        PSHD                                  ; Push the CPU Register D value into the Stack
                        LDD    IX_ContextRestore_u16          ; (IX) Load from a fixed memory location to Register D
                        PSHD                                  ; Push the CPU Register D value into the Stack
                        LDD    D_ContextRestore_u16           ; (D || BA) Load from a fixed memory location to Register D
                        PSHD                                  ; Push the CPU Register D value into the Stack
                        LDD    CCR_ContextRestore_u16         ; (CCR) Load from a fixed memory location to Register D
                        PSHD                                  ; Push the CPU Register D value into the Stack
                    }
                    EnableAllInterrupts();
                    asm(RTI);
                }
                else
                {
                    TaskControlBlock[TaskExecuted_ID].Task_State = RUNNING;
                    TaskConfigInitial->ptr_Task[TaskExecuted_ID].TaskCallback();
                    NoTaskExecuted = FALSE;
                }
            }
        }
        while((IndexPriority != 0U)&&(NoTaskExecuted));
        if((IndexPriority == 0U) && (NoTaskExecuted))
```

Figure 2

Test Case	ID	Status
Task Manager Services	**2.1**	**Done**
Requirements covered		
3.1		
Test Procedure		
The void EnableAllInterrupts (void) API service enables all interrupts that were enabled prior to the previous call to DisableAllInterrupts() and identifies the end of a critical section. - Within the critical section, no API service calls are allowed. - If DisableAllInterrupts() was not previously called, the action taken is undefined by the specification.		

Expected Results	
Interrupts Enabled	
Actual Results	**Test Results**
Interrupts Enabled See Figure 2 & Figure 3	PASS
Comments	

```
void DisableAllInterrupts(void){

 if(gInterrupt_nested_flag==0) {
    DisableInterrupts;
 }
 gInterrupt_nested_flag++;
}

/*******************************
*   Function: EnableAllInterrupts
*
*   Description: To enable MCU ir
*                function.
*
*   Caveats: Non Reentrant
*******************************
void EnableAllInterrupts(void){

 gInterrupt_nested_flag--;

 if(gInterrupt_nested_flag==0){
    EnableInterrupts;
 }
}
```

Figure 2

```
void Dispatcher(void)
{
    u16 IndexPriority = MAX_PRIORITY;
    u16 NoTaskExecuted = TRUE;
    u16 DispatcherDone = FALSE;

    while(!DispatcherDone)
    {
        do
        {
            IndexPriority--;
            if(DispacherArray[IndexPriority][0U] != 0xFFFF)
            {
                TaskExecuted_ID = DispacherArray[IndexPriority][0U];
                if(TaskControlBlock[TaskExecuted_ID].Task_Interrupted == TASK_PREEMPTED)
                {
                    DisableAllInterrupts();
                    /* Return context to the Interrupted Task */
                    TaskControlBlock[TaskExecuted_ID].Task_Interrupted = TASK_NOPREEMPTED;
                    PPAGE_ContextRestore_u8  =  TaskControlBlock[TaskExecuted_ID].Task_ContextSave.PPAGE_TaskContext_u16;
                    PC_ContextRestore_u16    =  TaskControlBlock[TaskExecuted_ID].Task_ContextSave.PC_TaskContext_u16;
                    IY_ContextRestore_u16    =  TaskControlBlock[TaskExecuted_ID].Task_ContextSave.X_TaskContext_u16;
                    IX_ContextRestore_u16    =  TaskControlBlock[TaskExecuted_ID].Task_ContextSave.Y_TaskContext_u16;
                    D_ContextRestore_u16     =  TaskControlBlock[TaskExecuted_ID].Task_ContextSave.D_TaskContext_u16;
                    CCR_ContextRestore_u16   =  TaskControlBlock[TaskExecuted_ID].Task_ContextSave.CCR_TaskContext_u16;
                    SP_ContextRestore_u16    =  TaskControlBlock[TaskExecuted_ID].Task_ContextSave.SP_TaskContext_u16;
                    __asm
                    {
                        LDS    SP_ContextRestore_u16          ; (SP) Load from fixed memory location to SP
                        LDAA   PPAGE_ContextRestore_u8        ; (P_PAGE) Load from a fixed memory location to Register A
                        PSHA                                  ; Push the CPU Register A value into the Stack
                        LDD    PC_ContextRestore_u16          ; (PC) Load from a fixed memory location to Register D
                        PSHD                                  ; Push the CPU Register D value into the Stack
                        LDD    IY_ContextRestore_u16          ; (IY) Load from a fixed memory location to Register D
                        PSHD                                  ; Push the CPU Register D value into the Stack
                        LDD    IX_ContextRestore_u16          ; (IX) Load from a fixed memory location to Register D
                        PSHD                                  ; Push the CPU Register D value into the Stack
                        LDD    D_ContextRestore_u16           ; (D || BA) Load from a fixed memory location to Register D
                        PSHD                                  ; Push the CPU Register D value into the Stack
                        LDD    CCR_ContextRestore_u16         ; (CCR) Load from a fixed memory location to Register D
                        PSHD                                  ; Push the CPU Register D value into the Stack
                    }
                    EnableAllInterrupts();
                    asm(rti);
                }
                else
                {
                    TaskControlBlock[TaskExecuted_ID].Task_State = RUNNING;
                    TaskConfigInitial->ptr_Task[TaskExecuted_ID].TaskCallback();
                    NoTaskExecuted = FALSE;
                }
            }
        }
        while((IndexPriority != 0U)&&(NoTaskExecuted));
        if((IndexPriority == 0U) && (NoTaskExecuted))
```

```
/*****************************************************************************
*   Global Function Prototypes
*****************************************************************************

extern Status_Type ActivateTask(TaskType taskID);
extern Status_Type TerminateTask (void);
extern Status_Type GetTaskID(TaskRefType taskIDRef);
extern Status_Type GetTaskState(TaskType taskID, TaskStateRefType stateRef);
extern void Dispatcher(void);
```

Figure 3

Test Case	ID	Status
Configuration Support	2.2	Done
Requirements covered		
3.3		
Test Procedure		
DisableAllInterrupts and EnableAllInterrupts system services shall support nested calls in order to avoid any enabling interrupt when not yet required.		
Expected Results		
Interrupts Disabled/Enabled Nested		
Actual Results		**Test Results**
Interrupts Disabled/Enabled Nested		PASS
Comments		
See Figure 4		

```
void DisableAllInterrupts(void){

 if(gInterrupt_nested_flag==0) {
    DisableInterrupts;
 }
 gInterrupt_nested_flag++;
}

/*******************************
*   Function: EnableAllInterrupts
*
*   Description: To enable MCU ir
*                function.
*
*   Caveats: Non Reentrant
*******************************/
void EnableAllInterrupts(void){

 gInterrupt_nested_flag--;

 if(gInterrupt_nested_flag==0){
    EnableInterrupts;
 }
}
```

Figure 4

Test Case	ID	Status
Task Configuration	2.3	Done
Requirements covered		
3.9		
Test Procedure		
Dispatcher shall only be called at the end of the following system services: – ActivateTask – TerminateTask – End of CAT 2 ISR's		
Expected Results		

Dispatcher Called Only after Activate Task, Terminate Task and ISR.	
Actual Results	**Test Results**
Dispatcher Called Only after Activate Task, Terminate Task and ISR.	PASS
Comments	
See Figure 5 & Figure 6	

```
void interrupt  vfnPIT_Channel1_Isr( void  )
{
        __asm
    {
          PULD                              ; (CCR) Pull stack into the CPU Register D
          STD     CCR_ContextSaving_u16     ; Store the CPU Register D value in fixed memory
          PULD                              ; (D || BA) Pull the stack into the CPU Register D
          STD     D_ContextSaving_u16       ; Store the CPU Register D value in fixed memory

                                    ....
                                    More Code
                                    ....

    /* Verify that Real Time Interrupt caused the interrupt */
    if( PITTF_PTF1 == 1u )
    {
        /*call callback function, if initialized*/
        if((Gpt_ConfigType_initial->ptr_Pit_ChannelConfig[1].Pit_Channel_Callback !=
        {
            Gpt_ConfigType_initial->ptr_Pit_ChannelConfig[1].Pit_Channel_Callback();
        }
    }
    /* Clear the real time interrupt flag */
    PITTF_PTF1 = 1u;
              __asm
    {
         MOVW @Dispatcher, 2, -sp;
         RTS
    }
```

Figure 5

```
*****************************************************************************************************
Status_Type TerminateTask (void)
{
  Status_Type Status = E_OK;
  u16 IndexQueue=0;

  TaskControlBlock[TaskExecuted_ID].Task_State = SUSPENDED;
  for(IndexQueue;IndexQueue < CNF_MAXTASKQUEUE;IndexQueue++)
  {
     DispacherArray[TaskConfigInitial->ptr_Task[TaskExecuted_ID].Task_Priority][IndexQueue] =
     DispacherArray[TaskConfigInitial->ptr_Task[TaskExecuted_ID].Task_Priority][IndexQueue+1];
  }
  DispacherArray[TaskConfigInitial->ptr_Task[TaskExecuted_ID].Task_Priority][IndexQueue-1] = 0xFFFF;
  TaskExecuted_ID = 0xFFFF;
  Dispatcher();
  return Status;
}
```

Figure 6

Test Case	**ID**	**Status**
Data definitions	**2.4**	**Done**
Requirements covered		
3.10, 3.11,3.12,3.13,3.14,3.22		
Test Procedure		
If higher priority task than the current running task is ready for execution, the dispatcher shall switch tasks context and run the new task - Previous task shall be queued in its corresponding priority buffer - Task states shall be changed accordingly		
Expected Results		

Task must be preempt accordantly their priority.	
Actual Results	**Test Results**
Task are preempt accordantly their priority.	PASS
Comments	
See Figure 7, Figure 8, Figure 9	

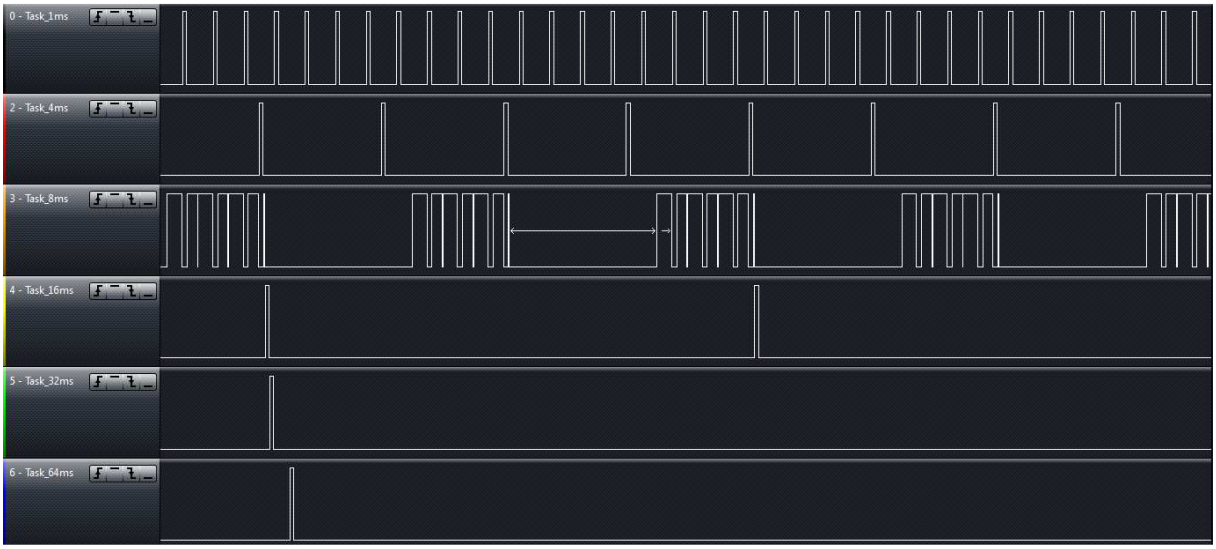


Figure 7

```
const Task_Descriptor TaskInitial[]=
{
  {TASK_1MS, PRIORITY_5, MASK_1MS, OFFSET_00,0,0,Task_1ms},
  {TASK_4MS, PRIORITY_4, MASK_4MS, OFFSET_01,0,0,Task_4ms},
  {TASK_8MS, PRIORITY_3, MASK_8MS, OFFSET_03,0,0,Task_8ms},
  {TASK_16MS,PRIORITY_2,MASK_16MS, OFFSET_05,0,0,Task_16ms},
  {TASK_32MS,PRIORITY_1,MASK_32MS, OFFSET_07,0,0,Task_32ms},
  {TASK_64MS,PRIORITY_0,MASK_64MS, OFFSET_11,0,0,Task_64ms}
};
```

Figure 8

```
TASK (Task_1ms)
{
  u16 index;
  Status_Type Error_Result = E_OK;
  PORTB_PB0= 1;

  for (index=0;index<500;index++)
  {
     PORTB_PB0= 1;
  }
  PORTB_PB0= 0;
  Error_Result=TerminateTask();
}
```

```
TASK (Task_4ms)
{
  u16 index;
  Status_Type Error_Result = E_OK;
  PORTB_PB1= 1;

  for (index=0;index<500;index++)
  {
    PORTB_PB1= 1;
  }
  PORTB_PB1= 0;
  Error_Result=TerminateTask();
}
```

```
TASK (Task_8ms)
{
  u16 index;
  Status_Type Error_Result = E_OK;
  PORTB_PB2= 1;
  for (index=0;index<10000;index++)
  {
    PORTB_PB2= 1;
  }
  PORTB_PB2= 0;
  Error_Result=TerminateTask();
}
```

Figure 9

Test Case	ID	Status
Additional Information	2.5	Done
Requirements covered		
3.15,3.16,3.17		
Test Procedure		
Task Stack shall be allocated with Memory Allocation interface		
Expected Results		
Task Stack shall be allocated with Memory Allocation interface		
Actual Results		**Test Results**
Task Stack is be allocated with Memory Allocation interface		PASS
Comments		
See Figure 10		

SchM.c

```
/* Initialize Dispacher Array Vector */
for(index_a=0U;index_a < MAX_PRIORITY;index_a++)
{
  for(index_b=0U;index_b < CNF_MAXTASKQUEUE;index_b++)
  {
     DispacherArray[index_a][index_b] = 0xFFFF;
  }
}

BackgroundControlBlock.BackgroundContextSave.CCR_TaskContext_u16    = 0U;
BackgroundControlBlock.BackgroundContextSave.D_TaskContext_u16      = 0U;
BackgroundControlBlock.BackgroundContextSave.X_TaskContext_u16      = 0U;
BackgroundControlBlock.BackgroundContextSave.Y_TaskContext_u16      = 0U;
BackgroundControlBlock.BackgroundContextSave.PC_TaskContext_u16     = 0U;
BackgroundControlBlock.BackgroundContextSave.PPAGE_TaskContext_u16  = 0U;
BackgroundControlBlock.BackgroundContextSave.SP_TaskContext_u16     = 0U;
BackgroundControlBlock.BackgroundInterrupted                        = TASK_NOPREEMPTED;

TaskControlBlock = ((Task_Control_Block *__far)Mem_Alloc((MAX_NUM_TASK*sizeof(Task_Control_Block))))
for(index_ControlBlock=0U;index_ControlBlock < MAX_NUM_TASK;index_ControlBlock++)
{
  TaskControlBlock[index_ControlBlock].Task_ID                                  = TaskConfigInitial
  TaskControlBlock[index_ControlBlock].Task_Priority                            = TaskConfigInitial
  TaskControlBlock[index_ControlBlock].Task_State                               = SUSPENDED;
  TaskControlBlock[index_ControlBlock].Stack_Information.StartAddress           = 0U;
  TaskControlBlock[index_ControlBlock].Stack_Information.EndAddress             = 0U;
  TaskControlBlock[index_ControlBlock].Task_Deadline.Relative                   = 0U;
  TaskControlBlock[index_ControlBlock].Task_Deadline.Absolute                   = 0U;
  TaskControlBlock[index_ControlBlock].Task_ContextSave.CCR_TaskContext_u16      = 0U;
  TaskControlBlock[index_ControlBlock].Task_ContextSave.D_TaskContext_u16        = 0U;
  TaskControlBlock[index_ControlBlock].Task_ContextSave.X_TaskContext_u16        = 0U;
  TaskControlBlock[index_ControlBlock].Task_ContextSave.Y_TaskContext_u16        = 0U;
  TaskControlBlock[index_ControlBlock].Task_ContextSave.PC_TaskContext_u16       = 0U;
  TaskControlBlock[index_ControlBlock].Task_ContextSave.PPAGE_TaskContext_u16    = 0U;
  TaskControlBlock[index_ControlBlock].Task_ContextSave.SP_TaskContext_u16       = 0U;
  TaskControlBlock[index_ControlBlock].Task_Interrupted                         = TASK_NOPREEMPTED;
}

while(1)
{
  SchM_Background();
  _FEED_COP();
}
```

Figure 10

Test Case	ID	Status

Scheduler Services	2.6	Done
Requirements covered		
3.21		
Test Procedure		
Background Task shall execute RotaBit algorithm using LED 0 .. LED 3 (PA0 .. PA3) from EP100 Board. – Increment a counter on the event driven task Timed Task 1 – This counter shall served as timer counter for the RotaBit algorithm		
Expected Results		
Background Task shall execute RotaBit algorithm using LED 0 .. LED 3 (PA0 .. PA3) from EP100 Board. – Increment a counter on the event driven task Timed Task 1 – This counter shall served as timer counter for the RotaBit algorithm		
Actual Results		**Test Results**
Background Task is execute RotaBit algorithm using LED 0 .. LED 3 (PA0 .. PA3) from EP100 Board. – Increment a counter on the event driven task Timed Task 1 – This counter shall served as timer counter for the RotaBit algorithm		PASS
Comments		
See Figure 11, Figure 12		

```
switch(gRotaBit_counter){

      case 1:
        PORTA_PA0 = 1;
        PORTA_PA1 = 0;
        PORTA_PA2 = 0;
        PORTA_PA3 = 0;
      break;
      case 2:
        PORTA_PA0 = 0;
        PORTA_PA1 = 1;
        PORTA_PA2 = 0;
        PORTA_PA3 = 0;
      break;
      case 3:
        PORTA_PA0 = 0;
        PORTA_PA1 = 0;
        PORTA_PA2 = 1;
        PORTA_PA3 = 0;
      break;
      case 4:
        PORTA_PA0 = 0;
        PORTA_PA1 = 0;
        PORTA_PA2 = 0;
        PORTA_PA3 = 1;
      break;
      default:
        PORTA_PA0 = 0;
        PORTA_PA1 = 0;
        PORTA_PA2 = 0;
        PORTA_PA3 = 0;
      break;
}
```

Figure 11

```
TASK (Timed_Task1){
  PORTB_PB6= 1;

  PORTA_PA4= ~PORTA_PA4;
  if(gRotaBit_counter==4){
     gRotaBit_counter=0;
  }
  gRotaBit_counter++;
  PORTB_PB6= 0;
}
```

Figure 12

Test Case	**ID**	**Status**
Background	**2.7**	**Done**
Requirements covered		
2.13		
Test Procedure		
Call SchM_Start inside of main, then call SchM_Background.		
Expected Results		
SchM_Start must be called from main, after that SchM_Background should be called and must never ends		
Actual Results		**Test Results**
SchM_Background is called after SchM_Start inside of main and never ends		PASS
Comments		
See Figure 13 & Figure 14		

Main.c

```
/*****************************************************************************
*   Function: main()
*
*   Description: what I can say about it, it just the tipical main
*
*   Caveats:
*****************************************************************************/
void main(void)
{
  /*Disable interrupts to start the MCU configuration*/
  DisableInterrupts;
  /* Intilialize the interupt vector base address for default location */
  Init_Interrupt_Vector();

  Mcu_Init();
  DDRA =0x0F;
  DDRB =0x0F;

  EnableInterrupts;

  SchM_Init(TaskConfigInitial);
  /* Loop Forever and ever */
  SchM_Start();
}
```

Figure 13

SchM.c

```
/*****************************************************************************************************
*   Function: SchM_Start
*
*   Description: Starts the scheduler
*
*   Caveats:
*****************************************************************************************************/
void SchM_Start(void)
{
    u16 index_a;
    u16 index_b;
    u16 index_ControlBlock;
    /*Start the timer for the OS tick*/
    Gpt_StartTimer(CHANNEL_0, 500u);
    /*Enable notification*/
    Gpt_EnableNotification(CHANNEL_0);
    /* Initialize Dispacher Array Vector */
    for(index_a=0U;index_a < MAX_PRIORITY;index_a++)
    {
      for(index_b=0U;index_b < CNF_MAXTASKQUEUE;index_b++)
      {
        DispacherArray[index_a][index_b] = 0xFFFF;
      }
    }

    TaskControlBlock = ((Task_Control_Block *__far)Mem_Alloc((MAX_NUM_TASK*sizeof(Task_Control_Block))));
    for(index_ControlBlock=0U;index_ControlBlock < MAX_NUM_TASK;index_ControlBlock++)
    {
      TaskControlBlock[index_ControlBlock].Task_ID                      = TaskConfigInitial->ptr_Task[index_ControlBlock].Task_ID;
      TaskControlBlock[index_ControlBlock].Task_Priority                = TaskConfigInitial->ptr_Task[index_ControlBlock].Task_Priority;
      TaskControlBlock[index_ControlBlock].Task_State                   = SUSPENDED;
      TaskControlBlock[index_ControlBlock].Stack_Information.StartAddress = 0U;
      TaskControlBlock[index_ControlBlock].Stack_Information.EndAddress   = 0U;
      TaskControlBlock[index_ControlBlock].Task_Deadline.Relative        = 0U;
      TaskControlBlock[index_ControlBlock].Task_Deadline.Absolute        = 0U;
    }

    while(1)
    {
      SchM_Background();
      _FEED_COP();
    }

}
```

Figure 14

Test Case	ID	Status
Priority	**2.8**	**Done**
Requirements covered		
2.14, 2.15		
Test Procedure		
Assign the lowest priority value to the lowest priority task, assign the same priority to a few tasks		
Expected Results		
Task execution can be performed according its priority assigned even if they have the same priority		

Actual Results	Test Results
Tasks are executed according its priority no matter whether or not they have the same priority	PASS
Comments	
Task priorities are configured according requirement 2.14 initially, but tasks sharing the same priority according requirement 2.15, are fully supported. See Figure 15	

Os_TaskCfg.c

```
/*****************************************************************
*   Global Variable Definitions
*****************************************************************
const Task_Descriptor TaskInitial[]=
{
  {TASK_1MS, PRIORITY_5, MASK_1MS, OFFSET_00,0,0,Task_1ms},
  {TASK_4MS, PRIORITY_4, MASK_4MS, OFFSET_01,0,0,Task_4ms},
  {TASK_8MS, PRIORITY_3, MASK_8MS, OFFSET_03,0,0,Task_8ms},
  {TASK_16MS,PRIORITY_2,MASK_16MS, OFFSET_05,0,0,Task_16ms},
  {TASK_32MS,PRIORITY_1,MASK_32MS, OFFSET_07,0,0,Task_32ms},
  {TASK_64MS,PRIORITY_0,MASK_64MS, OFFSET_11,0,0,Task_64ms}
};
```

Figure 15

3. Integration Test Cases

Test Case	ID	Status
Running project	**2.9**	**Done**
Requirements covered		
2.16, 2.17		
Test Procedure		
Set a pin level to high when tasks start its execution. Set a pin level low when tasks end its execution.		
Expected Results		
Pin out level must be set high every time tasks start its execution, and must be set low every time the tasks ends its execution.		
Actual Results		**Test Results**
See the next figures which contains different configurations for the task properties.		PASS
Comments		
Each result is composed with two images, the first shows the configuration of the task and the second shows the result and order in which the task were executed.		

Next figures show each result.

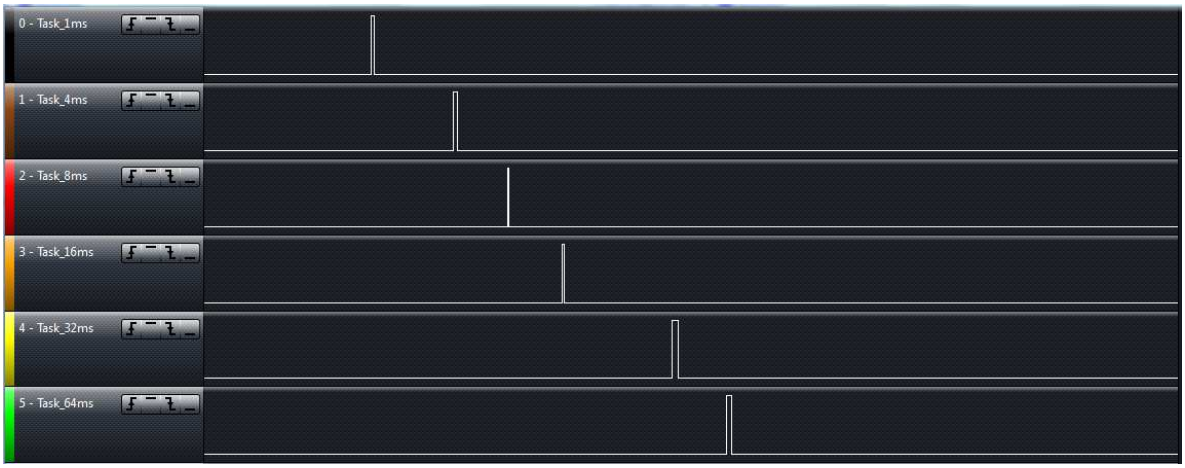
```
const Task_Descriptor TaskInitial[]=
{
  {TASK_1MS, PRIORITY_0, MASK_32MS, OFFSET_00,0,0,Task_1ms},
  {TASK_4MS, PRIORITY_1, MASK_32MS, OFFSET_00,0,0,Task_4ms},
  {TASK_8MS, PRIORITY_2, MASK_32MS, OFFSET_00,0,0,Task_8ms},
  {TASK_16MS,PRIORITY_3,MASK_32MS, OFFSET_00,0,0,Task_16ms},
  {TASK_32MS,PRIORITY_4,MASK_32MS, OFFSET_00,0,0,Task_32ms},
  {TASK_64MS,PRIORITY_5,MASK_32MS, OFFSET_00,0,0,Task_64ms}
};
```

Code Test 1

Result Test 1

```
const Task_Descriptor TaskInitial[]=
{
  {TASK_1MS, PRIORITY_0, MASK_32MS, OFFSET_00,0,0,Task_1ms},
  {TASK_4MS, PRIORITY_1, MASK_32MS, OFFSET_03,0,0,Task_4ms},
  {TASK_8MS, PRIORITY_2, MASK_32MS, OFFSET_05,0,0,Task_8ms},
  {TASK_16MS,PRIORITY_3,MASK_32MS, OFFSET_07,0,0,Task_16ms},
  {TASK_32MS,PRIORITY_4,MASK_32MS, OFFSET_11,0,0,Task_32ms},
  {TASK_64MS,PRIORITY_5,MASK_32MS, OFFSET_13,0,0,Task_64ms}
};
```

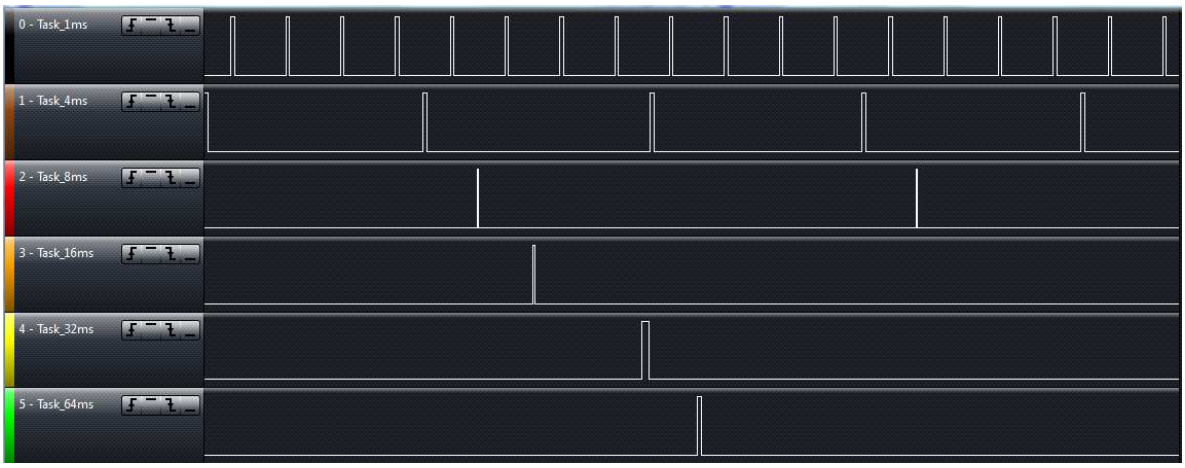
Code Test 2



Result Test 2

```
const Task_Descriptor TaskInitial[]=
{
  {TASK_1MS, PRIORITY_0, MASK_1MS, OFFSET_00,0,0,Task_1ms},
  {TASK_4MS, PRIORITY_1, MASK_4MS, OFFSET_03,0,0,Task_4ms},
  {TASK_8MS, PRIORITY_2, MASK_8MS, OFFSET_05,0,0,Task_8ms},
  {TASK_16MS,PRIORITY_3,MASK_16MS, OFFSET_07,0,0,Task_16ms},
  {TASK_32MS,PRIORITY_4,MASK_32MS, OFFSET_11,0,0,Task_32ms},
  {TASK_64MS,PRIORITY_5,MASK_64MS, OFFSET_13,0,0,Task_64ms}
};
```

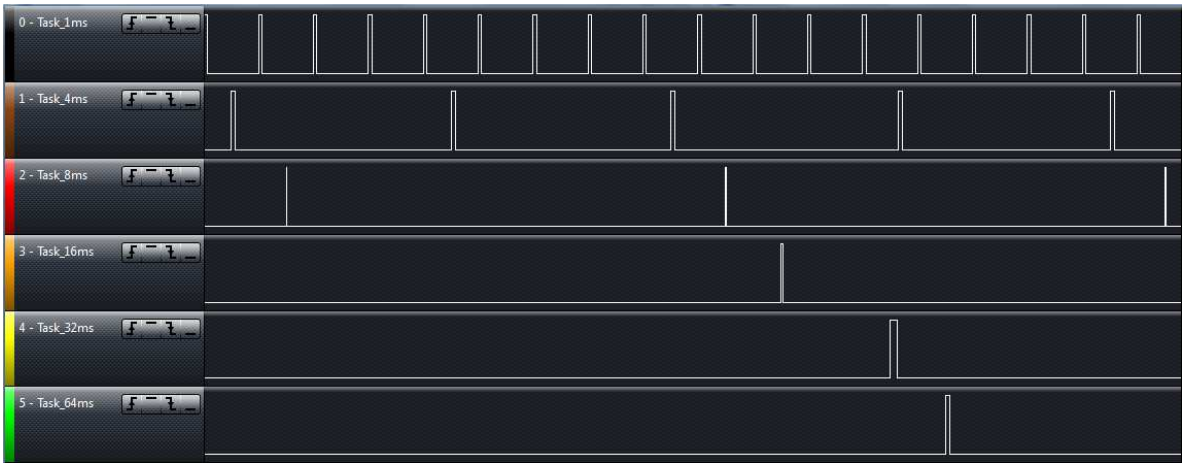
Code Test 3



Result Test 3

```
const Task_Descriptor TaskInitial[]=
{
  {TASK_1MS, PRIORITY_5, MASK_1MS, OFFSET_00,0,0,Task_1ms},
  {TASK_4MS, PRIORITY_1, MASK_4MS, OFFSET_03,0,0,Task_4ms},
  {TASK_8MS, PRIORITY_2, MASK_8MS, OFFSET_05,0,0,Task_8ms},
  {TASK_16MS,PRIORITY_3,MASK_16MS, OFFSET_07,0,0,Task_16ms},
  {TASK_32MS,PRIORITY_2,MASK_32MS, OFFSET_11,0,0,Task_32ms},
  {TASK_64MS,PRIORITY_3,MASK_64MS, OFFSET_13,0,0,Task_64ms}
};
```

Code Test 4



Result Test 4

Test Case	**ID**	**Status**
Running project	**2.10**	**Done**
Requirements covered		
2.18		
Test Procedure		

Define the basic states Suspended, Ready & Running for all Tasks States.	
Expected Results	
Tasks states will be transitioning the basic states Suspended, Ready & Running.	
Actual Results	**Test Results**
See Figure 16 & Figure 17.	PASS
Comments	
Tasks will be set in suspended state every time they're called to terminate in TerminateTask API. Tasks in ready state will be set every time they're put into the priority buffer by ActivateTask API. Tasks in running state are set every time they're called inside the Dispatcher API.	

```
/*****************************************************************************************************
*   Function: TerminateTask(void)
*
*   Description: This function terminates the task that invoked the service,
*   transferring it from the RUNNING to the SUSPENDED state, and move the task out
*   of the priority buffer queue
*
*   Caveats:
*****************************************************************************************************
Status_Type TerminateTask (void)
{
  Status_Type Status = E_OK;
  u16 IndexQueue=0;

  TaskControlBlock[TaskExecuted_ID].Task_State = SUSPENDED;
  for(IndexQueue;IndexQueue < CNF_MAXTASKQUEUE;IndexQueue++)
  {
     DispacherArray[TaskConfigInitial->ptr_Task[TaskExecuted_ID].Task_Priority][IndexQueue] =
     DispacherArray[TaskConfigInitial->ptr_Task[TaskExecuted_ID].Task_Priority][IndexQueue+1];
  }
  DispacherArray[TaskConfigInitial->ptr_Task[TaskExecuted_ID].Task_Priority][IndexQueue-1] = 0xFFFF;
  TaskExecuted_ID = 0xFFFF;
  return Status;
}
```

Figure 16

```
/*****************************************************************************************************
*   Function: ActivateTask(TaskType taskID)
*
*   Description: This API service moves the task identified by the input
*   taskID from the SUSPENDED state to the READY state if
*   no error occurred, and put the task into the priority buffer
*   queue tail
*
*   Caveats:
*****************************************************************************************************
Status_Type ActivateTask(TaskType taskID)
{
  u16 index = 0;
  Status_Type Status = E_OK;
  u8 Processing_Done = FALSE;
  for(index = 0; index < CNF_MAXTASKQUEUE; index++)
  {
    if(Processing_Done == FALSE)
    {
      if(DispacherArray[TaskConfigInitial->ptr_Task[taskID].Task_Priority][index] == 0xFFFF)
      {
         DispacherArray[TaskConfigInitial->ptr_Task[taskID].Task_Priority][index] = taskID;

         Processing_Done = TRUE;
      }
    }
  }
  TaskControlBlock[taskID].Task_State = READY;
  if(Processing_Done == FALSE)
  {
     Status = E_OS_LIMIT;
  }
  return Status;
}
```

```
/*****************************************************************************
*   Function: Dispatcher
*
*   Description: Only run in Background task
*
*   Caveats: Non Reentrant
*****************************************************************************/
void Dispatcher(void)
{
  u16 IndexPriority = MAX_PRIORITY;
  u16 NoTaskExecuted = TRUE;
  do
  {
     IndexPriority--;
     if(DispacherArray[IndexPriority][0U] != 0xFFFF)
     {
          TaskExecuted_ID = DispacherArray[IndexPriority][0U];
          TaskControlBlock[TaskExecuted_ID].Task_State = RUNNING;

          TaskConfigInitial->ptr_Task[TaskExecuted_ID].TaskCallback();
          NoTaskExecuted = FALSE;
     }
  }
  while((IndexPriority != 0U)&&(NoTaskExecuted));
}
```

Figure 17

Test Case	**ID**	**Status**
Running project	**2.11**	**Done**
Requirements covered		
2.19		
Test Procedure		
Place ready tasks IDs into its correspondent priority buffer.		
Expected Results		
Tasks IDs are placed into its correspondent priority buffer every time they turn ready.		
Actual Results		**Test Results**
Tasks IDs are placed into its correspondent priority buffer every time they turn ready. As figure 18 and 19.		PASS
Comments		

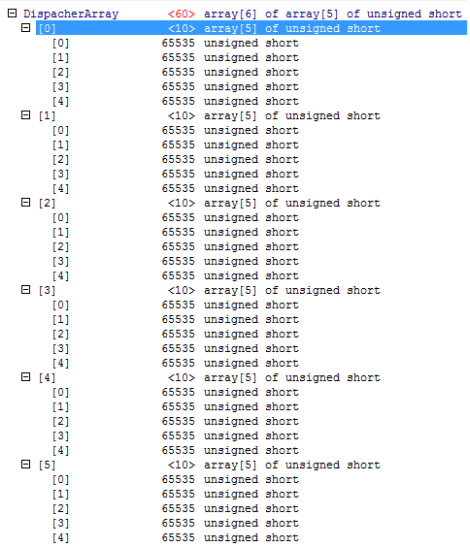


Figure 18

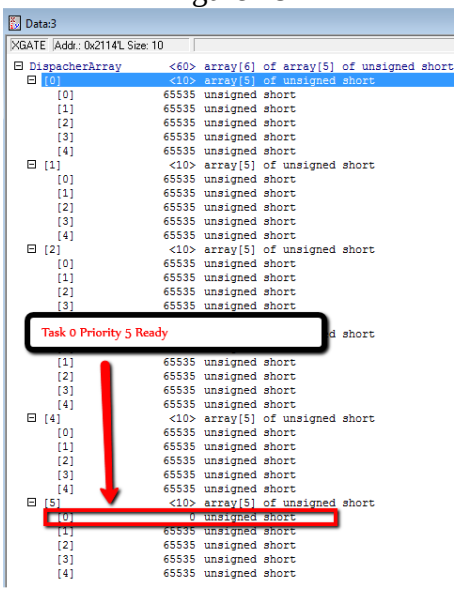


Figure 19

Test Case	ID	Status
Running project	2.12	Done
Requirements covered		
2.20		
Test Procedure		
Move out tasks which transition from ready to suspended from its correspondent priority buffer		
Expected Results		
Tasks to be suspended are moved out from its last priority buffer.		
Actual Results		**Test Results**
Tasks to be suspended are moved out from its last priority buffer. As figure 20 and 21		PASS
Comments		

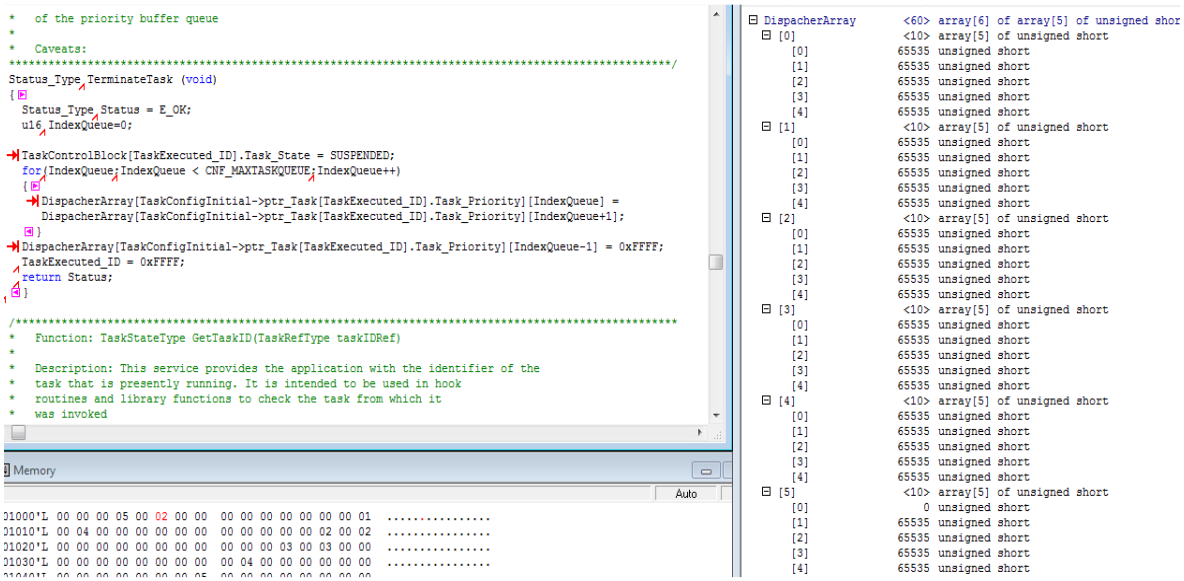


Figure 20

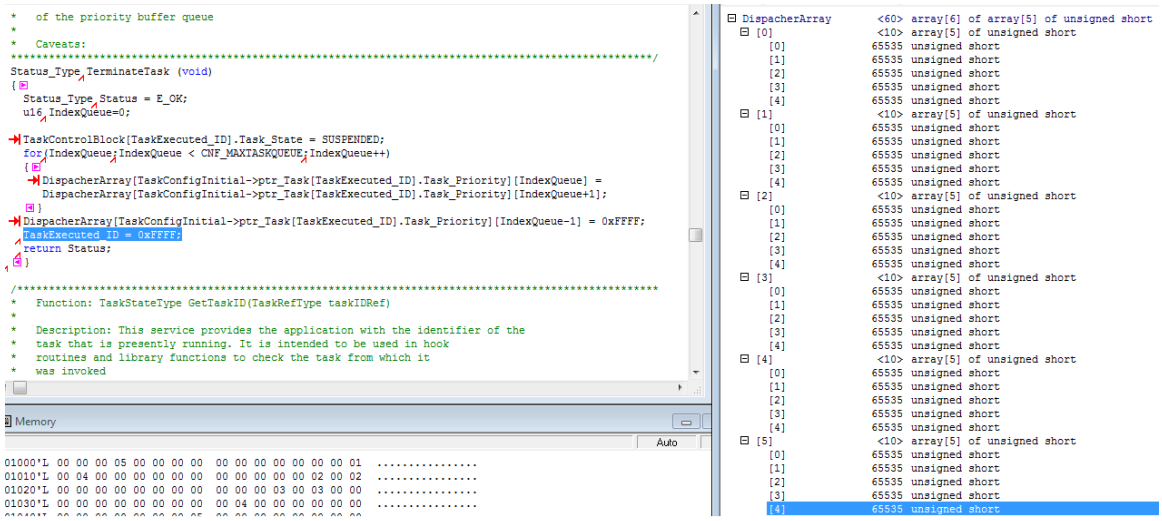


Figure 21

Test Case	**ID**	**Status**
Internal requirements	**2.13**	**Done**
Requirements covered		
2.21		
Test Procedure		
Include memory allocation driver from previous projects.		
Expected Results		
Memory allocation will be available for usage in Task manager project.		
Actual Results		**Test Results**
Memory allocation will be available for usage in Task manager project. As figure 22 and 23		PASS
Comments		

```
void Mem_Init(void)
{

    u16 i = 0;


    /* Head initial address pointer */
    mcal_mem_MemHandler.memStart = &mcal_mem_InitMem;
    /* Define Heap Memory Size */
    mcal_mem_TotalMemSizeBytes = (CNF_NO_OF_RAM_PAGES * CNF_PAGE_SIZE) - 1;
    /* Define Heap End Address */
    mcal_mem_MemHandler.memEnd = mcal_mem_TotalMemSizeBytes + mcal_mem_MemHandler.memStart;
    /* Current address points to heap's start address */
    mcal_mem_MemHandler.currAddr = mcal_mem_MemHandler.memStart;


    /* Set initial values of Heap memory */
    for(i=0;i < mcal_mem_TotalMemSizeBytes;i++)
    {
        *mcal_mem_MemHandler.currAddr = 0x00;
        mcal_mem_MemHandler.currAddr++;
    }

    /* Memory is empty */
    mcal_mem_MemHandler.memFull = FALSE;
    /* Current address points to Heap's start address */
    mcal_mem_MemHandler.currAddr = mcal_mem_MemHandler.memStart;

}
```

Figure 22

Figure 23

Test Case	**ID**	**Status**
Internal requirements	**2.14**	**Done**
Requirements covered		
2.22		
Test Procedure		
Reserve memory for control block structure using memory allocation.		
Expected Results		
Reserved memory section is initialized to 0 and its size is the control block structure size.		
Actual Results		**Test Results**

Reserved memory section is initialized to 0 and its size is the control block structure size. As figure 24 and 25	PASS
Comments	

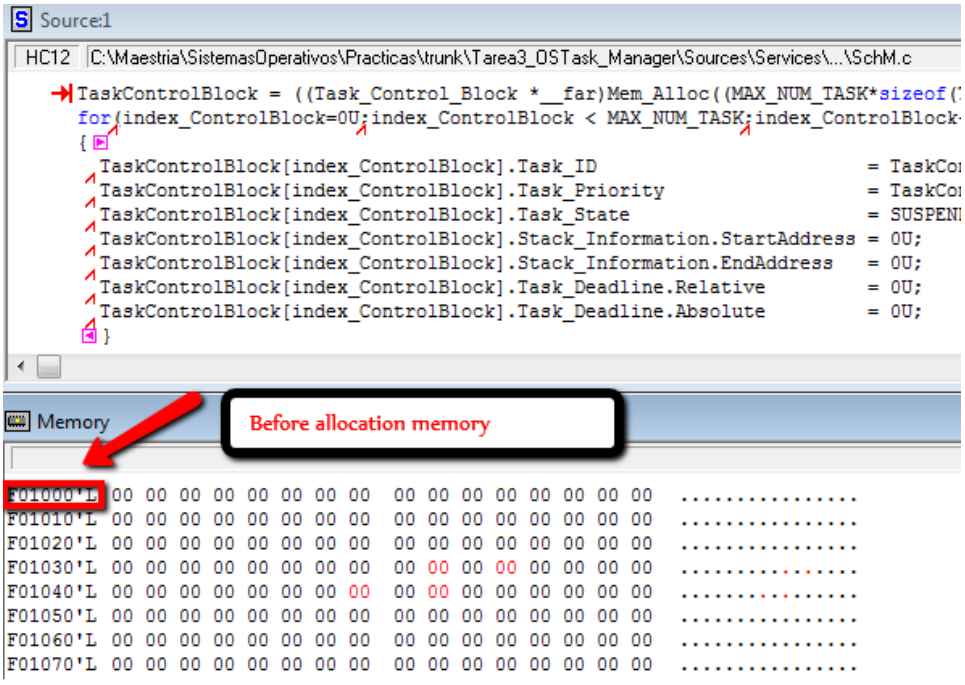


Figure 24

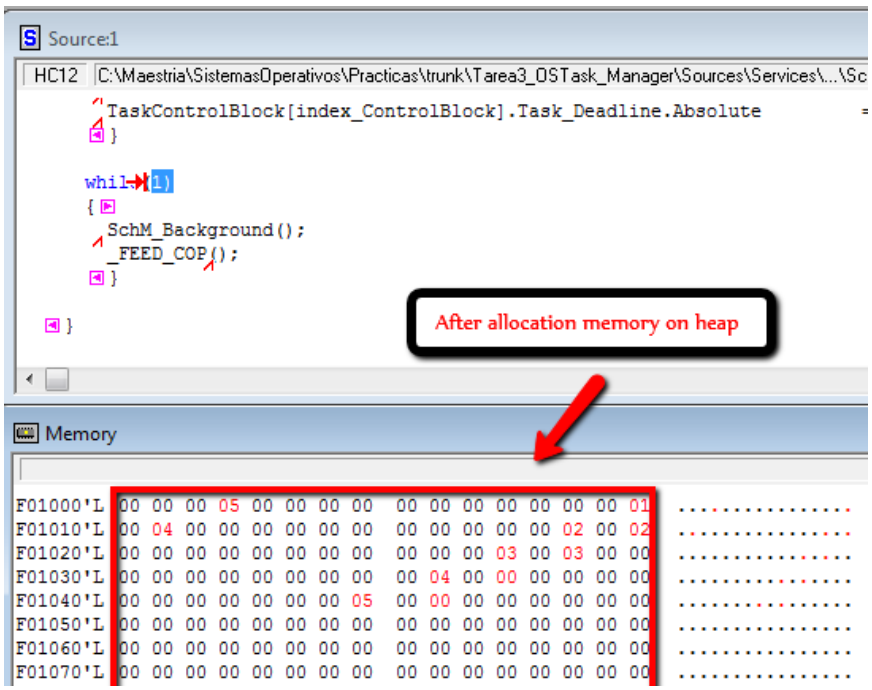


Figure 25

4. Scheduler Test Cases

Test Case	ID	Status
	1.0	
Requirements covered		
1.1		
Test Procedure		
Scheduler initialization shall be supported through SchM_Init API		
Expected Results		
Initialization of Scheduler via SchM_Init.		
Actual Results		**Test Results**
Scheduler is initialized by SchM_Init API		PASS
Comments		
Parameters are not supported currently since there not stack allocation. See Figure 1		

SchM.c

Path: C:\Users\Usuario\Documents\MDE\DSOAE\Projects\Tarea

```
/************************************************************
*   Function: SchM_Init(SchM_TaskConfigType *SchM_Config)
*
*   Description: Scheduler Initialization
*
*   Caveats:
************************************************************/
void SchM_Init(const SchM_TaskConfigType *SchM_Config)
{

    /*Initialize timer configuration for the OS tick*/
    Gpt_Init(&Gpt_ConfigType_initial[0]);

  /* Initialize Ostick and Scheduler flags */
    SchM_OSTickEnabled = SCHM_OSTICK_DISABLED;
    SchM_SchedulerEnabled =  SCHEDULER_ENABLED;
    SchM_OSTickCounter = 0;
}
```

Figure 1

Test Case	ID	Status
	1.1	
Requirements covered		
1.2		
Test Procedure		
Scheduler De-Initialization shall be supported through SchM_DeInit API		
Expected Results		
De-Initialization of Scheduler via SchM_DeInit		

Actual Results	Test Results
Scheduler is De-Initialized by SchM_DeInit API	PASS
Comments	
Only De-Initialization Scheduler flag is updated since there are not resources to de-initialize. See Figure 2	

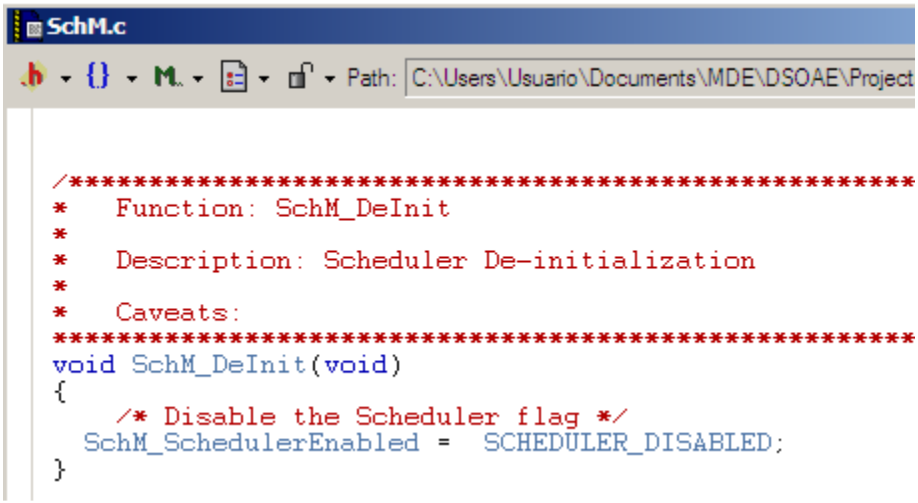


Figure 2

Test Case	ID	Status
	1.2	
Requirements covered		
1.3		
Test Procedure		
Scheduler Start shall be supported through SchM_Start API		
Expected Results		
Start of the Scheduler via SchM_Start API		
Actual Results		**Test Results**
Scheduler is started by SchM_Start API		PASS
Comments		
See Figure 3.		

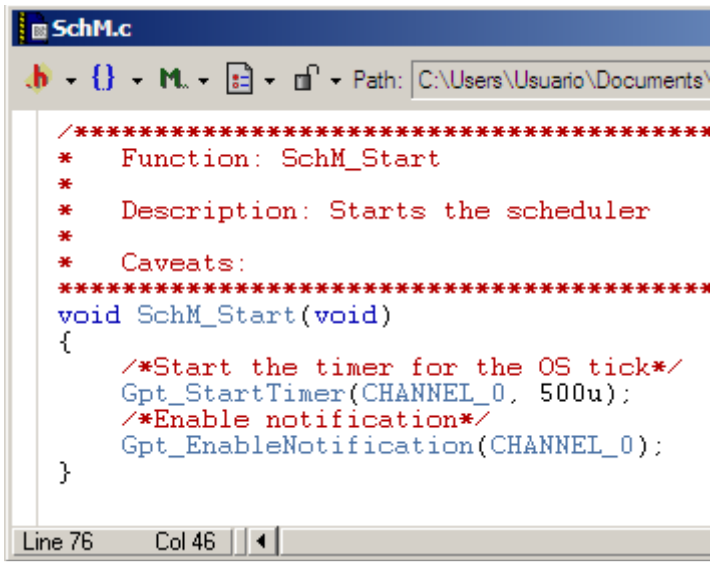


Figure 3

Test Case	**ID**	**Status**
	1.3	
Requirements covered		
1.4		
Test Procedure		
OS Tick Callback shall be supported through SchM_OsTick API		
Expected Results		
OS Tick Callback support via SchM_OsTick API		
Actual Results		**Test Results**
OS Tick Callback is supported by SchM_OsTick API		PASS
Comments		
See Figure 4		

SchM.c

Path: C:\Users\Usuario\Documents\MDE\DSOAE\Projects\Tarea2\Tarea2_OS_Scheduler\Source

```
/*******************************************************************************
*   Function: SchM_OsTick
*
*   Description: Callback handled once an interrupt is generated by Channel 0.
*
*   Caveats: Non Reentrant
*******************************************************************************/
void SchM_OsTick(void)
{

  SchM_OSTickCounter ++;

  if(!SchM_OSTickEnabled)
  {
    SchM_OSTickEnabled = SCHM_OSTICK_ENABLED;
  }
  else
  {
    /* ERROR: Ostick flag was not disabled */
  }
  PORTB_PB2= ~PORTB_PB2;
}
```

Figure 4

Test Case	**ID**	**Status**
	1.4	
Requirements covered		
1.5		
Test Procedure		
Background task shall be supported through SchM_Background API		
Expected Results		
Background task support via SchM_Background API		
Actual Results		**Test Results**
Background task is supported by SchM_Background API		PASS
Comments		
See Figure 5		

SchM.c

Path: C:\Users\Usuario\Documents\MDE\DSOAE\Projects\Tarea2\Tarea2_OS_Scheduler\Sources\Services\Syst...\SchM.c

```
*   Function: SchM_Background
*
*   Description: BackGround Task
*
*   Caveats: Non Reentrant
*******************************************************************************************************
void SchM_Background(void)
{
   u8 Task_Index = 0;
   if(SchM_SchedulerEnabled)
   {
     if(SchM_OSTickEnabled)
     {
      PORTB_PB3= 1;
         while (Task_Index < SchM_TaskConfigInitial[0U].SchM_TaskNumberConfig)
         {
           if((SchM_OSTickCounter & SchM_TaskConfigInitial->ptr_SchM_Task[Task_Index].Mask) ==
               SchM_TaskConfigInitial->ptr_SchM_Task[Task_Index].Offset)
           {
             SchM_TaskConfigInitial->ptr_SchM_Task[Task_Index].SchM_TaskCallback();
           }
           Task_Index++;
         }
         /*Wait for the next OS Tick to enable it*/
         SchM_OSTickEnabled = SCHM_OSTICK_DISABLED;
      PORTB_PB3= 0;
     }
     else
     {
        /*For future purposes*/
     }
   }
   else{
     /*Scheduler is not enabled*/
   }
}
```

Figure 5

Test Case	ID	Status
	1.5	
Requirements covered		
1.6		
Test Procedure		
Callback functions shall be referred as per the task period: SchM_Task_##period(void) E.g. SchM_Task_1p56ms(void)		
Expected Results		
Callback functions referred as per the task period		
Actual Results		**Test Results**
Callback functions are referred as per the task period		PASS
Comments		
See Figure 6		

```
/************************************
*   Global Function Prototypes
************************************/
extern void SchM_Task_1ms(void);
extern void SchM_Task_4ms(void);
extern void SchM_Task_8ms(void);
extern void SchM_Task_16ms(void);
extern void SchM_Task_32ms(void);
extern void SchM_Task_64ms(void);
```

Figure 6

Test Case	**ID**	**Status**
	1.6	
Requirements covered		
1.9		
Test Procedure		
Scheduler Module Shall be allocated at BSW and Services layer from AUTOSAR		
Expected Results		
Scheduler module allocated at BSW and Services layer from AUTOSAR		
Actual Results		**Test Results**
Scheduler module is allocated at BSW and Services layer from AUTOSAR		PASS
Comments		
See Figure 7		

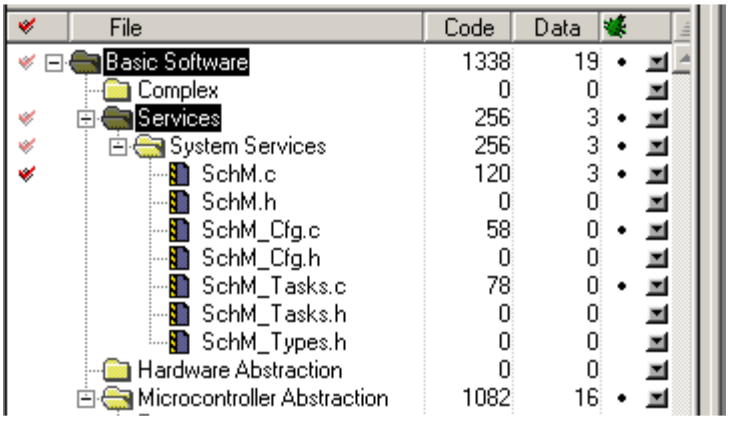


Figure 7

Test Case	ID	Status
	1.7	
Requirements covered		
1.11		
Test Procedure		
SchM_Cfg.c & SchM_Cfg.h shall provide the task configuration table		
Expected Results		
Task configuration table provided in SchM_Cfg.c & SchM_Cfg.h		
Actual Results		**Test Results**
Task configuration table is included in SchM_Cfg.c & SchM_Cfg.h		PASS
Comments		
See Figure 8 & Figure 9		

SchM_Cfg.c

Path: C:\Users\Usuario\Documents\MDE\DSOAE\Projects\Tarea2

```
*   Global Variable Definitions
*******************************************************************
const SchM_Task_type SchM_TaskInitial[]=
{
  {TASK_1MS,MASK_1MS, OFFSET_00,SchM_Task_1ms},
  {TASK_4MS,MASK_4MS, OFFSET_01,SchM_Task_4ms},
  {TASK_8MS,MASK_8MS, OFFSET_03,SchM_Task_8ms},
  {TASK_16MS,MASK_16MS,OFFSET_05,SchM_Task_16ms},
  {TASK_32MS,MASK_32MS,OFFSET_07,SchM_Task_32ms},
  {TASK_64MS,MASK_64MS,OFFSET_11,SchM_Task_64ms}
};

const SchM_TaskConfigType SchM_TaskConfigInitial[]=
{
  {
    sizeof(SchM_TaskInitial)/sizeof(SchM_TaskInitial[0]),
    &SchM_TaskInitial[0U]
  }
};
/*******************************************************************
```

Figure 8

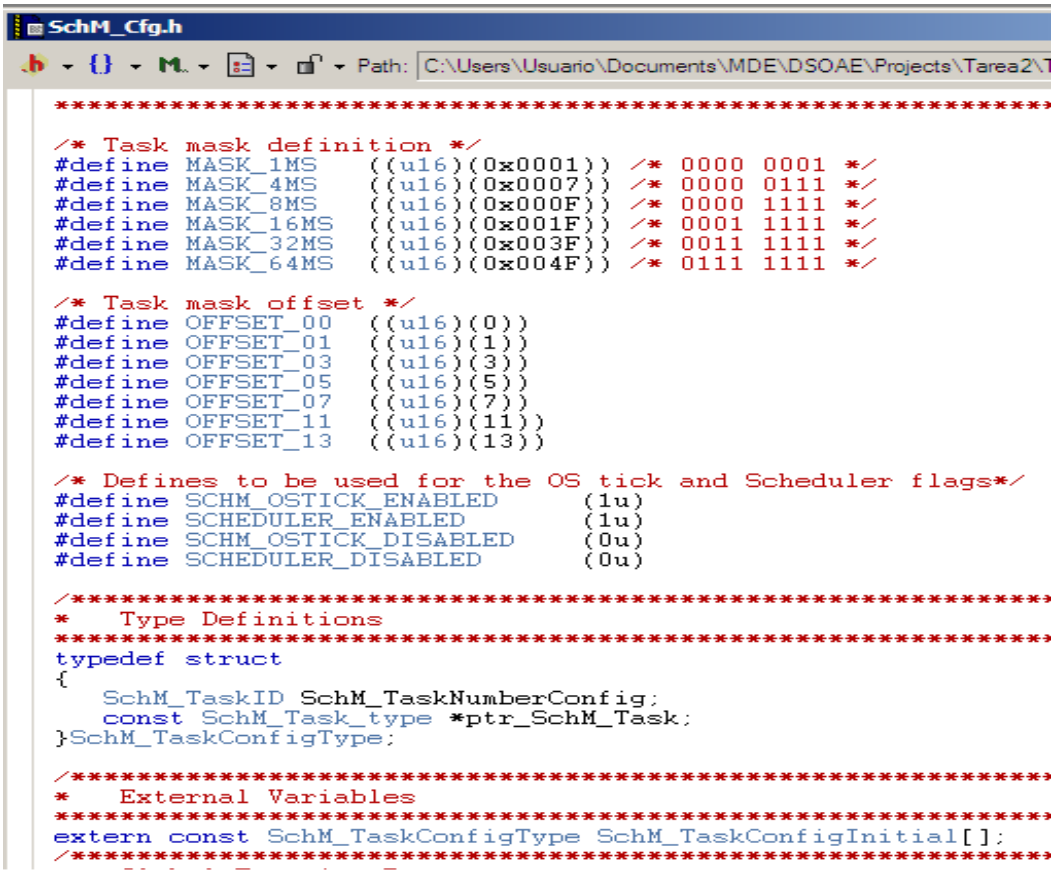


Figure 9

Test Case	ID	Status
	1.8	
Requirements covered		
1.12		
Test Procedure		
SchM.c & SchM.h shall provide the main functionality of the Scheduler. OS-Tick callback shall be allocated in these files		
Expected Results		
Main functionality placed in SchM.c & SchM.h		
Actual Results		**Test Results**
Main functionality is allocated in SchM.c & SchM.h		PASS
Comments		
See Figure 10, Figure 11 & Figure 12		

SchM.c

Path: C:\Users\Usuario\Documents\MDE\DSOAE\Projects\Tarea2\

```
/*****************************************************************
*   Function: SchM_Init(SchM_TaskConfigType *SchM_Config)
*
*   Description: Scheduler Initialization
*
*   Caveats:
*****************************************************************
void SchM_Init(const SchM_TaskConfigType *SchM_Config)
{

    /*Initialize timer configuration for the OS tick*/
    Gpt_Init(&Gpt_ConfigType_initial[0]);

  /* Initialize Ostick and Scheduler flags */
    SchM_OSTickEnabled = SCHM_OSTICK_DISABLED;
    SchM_SchedulerEnabled =  SCHEDULER_ENABLED;
    SchM_OSTickCounter = 0;
}


/*****************************************************************
*   Function: SchM_Init
*
*   Description: Scheduler De-initialization
*
*   Caveats:
*****************************************************************
void SchM_DeInit(void)
{
    /* Disable the Scheduler flag */
  SchM_SchedulerEnabled =  SCHEDULER_DISABLED;
}

/*****************************************************************
*   Function: SchM_Start
*
*   Description: Starts the scheduler
*
*   Caveats:
*****************************************************************
void SchM_Start(void)
{
    /*Start the timer for the OS tick*/
    Gpt_StartTimer(CHANNEL_0, 500u);
    /*Enable notification*/
    Gpt_EnableNotification(CHANNEL_0);
}

/*****************************************************************
*   Function: SchM_OsTick
*
*   Description: Callback handled once an interrupt is
*
*   Caveats: Non Reentrant
*****************************************************************
void SchM_OsTick(void)
{

  SchM_OSTickCounter ++;

  if(!SchM_OSTickEnabled)
  {
    SchM_OSTickEnabled = SCHM_OSTICK_ENABLED;
  }
  else
  {
    /* ERROR: Ostick flag was not disabled */
  }
  PORTB_PB2= ~PORTB_PB2;
}
```

Figure 10

SchM.c

Path: C:\Users\Usuario\Documents\MDE\DSOAE\Projects\Tarea2\Tarea2_OS_Scheduler\Sources\Services\System Ser...\SchM.c

```
/*****************************************************************************************************
*   Function: SchM_Background
*
*   Description: BackGround Task
*
*   Caveats: Non Reentrant
*****************************************************************************************************/
void SchM_Background(void)
{
   u8 Task_Index = 0;
   if(SchM_SchedulerEnabled)
   {
     if(SchM_OSTickEnabled)
     {
      PORTB_PB3= 1;
         while (Task_Index < SchM_TaskConfigInitial[0U].SchM_TaskNumberConfig)
         {
           if((SchM_OSTickCounter & SchM_TaskConfigInitial->ptr_SchM_Task[Task_Index].Mask) ==
               SchM_TaskConfigInitial->ptr_SchM_Task[Task_Index].Offset)
           {
             SchM_TaskConfigInitial->ptr_SchM_Task[Task_Index].SchM_TaskCallback();
           }
           Task_Index++;
         }
         /*Wait for the next OS Tick to enable it*/
         SchM_OSTickEnabled = SCHM_OSTICK_DISABLED;
      PORTB_PB3= 0;
     }
     else
     {
        /*For future purposes*/
     }
   }
   else{
     /*Scheduler is not enabled*/
   }
}
```

Figure 11

SchM.h

Path: C:\Users\Usuario\Documents\MDE\DSOAE\Projects\Tarea2\Tarea2_OS_Schedu...\SchM

```
#include "SchM_Tasks.h"
#include "SchM_Cfg.h"
/*****************************************************************************************************
*   Macro Definitions
*****************************************************************************************************/

/*****************************************************************************************************
*   Type Definitions
*****************************************************************************************************/

/*****************************************************************************************************
*   External Variables
*****************************************************************************************************/

/*****************************************************************************************************
*   Global Function Prototypes
*****************************************************************************************************/
/* vfnPIT0_Callback, function to be called upon timeout of PIT channel 0 */
extern void SchM_OsTick(void);
extern void SchM_Init(const SchM_TaskConfigType *SchM_Config);
extern void SchM_DeInit(void);
extern void SchM_Start(void);
extern void SchM_Background(void);
```

Figure 12

Test Case	ID	Status
	1.9	
Requirements covered		
1.13		
Test Procedure		
SchM_Types.h shall provide the Scheduler type definitions		
Expected Results		
Scheduler type definitions included in SchM_Types.h		
Actual Results		**Test Results**
Scheduler type definitions are included in SchM_Types.h		PASS
Comments		
See Figure 13		

SchM_Types.h

Path: C:\Users\Usuario\Documents\MDE\DSOAE

```
*    ------------    ------------    --------    ------
*    03-01-14         02              SPA         OS Ta
**************************************************

/**************************************************
*    Include Files
**************************************************
#include "typedefs.h"
/**************************************************
*    Macro Definitions
**************************************************
typedef u16 SchM_TaskOffsetType;
typedef u16 SchM_TaskMaskType;
/**************************************************
*    Type Definitions
**************************************************
typedef enum
{
   TASK_1MS = 0,
   TASK_4MS,
   TASK_8MS,
   TASK_16MS,
   TASK_32MS,
   TASK_64MS
}SchM_TaskID;

typedef struct
{
  SchM_TaskID          TaskID;
  SchM_TaskMaskType    Mask;
  SchM_TaskOffsetType  Offset;
  void(*SchM_TaskCallback)(void) ;
}SchM_Task_type;
```

Figure 13

Test Case	ID	Status
	1.10	
Requirements covered		
1.14		
Test Procedure		
SchM_Tasks.c & SchM_Tasks.h shall allocate the module's periodic tasks		
Expected Results		
Allocation of the periodic Tasks in SchM_Tasks.c & SchM_Tasks.h		
Actual Results		**Test Results**
Periodic Tasks are allocated in SchM_Tasks.c & SchM_Tasks.h		PASS
Comments		
See Figure 14, Figure 15 & Figure 16		

SchM_Tasks.c

Path: C:\Users\Usuario\Documents\MDE\DSOAE\Projects\Tarea2\Tare...\S

```
*   Function: SchM_Task_1ms
*
*   Description: Scheduler task callback function 1ms
*
*   Caveats: Non Reentrant
*******************************************************************
void SchM_Task_1ms(void)
{
  u8 index;
  PORTA_PA0= 1;

  for (index=0;index<100;index++){
  }
  PORTA_PA0= 0;
}
/*******************************************************************
*   Function: SchM_Task_4ms
*
*   Description: Scheduler task callback function 4ms
*
*   Caveats: Non Reentrant
*******************************************************************
void SchM_Task_4ms(void)
{
  u8 index;
  PORTA_PA1= 1;
  for (index=0;index<100;index++){
  }
  PORTA_PA1= 0;
}
/*******************************************************************
*   Function: SchM_Task_8ms
*
*   Description: Scheduler task callback function 8ms
*
*   Caveats: Non Reentrant
*******************************************************************
void SchM_Task_8ms(void)
{
  u8 index;
  PORTA_PA2= 1;
  for (index=0;index<100;index++){
  }
  PORTA_PA2= 0;
}
```

Figure 14

SchM_Tasks.c

Path: C:\Users\Usuario\Documents\MDE\DSOAE\Projects\Tarea2\Tare...\Sc

```
*   Function: SchM_Task_16ms
*
*   Description: Scheduler task callback function 16ms
*
*   Caveats: Non Reentrant
*******************************************************************************
void SchM_Task_16ms(void)
{
  u8 index;
  PORTA_PA3= 1;
  for (index=0;index<100;index++){
  }
  PORTA_PA3= 0;
}
/*******************************************************************************
*   Function: SchM_Task_32ms
*
*   Description: Scheduler task callback function 32ms
*
*   Caveats: Non Reentrant
*******************************************************************************
void SchM_Task_32ms(void)
{
  u8 index;
  PORTB_PB0= 1;
  for (index=0;index<100;index++){
  }
  PORTB_PB0= 0;
}

/*******************************************************************************
*   Function: SchM_Task_64ms
*
*   Description: Scheduler task callback function 64ms
*
*   Caveats: Non Reentrant
*******************************************************************************
void SchM_Task_64ms(void)
{
  u8 index;
  PORTB_PB1= 1;
  for (index=0;index<100;index++){
  }
  PORTB_PB1= 0;
}
```

Figure 15

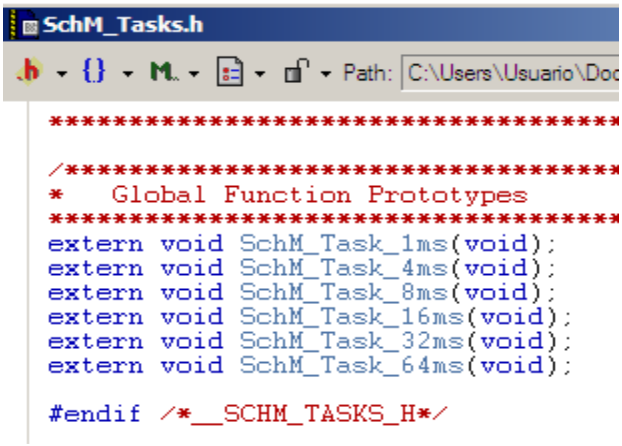
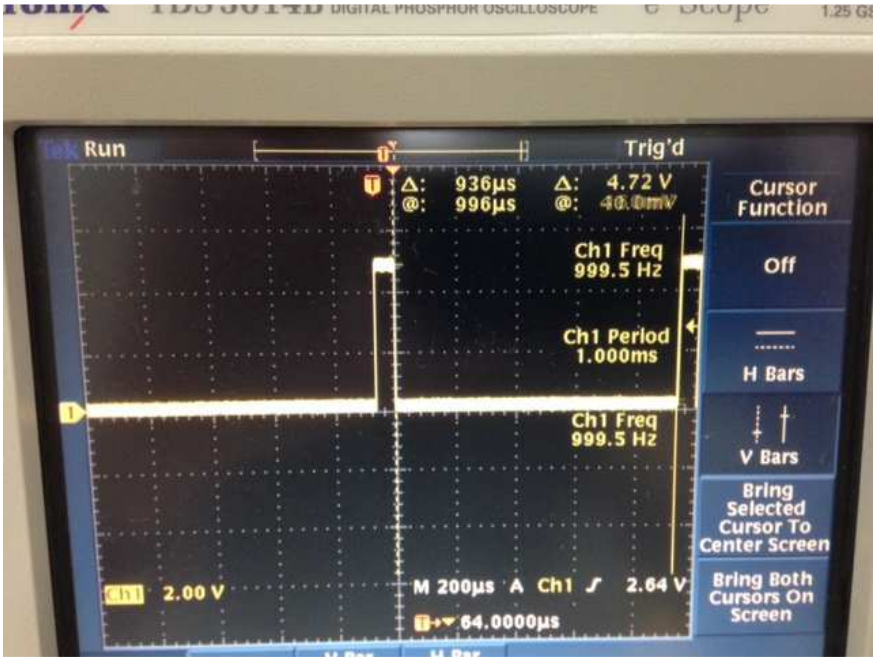


Figure 16

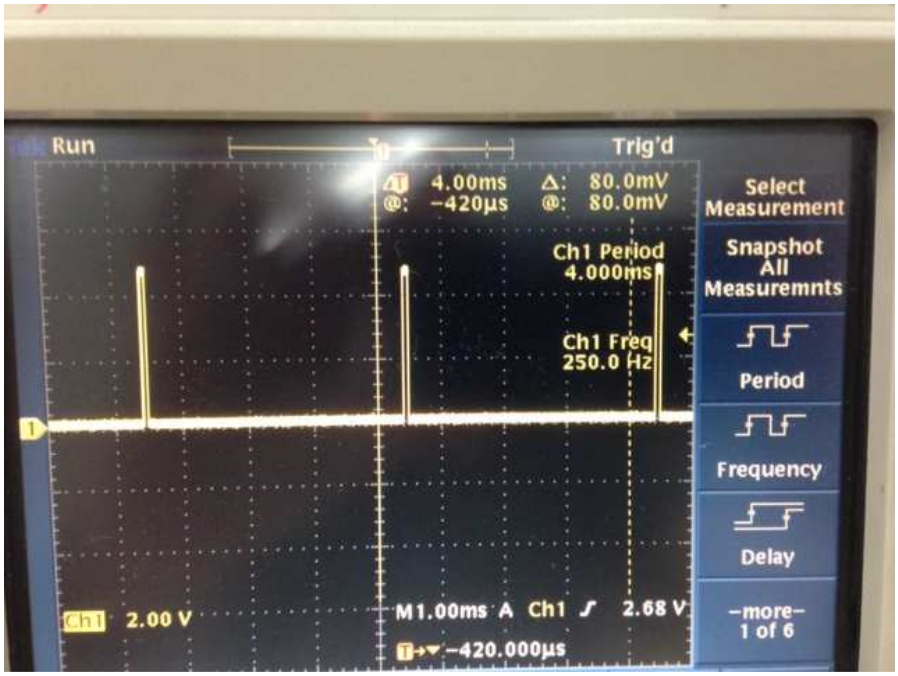
5. Integration Test Cases

Test Case	**ID**	**Status**
	1.11	
Requirements covered		
1.15		
Test Procedure		
Turn a Pin level ON at the entrance of a task and turn the Pin level OFF at the end of a task execution		
Expected Results		
Using an oscilloscope verify that every Task is working at the specified period		
Actual Results		**Test Results**
See figure 17		PASS
Comments		
As we could see in the next figure, there are different signals for each task, for the tick and for the background. We can see in the figure 17 how each task is activated at different time(1ms, 4ms, 8 ms, 16 ms, 32 ms and 64ms)		

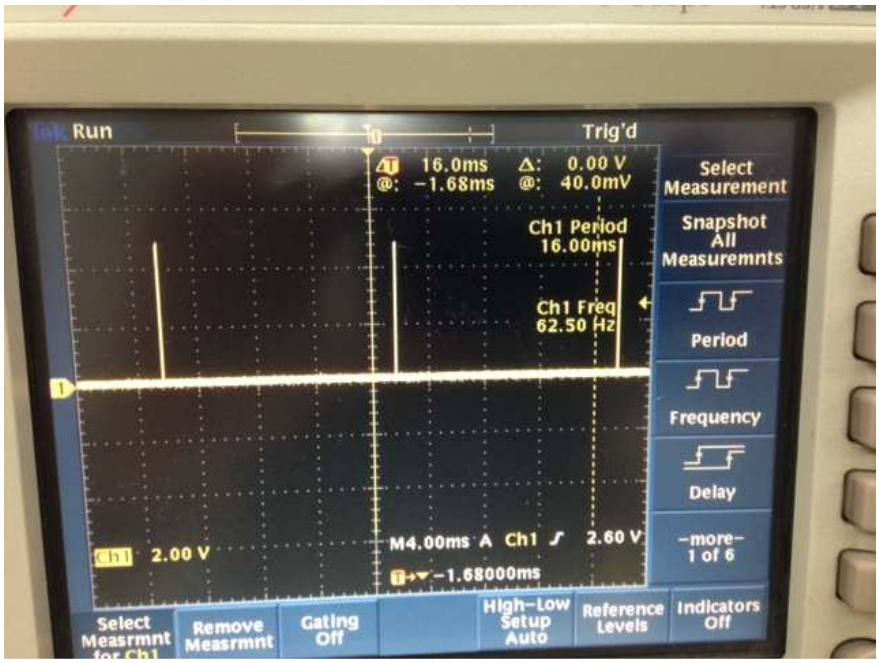
Next figures show each task period



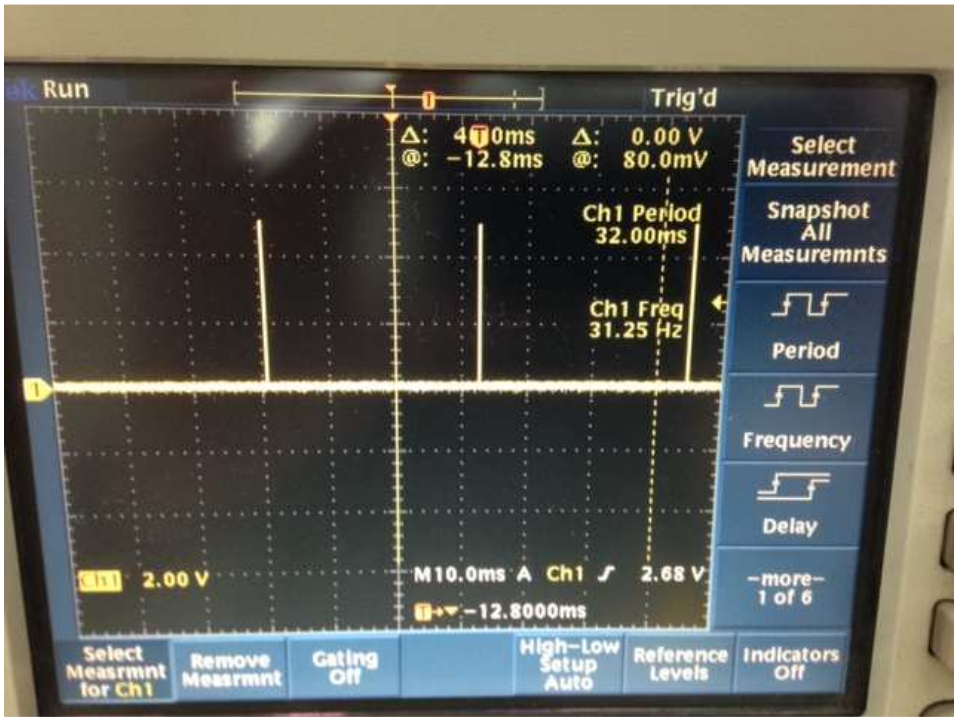
Task 1ms



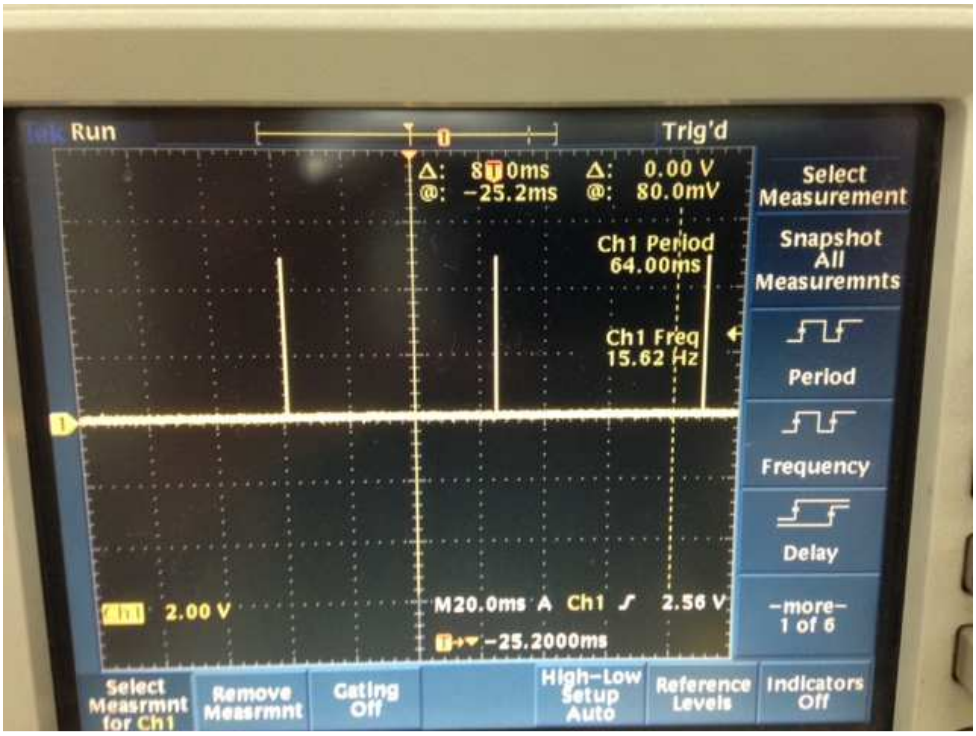
Task 4ms



Task 16ms



Task 32 ms



Task 64ms

Test Case	**ID**	**Status**
	1.12	
Requirements covered		
1.16		
Test Procedure		
Turn a pin level ON before entering the Background Task and turn the pin level OFF at the end of the Background Task		
Expected Results		
Using an oscilloscope verify the CPU Load at the specified pin level		
Actual Results		**Test Results**
See figure 17		PASS
Comments		
As we could see in the next figure, there are different signals for each task, for the tick and for the background. To consider that this test case passed, we could see that background signal is activated every time a task is executed. So, background signal is aligned with each width of each task.		

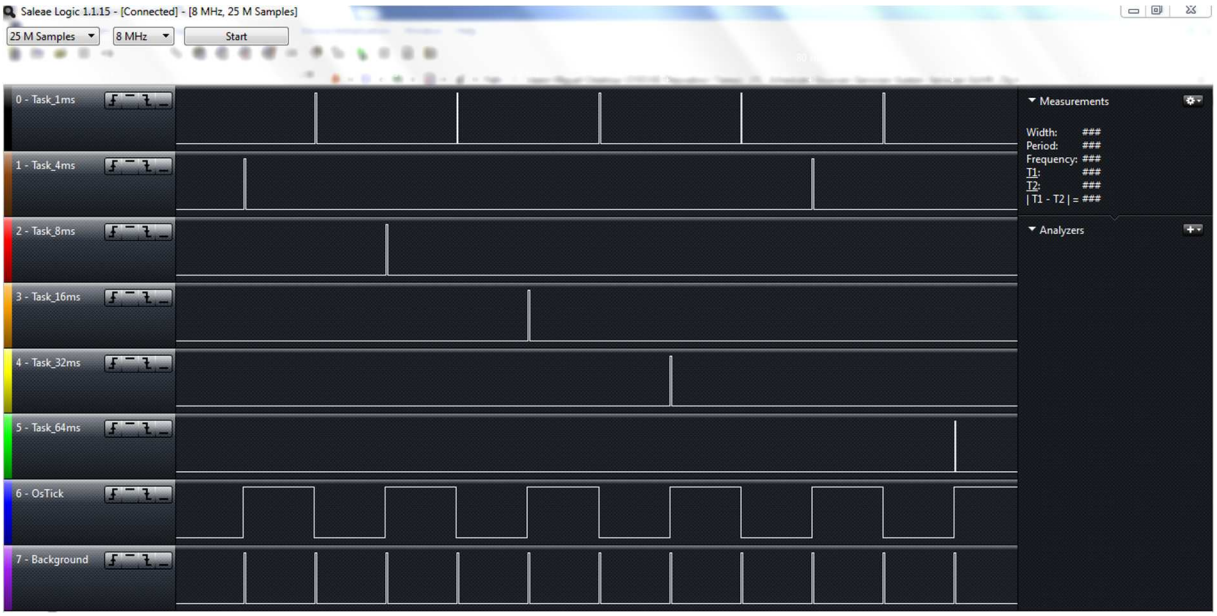


Figure 17

Test Case	**ID**	**Status**
	1.13	
Requirements covered		
1.17		
Test Procedure		
Modify the CPU Load by adding workload to the tasks		
Expected Results		
Using an oscilloscope verify the CPU Load at the specified pin level		
Actual Results		**Test Results**
See figure 18 and figure 19		PASS
Comments		
As explained in the Test case ID 1.12, each task has a row in the next figures 18 and 19. For these images we modified the workload of each task, you could see the differences in each task for both images. It can be clearly seen in the width of each pulse in each task. Again, in the background task we see how the width of the pulse of the activated task in that time is aligned with the width of the background task.		

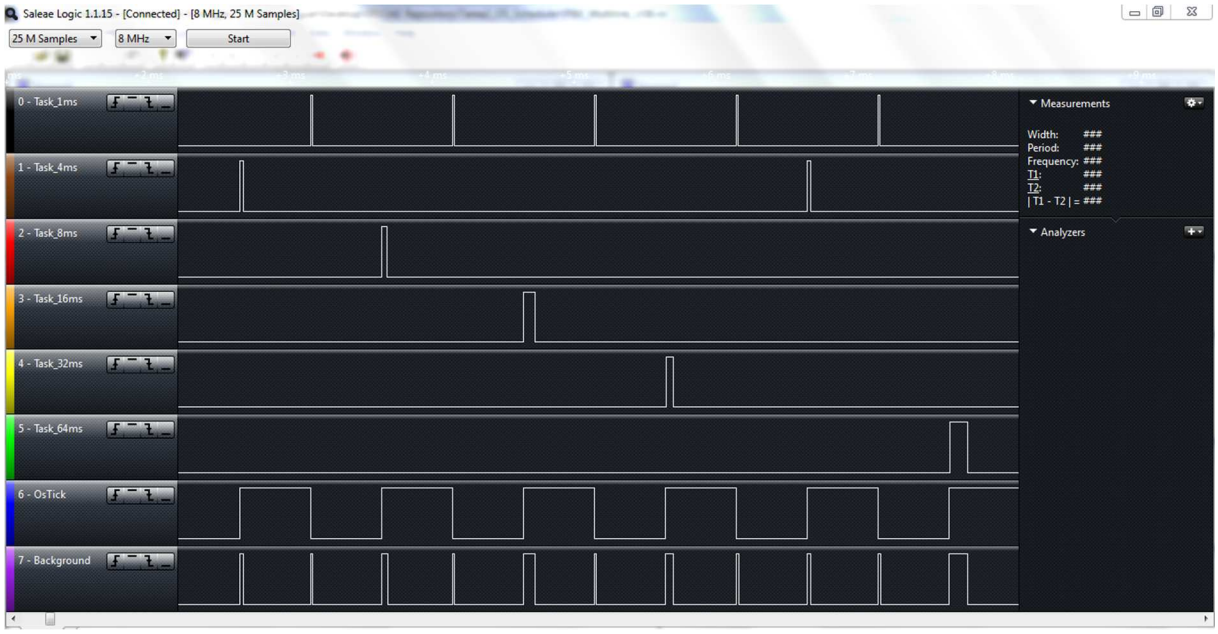


Figure 18

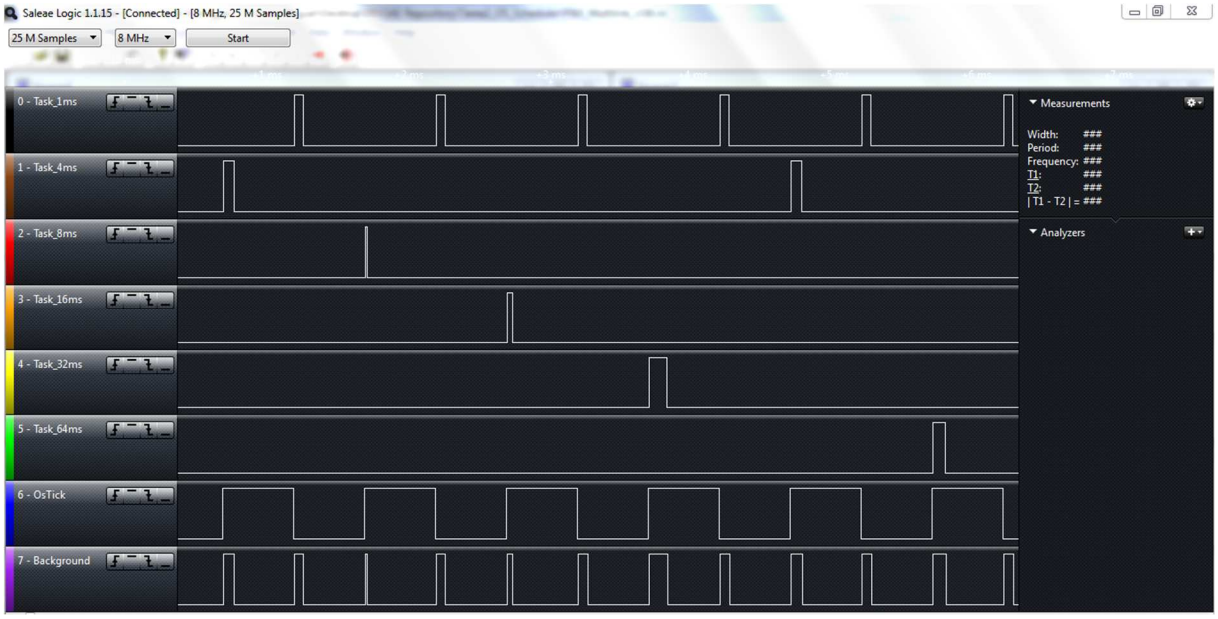


Figure 19

6. Task Manager Test Cases

Test Case	ID	Status
File Structure	**2.0**	**Done**
Requirements covered		
2.0		
Test Procedure		
Os folder will be provided in Services layer containing the following files: Os_TaskM.c/Os_TaskM.h, Os_TaskCfg.c/Os_TaskCfg.h, Os_Task.c/Os_Task.h, Os_Types.h		
Expected Results		
Os folder will be provided in Services layer.		
Actual Results		**Test Results**
Os folder is provided in Services layer. See Figure 1		PASS
Comments		

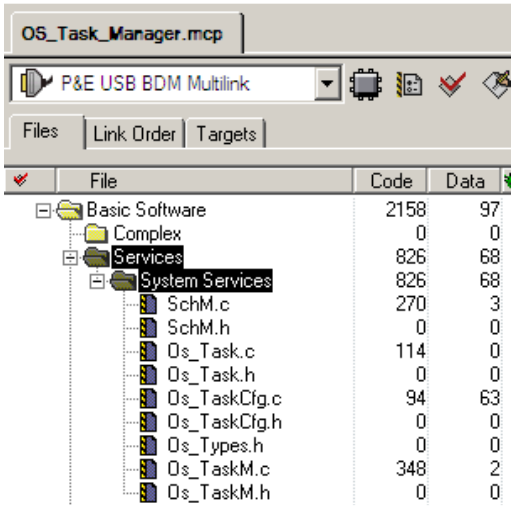


Figure 1

Test Case	ID	Status
Task Manager Services	**2.1**	**Done**
Requirements covered		
2.1		
Test Procedure		

Os_TaskM.c/Os_TaskM.h: provide the following Task Manager Services: AvtivateTask, TerminateTask, GetTaskID, GetTaskState	
Expected Results	
Task Manager Services included in Os_TaskM.c & Os_TaskM.h	
Actual Results	**Test Results**
Task Manager Services are provided in Os_TaskM.c & Os_TaskM.h, See Figure 2 & Figure 3	PASS
Comments	

Os_TaskM.c

```
/*****************************************************************************************************
*   Function: ActivateTask(TaskType taskID)
*
*   Description: This API service moves the task identified by the input
*   taskID from the SUSPENDED state to the READY state if
*   no error occurred, and put the task into the priority buffer
*   queue tail
*
*   Caveats:
*****************************************************************************************************
Status_Type ActivateTask(TaskType taskID)
{
  u16 index = 0;
  Status_Type Status = E_OK;
  u8 Processing_Done = FALSE;
  for(index = 0; index < CNF_MAXTASKQUEUE; index++)
  {
    if(Processing_Done == FALSE)
    {
      if(DispacherArray[TaskConfigInitial->ptr_Task[taskID].Task_Priority][index] == 0xFFFF)
      {
         DispacherArray[TaskConfigInitial->ptr_Task[taskID].Task_Priority][index] = taskID;

         Processing_Done = TRUE;
      }
    }
  }
  TaskControlBlock[taskID].Task_State = READY;
  if(Processing_Done == FALSE)
  {
     Status = E_OS_LIMIT;
  }
  return Status;
}

/*****************************************************************************************************
*   Function: TerminateTask(void)
*
*   Description: This function terminates the task that invoked the service,
*   transferring it from the RUNNING to the SUSPENDED state, and move the task out
*   of the priority buffer queue
*
*   Caveats:
*****************************************************************************************************
Status_Type TerminateTask (void)
{
  Status_Type Status = E_OK;
  u16 IndexQueue=0;

  TaskControlBlock[TaskExecuted_ID].Task_State = SUSPENDED;
  for(IndexQueue;IndexQueue < CNF_MAXTASKQUEUE;IndexQueue++)
  {
     DispacherArray[TaskConfigInitial->ptr_Task[TaskExecuted_ID].Task_Priority][IndexQueue] =
     DispacherArray[TaskConfigInitial->ptr_Task[TaskExecuted_ID].Task_Priority][IndexQueue+1];
  }
  DispacherArray[TaskConfigInitial->ptr_Task[TaskExecuted_ID].Task_Priority][IndexQueue-1] = 0xFFFF;
  TaskExecuted_ID = 0xFFFF;
  return Status;
}

/*****************************************************************************************************
*   Function: TaskStateType GetTaskID(TaskRefType taskIDRef)
*
*   Description: This service provides the application with the identifier of the
*   task that is presently running. It is intended to be used in hook
*   routines and library functions to check the task from which it
*   was invoked
*   Caveats:
*****************************************************************************************************

Status_Type GetTaskID(TaskRefType taskIDRef)
{
  Status_Type Status = E_OK;
  *taskIDRef = TaskExecuted_ID;
  return Status;
}

 /*****************************************************************************************************
 *   Function: TaskStateType GetTaskState(TaskType taskID, TaskStateRefType stateRef)
 *
 *   Description: This service identifies the current state of the task identified
 *   by the taskID parameter and places this value into the
 *   variable referenced by stateRef
 *   Caveats:
 *****************************************************************************************************

 Status_Type GetTaskState(TaskType taskID, TaskStateRefType stateRef)
 {
       Status_Type Status = E_OK;
       *stateRef = TaskControlBlock[taskID].Task_State;
       return Status;
 }
```

Figure 2

Os_TaskM.h

```
/*****************************************************************************
*   Global Function Prototypes
*****************************************************************************/

extern Status_Type ActivateTask(TaskType taskID);
extern Status_Type TerminateTask (void);
extern Status_Type GetTaskID(TaskRefType taskIDRef);
extern Status_Type GetTaskState(TaskType taskID, TaskStateRefType stateRef);
extern void Dispatcher(void);
```

Figure 3

Test Case	**ID**	**Status**
Configuration Support	**2.2**	**Done**
Requirements covered		
2.2		
Test Procedure		
Os_TaskCfg.c/Os_TaskCfg.h: provide OS tasks configuration support, these files will replace SchM_Cfg.c/SchM_Cfg.h which will not be provided		
Expected Results		
Os_TaskCfg.c/Os_TaskCfg.h: are used to provide OS tasks configuration support. SchM_Cfg.c/SchM_Cfg.h not available		
Actual Results		**Test Results**
Os_TaskCfg.c/Os_TaskCfg.h: provide OS tasks configuration support. SchM_Cfg.c/SchM_Cfg.h are not longer available.		PASS
Comments		
See Figure 4 & Figure 5.		

Os_TaskCfg.h

```
*   Macro Definitions
*************************************************************
/* Task mask definition */
#define MASK_1MS   ((u16)(0x0001)) /* 0000 0001 */
#define MASK_4MS   ((u16)(0x0007)) /* 0000 0111 */
#define MASK_8MS   ((u16)(0x000F)) /* 0000 1111 */
#define MASK_16MS  ((u16)(0x001F)) /* 0001 1111 */
#define MASK_32MS  ((u16)(0x003F)) /* 0011 1111 */
#define MASK_64MS  ((u16)(0x007F)) /* 0111 1111 */

/* Task mask offset */
#define OFFSET_00  ((u16)(0))
#define OFFSET_01  ((u16)(1))
#define OFFSET_03  ((u16)(3))
#define OFFSET_05  ((u16)(5))
#define OFFSET_07  ((u16)(7))
#define OFFSET_11  ((u16)(11))
#define OFFSET_13  ((u16)(13))

/* Defines to be used for the OS tick and Scheduler flags*/
#define SCHM_OSTICK_ENABLED     (1u)
#define SCHEDULER_ENABLED       (1u)
#define SCHM_OSTICK_DISABLED    (0u)
#define SCHEDULER_DISABLED      (0u)

#define CNF_MAXTASKQUEUE ((u16)(5))
/*************************************************************
*   Type Definitions
*************************************************************
typedef struct
{
   TaskID TaskNumberConfig;
   const Task_Descriptor *ptr_Task;
}TaskConfigType;

/*************************************************************
*   External Variables
*************************************************************
extern const TaskConfigType TaskConfigInitial[];
extern u16 DispacherArray[MAX_PRIORITY][CNF_MAXTASKQUEUE];
extern Task_Control_Block *__far TaskControlBlock;
```

Figure 4

Os_TaskCfg.c

```
#include "Os_TaskCfg.h"
#include "Os_Task.h"
/*****************************************************************
*   Local Macro Definitions
*****************************************************************/

/*****************************************************************
*   Local Type Definitions
*****************************************************************/

/*****************************************************************
*   Local Function Declarations
*****************************************************************/

/*****************************************************************
*   Global Variable Definitions
*****************************************************************/
const Task_Descriptor TaskInitial[]=
{
  {TASK_1MS, PRIORITY_5, MASK_1MS, OFFSET_00,0,0,Task_1ms},
  {TASK_4MS, PRIORITY_4, MASK_4MS, OFFSET_01,0,0,Task_4ms},
  {TASK_8MS, PRIORITY_2, MASK_8MS, OFFSET_03,0,0,Task_8ms},
  {TASK_16MS,PRIORITY_3,MASK_16MS, OFFSET_05,0,0,Task_16ms},
  {TASK_32MS,PRIORITY_0,MASK_32MS, OFFSET_07,0,0,Task_32ms},
  {TASK_64MS,PRIORITY_0,MASK_64MS, OFFSET_11,0,0,Task_64ms}
};

const TaskConfigType TaskConfigInitial[]=
{
  {
    sizeof(TaskInitial)/sizeof(TaskInitial[0]),
    &TaskInitial[0U]
  }
};

u16 DispacherArray[MAX_PRIORITY][CNF_MAXTASKQUEUE];

Task_Control_Block *__far TaskControlBlock;
```

Figure 5

Test Case	**ID**	**Status**
Task Configuration	**2.3**	**Done**
Requirements covered		
2.3, 2.7, 2.9		
Test Procedure		
Os_Task.c/Os_Task.h: provide the configured tasks to the application replacing SchM_Task.c/SchM_Task.h		
Expected Results		
Configured tasks are provided in Os_Task.c & Os_Task.h		
Actual Results		**Test Results**
SchM_Task.c/SchM_Task.h are not longer available and Os_Task.c & Os_Task.h provide tasks configuration.		PASS
Comments		
See Figure 4 & Figure 5		

Os_Task.c

```
/*************************************************************
*   Function: Task_1ms
*
*   Description: Scheduler task callback function 1ms
*
*   Caveats: Non Reentrant
*************************************************************/
TASK (Task_1ms)
{
  u16 index;
  Status_Type Error_Result = E_OK;
  PORTA_PA0= 1;

  for (index=0;index<500;index++){
  }
  PORTA_PA0= 0;
  Error_Result=TerminateTask();
}

/*************************************************************
*   Function: Task_4ms
*
*   Description: Scheduler task callback function 4ms
*
*   Caveats: Non Reentrant
*************************************************************/
TASK (Task_4ms)
{
  u16 index;
  Status_Type Error_Result = E_OK;
  PORTA_PA1= 1;
  for (index=0;index<600;index++){
  }
  PORTA_PA1= 0;
  Error_Result=TerminateTask();
}
/*************************************************************
*   Function: Task_8ms
*
*   Description: Scheduler task callback function 8ms
*
*   Caveats: Non Reentrant
*************************************************************/
TASK (Task_8ms)
{
  u16 index;
  Status_Type Error_Result = E_OK;
  PORTA_PA2= 1;
  for (index=0;index<100;index++){
  }
  PORTA_PA2= 0;
  Error_Result=TerminateTask();
}

/*************************************************************
*   Function: Task_16ms
*
*   Description: Scheduler task callback function 16ms
*
*   Caveats: Non Reentrant
*************************************************************/
TASK (Task_16ms)
{
  u16 index;
  Status_Type Error_Result = E_OK;
  PORTA_PA3= 1;
  for (index=0;index<300;index++){
  }
  PORTA_PA3= 0;
  TerminateTask();
}
```

```
/*****************************************************
*   Function: Task_32ms
*
*   Description: Scheduler task callback function 32ms
*
*   Caveats: Non Reentrant
*****************************************************
TASK (Task_32ms)
{
  u16 index;
  Status_Type Error_Result = E_OK;
  PORTB_PB0= 1;
  for (index=0;index<1000;index++){
  }
  PORTB_PB0= 0;
  Error_Result=TerminateTask();
}

/*****************************************************
*   Function: Task_64ms
*
*   Description: Scheduler task callback function 64ms
*
*   Caveats: Non Reentrant
*****************************************************
TASK (Task_64ms)
{
  u16 index;
  Status_Type Error_Result = E_OK;
  PORTB_PB1= 1;
  for (index=0;index<700;index++){
  }
  PORTB_PB1= 0;
  Error_Result=TerminateTask();
}
```

Figure 4

Os_Task.h

```
/*********************************
*   Global Function Prototypes
*********************************
DeclareTask(Task_1ms);
DeclareTask(Task_4ms);
DeclareTask(Task_8ms);
DeclareTask(Task_16ms);
DeclareTask(Task_32ms);
DeclareTask(Task_64ms);

#endif /*__OS_TASK_H*/
```

Figure 5

Test Case	**ID**	**Status**
Data definitions	**2.4**	**Done**
Requirements covered		
2.4, 2.8, 2.10		
Test Procedure		
Provide the required data definitions for Task Management services support in Os_Types.h and replace SchM_Types.h		
Expected Results		
Data definitions for Task Management services support is provided in Os_Types.h		
Actual Results		**Test Results**
Os_Types.h is not longer available and all Data definitions for Task Management services are supported in Os_Types.h		PASS
Comments		
See Figure 6		

Os_Types.h

```
*   Macro Definitions
*************************************************************

#ifndef TASK
#define TASK(taskID) void taskID(void)
#endif

#define DeclareTask(taskID) extern TASK(taskID)

/*************************************************************
*   Type Definitions
*************************************************************
/* Identifies a task state */
typedef u8 TaskStateType;

/* Identifies a task, commonly associated with the TaskID */
typedef u16 TaskType;

/* Reference to a variable of type TaskOffsetType */
typedef u16 TaskOffsetType;

/* Reference to a variable of type TaskMaskType */
typedef u16 TaskMaskType;

/* Reference to a variable of type TaskStackSize */
typedef u16 TaskStackSize;

/* Reference to a variable of type TaskRelDeadline */
typedef u16 TaskRelDeadline;

/* Reference to a variable of type TaskType */
typedef TaskType* TaskRefType;

/* Reference to a variable of type TaskStateType */
typedef TaskStateType* TaskStateRefType;

typedef enum
{
   TASK_1MS = 0,
   TASK_4MS,
   TASK_8MS,
   TASK_16MS,
   TASK_32MS,
   TASK_64MS,
   MAX_NUM_TASK
}TaskID;
```

```
typedef enum
{
   PRIORITY_0 = 0,
   PRIORITY_1,
   PRIORITY_2,
   PRIORITY_3,
   PRIORITY_4,
   PRIORITY_5,
   MAX_PRIORITY
}TaskPriority;

typedef enum
{
   E_OK = 0,
   E_OS_ACCESS,
   E_OS_CALLEVEL,
   E_OS_ID,
   E_OS_LIMIT,
   E_OS_NOFUNC,
   E_OS_RESOURCE,
   E_OS_STATE,
   E_OS_VALUE
}Status_Type;

typedef enum
{
   SUSPENDED = 0,
   READY,
   RUNNING
}TaskStates;

typedef struct
{
  u16 StartAddress;
  u16 EndAddress;
}StackInformation;

typedef struct
{
  u16 Relative;
  u16 Absolute;
}TaskDeadline;
```

```
typedef struct
{
  TaskID              Task_ID;
  TaskPriority        Task_Priority;
  TaskStates          Task_State;
  StackInformation    Stack_Information;
  TaskDeadline        Task_Deadline;
}Task_Control_Block;

typedef struct
{
  TaskID              Task_ID;
  TaskPriority        Task_Priority;
  TaskMaskType        Mask;
  TaskOffsetType      Offset;
  TaskStackSize       Size;
  TaskRelDeadline     RelDeadline;
  void(*TaskCallback)(void);
}Task_Descriptor;
```

Figure 6

Test Case	**ID**	**Status**
Additional Information	**2.5**	**Done**
Requirements covered		
2.5, 2.6, 2.12		
Test Procedure		
Define/declare Dispatcher in SchM.c/SchM.h files respectively & provide the Background task in SchM.c file which must call Dispatcher service.		
Expected Results		
Dispatcher declaration is inside of SchM.c & SchM.h files. Additionally Background task included only in SchM.c		
Actual Results		**Test Results**
Dispatcher is defined in SchM.c & SchM.h and Background task is included in SchM.c only.		PASS
Comments		
See Figure 7 & Figure 8		

SchM.c

```
/*****************************************************************
*   Function: Dispatcher
*
*   Description: Only run in Background task
*
*   Caveats: Non Reentrant
*****************************************************************
void Dispatcher(void)
{
  u16 IndexPriority = MAX_PRIORITY;
  u16 NoTaskExecuted = TRUE;
  do
  {
     IndexPriority--;
     if(DispacherArray[IndexPriority][0U] != 0xFFFF)
     {
         TaskExecuted_ID = DispacherArray[IndexPriority][0U];
         TaskControlBlock[TaskExecuted_ID].Task_State = RUNNING;

         TaskConfigInitial->ptr_Task[TaskExecuted_ID].TaskCallback();
         NoTaskExecuted = FALSE;
     }
  }
  while((IndexPriority != 0U)&&(NoTaskExecuted));
}
/*****************************************************************
*    Function: SchM_Background
*
*    Description: BackGround Task
*
*    Caveats: Non Reentrant
*****************************************************************
void SchM_Background(void)
{

    if(SchM_SchedulerEnabled)
    {
      if(SchM_OSTickEnabled)
      {
       PORTB_PB3= 1;

       Dispatcher();
       /*Wait for the next OS Tick to enable it*/
       SchM_OSTickEnabled = SCHM_OSTICK_DISABLED;
       PORTB_PB3= 0;
      }
      else
      {
         /*For future purposes*/
      }
    }
    else
    {
      /*Scheduler is not enabled*/
    }
}
```

Figure 7

SchM.h

```
/*****************************************************************************
*   Global Function Prototypes
*****************************************************************************/
/* vfnPIT0_Callback, function to be called upon timeout of PIT channel 0 */
extern void SchM_OsTick(void);
extern void SchM_Init(const TaskConfigType *SchM_Config);
extern void SchM_DeInit(void);
extern void SchM_Start(void);
extern void SchM_Background(void);
extern void Dispatcher(void);


#endif /*__SCHM_H*/
```

Figure 8

Test Case	ID	Status
Scheduler Services	2.6	Done
Requirements covered		
2.11		
Test Procedure		
Execute scheduler algorithm and ActivateTask in SchM_OsTick		
Expected Results		
Scheduler algorithm and ActivateTask have to be performed in SchM_OsTick		
Actual Results		**Test Results**
Scheduler algorithm and ActivateTask are performed in SchM_OsTick		PASS
Comments		
See Figure 9		

SchM.c

```
/*****************************************************************************************************
*   Function: SchM_OsTick
*
*   Description: Callback handled once an interrupt is generated by Channel 0.
*
*   Caveats: Non Reentrant
*****************************************************************************************************/
void SchM_OsTick(void)
{
  u8 Task_Index = 0;
  Status_Type StatusErrorResult = E_OK;

  if(!SchM_OSTickEnabled)
  {
    SchM_OSTickEnabled = SCHM_OSTICK_ENABLED;
    for(Task_Index=0;Task_Index < TaskConfigInitial[0U].TaskNumberConfig;Task_Index++)
    {
       if((SchM_OSTickCounter & TaskConfigInitial->ptr_Task[Task_Index].Mask) ==
           TaskConfigInitial->ptr_Task[Task_Index].Offset)
       {
         StatusErrorResult = ActivateTask(((TaskType)TaskConfigInitial->ptr_Task[Task_Index].Task_ID));
       }
    }
  }
  else
  {
    /* ERROR: Ostick flag was not disabled  */
  }

  SchM_OSTickCounter ++;
  PORTB_PB2= ~PORTB_PB2;
}
```

Figure 9

Test Case	ID	Status
Background	2.7	Done
Requirements covered		
2.13		
Test Procedure		
Call SchM_Start inside of main, then call SchM_Background.		
Expected Results		
SchM_Start must be called from main, after that SchM_Background should be called and must never ends		
Actual Results		**Test Results**
SchM_Background is called after SchM_Start inside of main and never ends		PASS
Comments		
See Figure 13 & Figure 14		

Main.c

```
/*****************************************************************************
*   Function: main()
*
*   Description: what I can say about it, it just the tipical main
*
*   Caveats:
*****************************************************************************/
void main(void)
{
  /*Disable interrupts to start the MCU configuration*/
  DisableInterrupts;
  /* Intilialize the interupt vector base address for default location */
  Init_Interrupt_Vector();

  Mcu_Init();
  DDRA =0x0F;
  DDRB =0x0F;

  EnableInterrupts;

  SchM_Init(TaskConfigInitial);
  /* Loop Forever and ever */
  SchM_Start();
}
```

Figure 13

SchM.c

```
/*****************************************************************************************
*   Function: SchM_Start
*
*   Description: Starts the scheduler
*
*   Caveats:
*****************************************************************************************/
void SchM_Start(void)
{
    u16 index_a;
    u16 index_b;
    u16 index_ControlBlock;
    /*Start the timer for the OS tick*/
    Gpt_StartTimer(CHANNEL_0, 500u);
    /*Enable notification*/
    Gpt_EnableNotification(CHANNEL_0);
    /* Initialize Dispacher Array Vector */
    for(index_a=0U;index_a < MAX_PRIORITY;index_a++)
    {
      for(index_b=0U;index_b < CNF_MAXTASKQUEUE;index_b++)
      {
         DispacherArray[index_a][index_b] = 0xFFFF;
      }
    }

    TaskControlBlock = ((Task_Control_Block *__far)Mem_Alloc((MAX_NUM_TASK*sizeof(Task_Control_Block))));
    for(index_ControlBlock=0U;index_ControlBlock < MAX_NUM_TASK;index_ControlBlock++)
    {
      TaskControlBlock[index_ControlBlock].Task_ID                      = TaskConfigInitial->ptr_Task[index_ControlBlock].Task_ID;
      TaskControlBlock[index_ControlBlock].Task_Priority                = TaskConfigInitial->ptr_Task[index_ControlBlock].Task_Priority;
      TaskControlBlock[index_ControlBlock].Task_State                   = SUSPENDED;
      TaskControlBlock[index_ControlBlock].Stack_Information.StartAddress = 0U;
      TaskControlBlock[index_ControlBlock].Stack_Information.EndAddress   = 0U;
      TaskControlBlock[index_ControlBlock].Task_Deadline.Relative        = 0U;
      TaskControlBlock[index_ControlBlock].Task_Deadline.Absolute        = 0U;
    }

    while(1)
    {
      SchM_Background();
      _FEED_COP();
    }

}
```

Figure 14

Test Case	ID	Status
Priority	**2.8**	**Done**
Requirements covered		
2.14, 2.15		
Test Procedure		
Assign the lowest priority value to the lowest priority task, assign the same priority to a few tasks		
Expected Results		
Task execution can be performed according its priority assigned even if they have the same priority		
Actual Results		**Test Results**
Tasks are executed according its priority no matter whether or not they have the same priority		PASS
Comments		
Task priorities are configured according requirement 2.14 initially, but tasks sharing the same priority according requirement 2.15, are fully supported. See Figure 15		

Os_TaskCfg.c

```
/*****************************************************************
*   Global Variable Definitions
*****************************************************************/
const Task_Descriptor TaskInitial[]=
{
  {TASK_1MS, PRIORITY_5, MASK_1MS, OFFSET_00,0,0,Task_1ms},
  {TASK_4MS, PRIORITY_4, MASK_4MS, OFFSET_01,0,0,Task_4ms},
  {TASK_8MS, PRIORITY_3, MASK_8MS, OFFSET_03,0,0,Task_8ms},
  {TASK_16MS,PRIORITY_2,MASK_16MS, OFFSET_05,0,0,Task_16ms},
  {TASK_32MS,PRIORITY_1,MASK_32MS, OFFSET_07,0,0,Task_32ms},
  {TASK_64MS,PRIORITY_0,MASK_64MS, OFFSET_11,0,0,Task_64ms}
};
```

Figure 15

7. Integration Test Cases

Test Case	ID	Status
Running project	**2.9**	**Done**
Requirements covered		
2.16, 2.17		
Test Procedure		
Set a pin level to high when tasks start its execution. Set a pin level low when tasks end its execution.		
Expected Results		
Pin out level must be set high every time tasks start its execution, and must be set low every time the tasks ends its execution.		
Actual Results		**Test Results**
See the next figures which contains different configurations for the task properties.		PASS
Comments		
Each result is composed with two images, the first shows the configuration of the task and the second shows the result and order in which the task were executed.		

Next figures show each result.

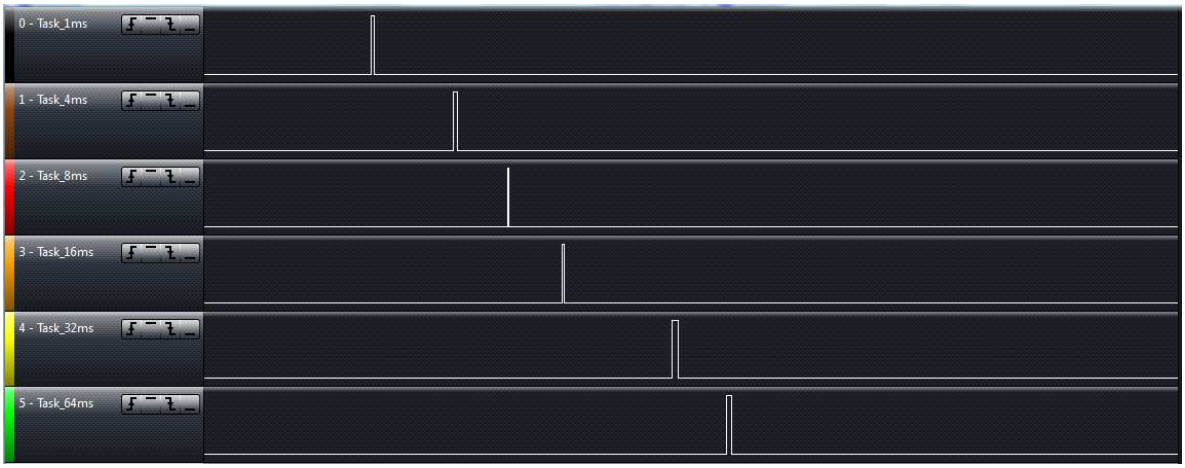
```
const Task_Descriptor TaskInitial[]=
{
  {TASK_1MS, PRIORITY_0, MASK_32MS, OFFSET_00,0,0,Task_1ms},
  {TASK_4MS, PRIORITY_1, MASK_32MS, OFFSET_00,0,0,Task_4ms},
  {TASK_8MS, PRIORITY_2, MASK_32MS, OFFSET_00,0,0,Task_8ms},
  {TASK_16MS,PRIORITY_3,MASK_32MS, OFFSET_00,0,0,Task_16ms},
  {TASK_32MS,PRIORITY_4,MASK_32MS, OFFSET_00,0,0,Task_32ms},
  {TASK_64MS,PRIORITY_5,MASK_32MS, OFFSET_00,0,0,Task_64ms}
};
```

Code Test 1

Result Test 1

```
const Task_Descriptor TaskInitial[]=
{
  {TASK_1MS, PRIORITY_0, MASK_32MS, OFFSET_00,0,0,Task_1ms},
  {TASK_4MS, PRIORITY_1, MASK_32MS, OFFSET_03,0,0,Task_4ms},
  {TASK_8MS, PRIORITY_2, MASK_32MS, OFFSET_05,0,0,Task_8ms},
  {TASK_16MS,PRIORITY_3,MASK_32MS, OFFSET_07,0,0,Task_16ms},
  {TASK_32MS,PRIORITY_4,MASK_32MS, OFFSET_11,0,0,Task_32ms},
  {TASK_64MS,PRIORITY_5,MASK_32MS, OFFSET_13,0,0,Task_64ms}
};
```

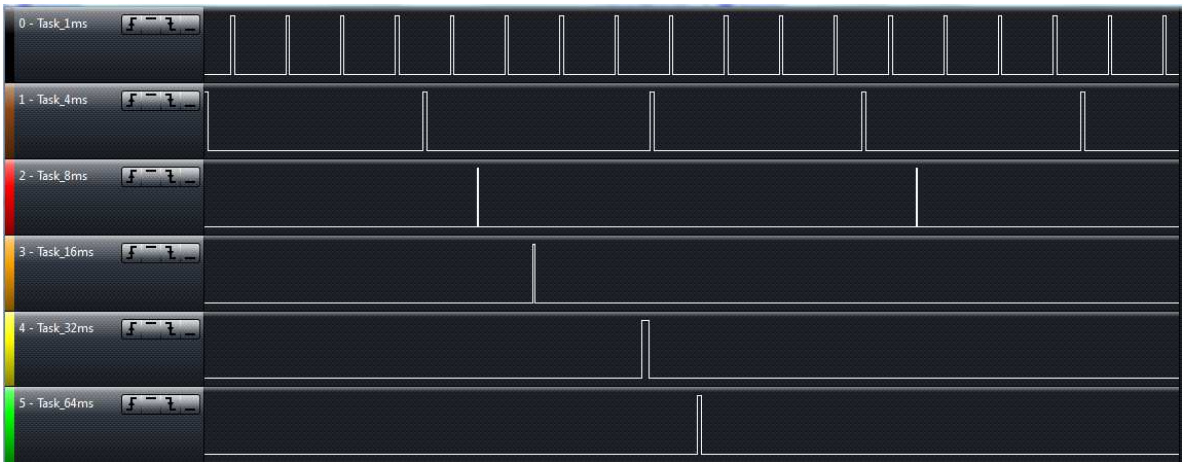
Code Test 2



Result Test 2

```
const Task_Descriptor TaskInitial[]=
{
  {TASK_1MS, PRIORITY_0, MASK_1MS, OFFSET_00,0,0,Task_1ms},
  {TASK_4MS, PRIORITY_1, MASK_4MS, OFFSET_03,0,0,Task_4ms},
  {TASK_8MS, PRIORITY_2, MASK_8MS, OFFSET_05,0,0,Task_8ms},
  {TASK_16MS,PRIORITY_3,MASK_16MS, OFFSET_07,0,0,Task_16ms},
  {TASK_32MS,PRIORITY_4,MASK_32MS, OFFSET_11,0,0,Task_32ms},
  {TASK_64MS,PRIORITY_5,MASK_64MS, OFFSET_13,0,0,Task_64ms}
};
```

Code Test 3



Result Test 3

```
const Task_Descriptor TaskInitial[]=
{
  {TASK_1MS, PRIORITY_5, MASK_1MS, OFFSET_00,0,0,Task_1ms},
  {TASK_4MS, PRIORITY_1, MASK_4MS, OFFSET_03,0,0,Task_4ms},
  {TASK_8MS, PRIORITY_2, MASK_8MS, OFFSET_05,0,0,Task_8ms},
  {TASK_16MS,PRIORITY_3,MASK_16MS, OFFSET_07,0,0,Task_16ms},
  {TASK_32MS,PRIORITY_2,MASK_32MS, OFFSET_11,0,0,Task_32ms},
  {TASK_64MS,PRIORITY_3,MASK_64MS, OFFSET_13,0,0,Task_64ms}
};
```

Code Test 4

Result Test 4

Test Case	**ID**	**Status**
Running project	**2.10**	**Done**
Requirements covered		
2.18		
Test Procedure		
Define the basic states Suspended, Ready & Running for all Tasks States.		
Expected Results		
Tasks states will be transitioning the basic states Suspended, Ready & Running.		
Actual Results		**Test Results**
See Figure 16 & Figure 17.		PASS
Comments		
Tasks will be set in suspended state every time they're called to terminate in TerminateTask API. Tasks in ready state will be set every time they're put into the priority buffer by ActivateTask API. Tasks in running state are set every time they're called inside the Dispatcher API.		

```
/*****************************************************************************************************
*   Function: TerminateTask(void)
*
*   Description: This function terminates the task that invoked the service,
*   transferring it from the RUNNING to the SUSPENDED state, and move the task out
*   of the priority buffer queue
*
*   Caveats:
*****************************************************************************************************/
Status_Type TerminateTask (void)
{
  Status_Type Status = E_OK;
  u16 IndexQueue=0;

  TaskControlBlock[TaskExecuted_ID].Task_State = SUSPENDED;
  for(IndexQueue;IndexQueue < CNF_MAXTASKQUEUE;IndexQueue++)
  {
     DispacherArray[TaskConfigInitial->ptr_Task[TaskExecuted_ID].Task_Priority][IndexQueue] =
     DispacherArray[TaskConfigInitial->ptr_Task[TaskExecuted_ID].Task_Priority][IndexQueue+1];
  }
  DispacherArray[TaskConfigInitial->ptr_Task[TaskExecuted_ID].Task_Priority][IndexQueue-1] = 0xFFFF;
  TaskExecuted_ID = 0xFFFF;
  return Status;
}
```

Figure 16

```
/*****************************************************************************************************
*   Function: ActivateTask(TaskType taskID)
*
*   Description: This API service moves the task identified by the input
*   taskID from the SUSPENDED state to the READY state if
*   no error occurred, and put the task into the priority buffer
*   queue tail
*
*   Caveats:
*****************************************************************************************************/
Status_Type ActivateTask(TaskType taskID)
{
  u16 index = 0;
  Status_Type Status = E_OK;
  u8 Processing_Done = FALSE;
  for(index = 0; index < CNF_MAXTASKQUEUE; index++)
  {
    if(Processing_Done == FALSE)
    {
      if(DispacherArray[TaskConfigInitial->ptr_Task[taskID].Task_Priority][index] == 0xFFFF)
      {
         DispacherArray[TaskConfigInitial->ptr_Task[taskID].Task_Priority][index] = taskID;

         Processing_Done = TRUE;
      }
    }
  }
  TaskControlBlock[taskID].Task_State = READY;
  if(Processing_Done == FALSE)
  {
     Status = E_OS_LIMIT;
  }
  return Status;
}
/*****************************************************************************************************
*   Function: Dispatcher
*
*   Description: Only run in Background task
*
*   Caveats: Non Reentrant
*****************************************************************************************************/
void Dispatcher(void)
{
  u16 IndexPriority = MAX_PRIORITY;
  u16 NoTaskExecuted = TRUE;
  do
  {
     IndexPriority--;
     if(DispacherArray[IndexPriority][0U] != 0xFFFF)
     {
         TaskExecuted_ID = DispacherArray[IndexPriority][0U];
         TaskControlBlock[TaskExecuted_ID].Task_State = RUNNING;

         TaskConfigInitial->ptr_Task[TaskExecuted_ID].TaskCallback();
         NoTaskExecuted = FALSE;
     }
  }
  while((IndexPriority != 0U)&&(NoTaskExecuted));
}
```

Figure 17

Test Case	ID	Status
Running project	**2.11**	**Done**
Requirements covered		
2.19		
Test Procedure		
Place ready tasks IDs into its correspondent priority buffer.		
Expected Results		
Tasks IDs are placed into its correspondent priority buffer every time they turn ready.		
Actual Results		**Test Results**
Tasks IDs are placed into its correspondent priority buffer every time they turn ready. As figure 18 and 19.		PASS
Comments		

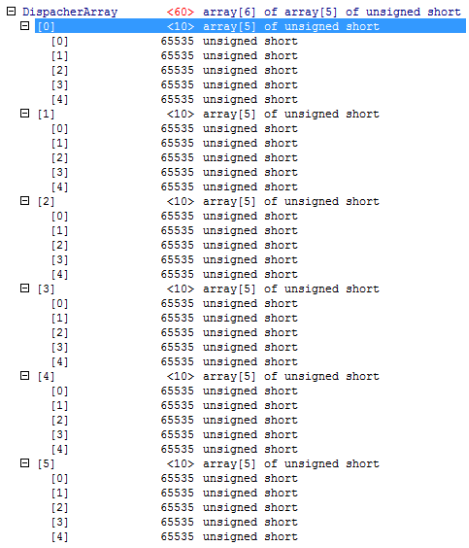


Figure 18

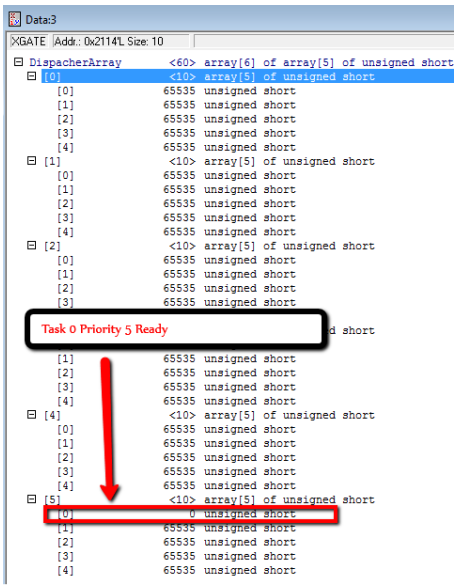


Figure 19

Test Case	**ID**	**Status**
Running project	**2.12**	**Done**
Requirements covered		
2.20		
Test Procedure		
Move out tasks which transition from ready to suspended from its correspondent priority buffer		
Expected Results		
Tasks to be suspended are moved out from its last priority buffer.		
Actual Results		**Test Results**
Tasks to be suspended are moved out from its last priority buffer. As figure 20 and 21		PASS
Comments		

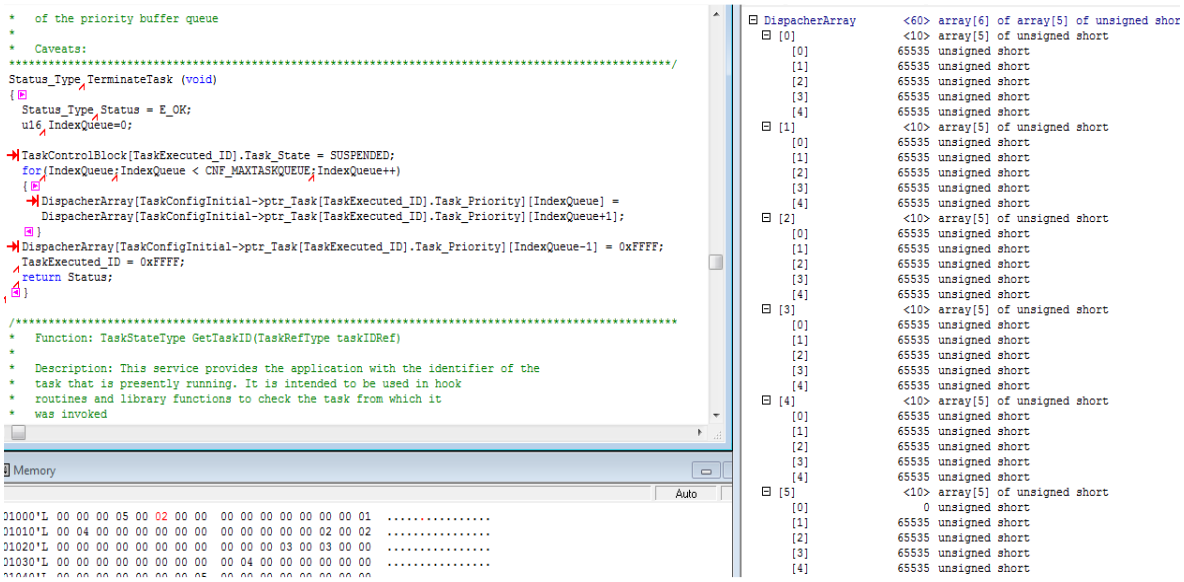


Figure 20

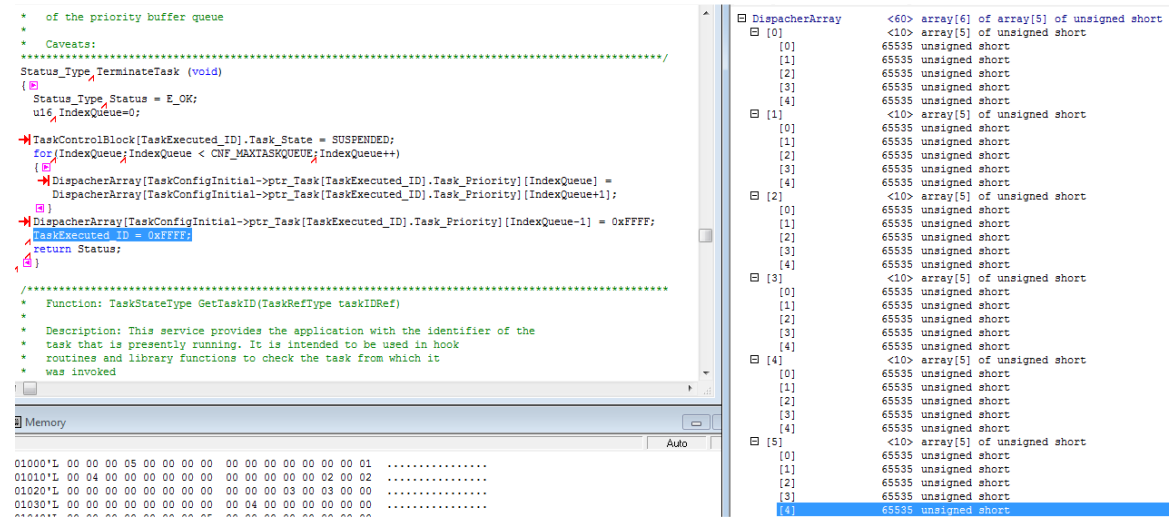


Figure 21

Test Case	ID	Status
Internal requirements	**2.13**	**Done**
Requirements covered		
2.21		
Test Procedure		
Include memory allocation driver from previous projects.		

Expected Results	
Memory allocation will be available for usage in Task manager project.	
Actual Results	**Test Results**
Memory allocation will be available for usage in Task manager project. As figure 22 and 23	PASS
Comments	

```
void Mem_Init(void)
{

    u16 i = 0;


    /* Head initial address pointer */
    mcal_mem_MemHandler.memStart = &mcal_mem_InitMem;
    /* Define Heap Memory Size */
    mcal_mem_TotalMemSizeBytes = (CNF_NO_OF_RAM_PAGES * CNF_PAGE_SIZE) - 1;
    /* Define Heap End Address */
    mcal_mem_MemHandler.memEnd = mcal_mem_TotalMemSizeBytes + mcal_mem_MemHandler.memStart;
    /* Current address points to heap's start address */
    mcal_mem_MemHandler.currAddr = mcal_mem_MemHandler.memStart;


    /* Set initial values of Heap memory */
    for(i=0;i < mcal_mem_TotalMemSizeBytes;i++)
    {
        *mcal_mem_MemHandler.currAddr = 0x00;
        mcal_mem_MemHandler.currAddr++;
    }

    /* Memory is empty */
    mcal_mem_MemHandler.memFull = FALSE;
    /* Current address points to Heap's start address */
    mcal_mem_MemHandler.currAddr = mcal_mem_MemHandler.memStart;

}
```

Figure 22

Figure 23

Test Case	ID	Status
Internal requirements	**2.14**	**Done**
Requirements covered		
2.22		

Test Procedure	
Reserve memory for control block structure using memory allocation.	
Expected Results	
Reserved memory section is initialized to 0 and its size is the control block structure size.	
Actual Results	**Test Results**
Reserved memory section is initialized to 0 and its size is the control block structure size. As figure 24 and 25	PASS
Comments	

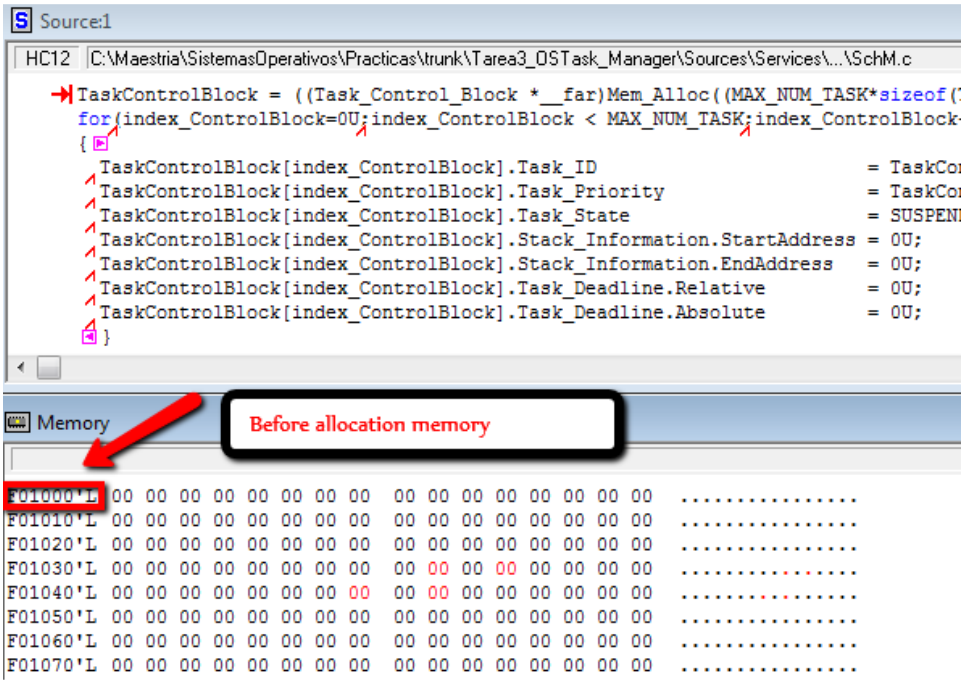


Figure 24

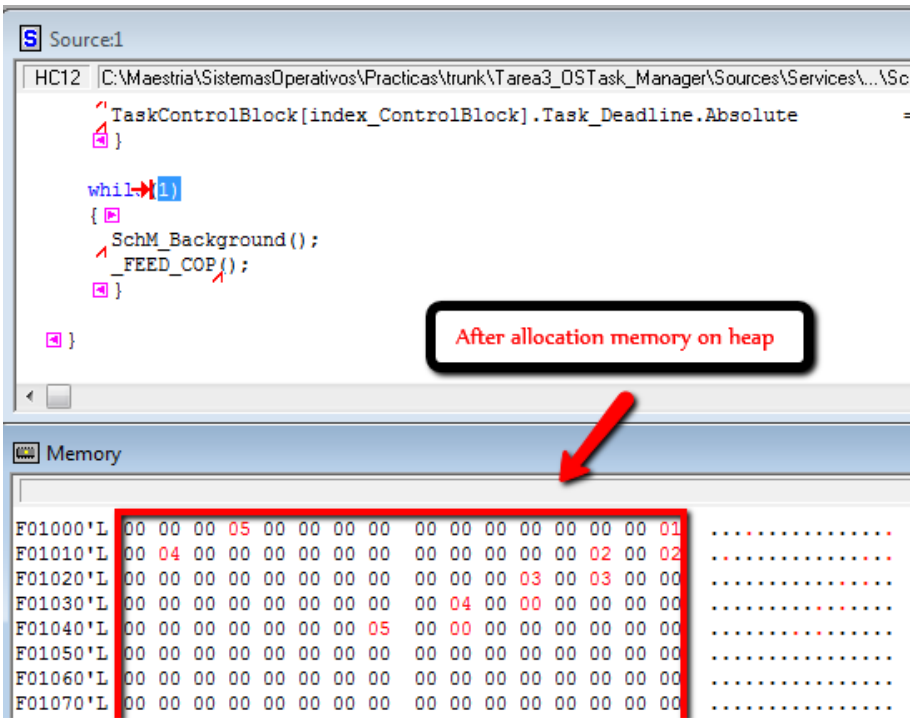
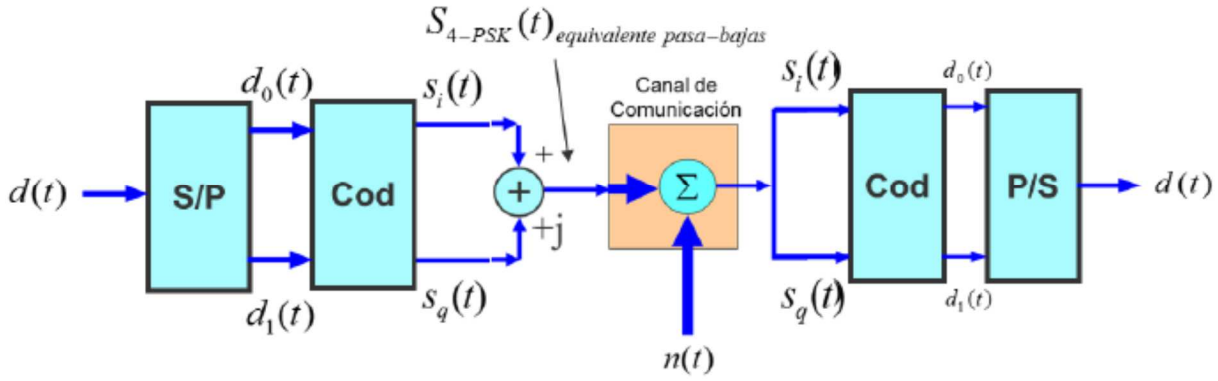

Source:1
HC12
C:\Maestria\SistemasOperativos\Practicas\trunk\Tarea3_OSTask_Manager\Sources\Services\...\Sc
TaskControlBlock[index_ControlBlock].Task_Deadline.Absolute
}
while(1)
{
SchM_Background();
_FEED_COP();
}
}
After allocation memory on heap
Memory
F01000'L 00 00 00 05 00 00 00 00 00 00 00 00 00 00 00 01 ....
F01010'L 00 04 00 00 00 00 00 00 00 00 00 00 00 02 00 02 ....
F01020'L 00 00 00 00 00 00 00 00 00 00 00 03 00 03 00 00 ....
F01030'L 00 00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 ....
F01040'L 00 00 00 00 00 00 00 05 00 00 00 00 00 00 00 00 ....
F01050'L 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
F01060'L 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
F01070'L 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

Figure 25

B. REPORTE DE TRANSMISIÓN DE DATOS EN EQUIVALENTE PASA-BAJAS CON MODULACIÓN 4-PSK

Transmisión de datos en equivalente pasa-bajas con modulación 4-PSK

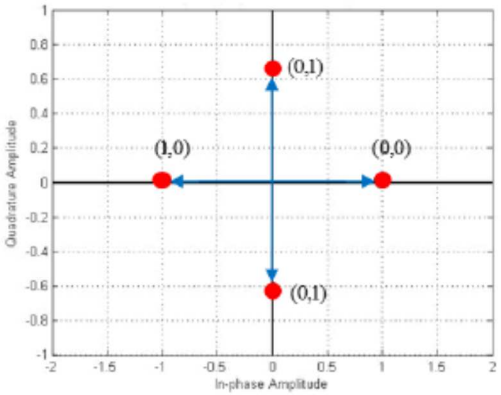
Realizar en la plataforma de simulación MATLAB un proceso de modulación digital pasa-baja 4-PSK representado por su equivalente pasa-bajas. El sistema debe considerar la adición de ruido en la transmisión, debe ser capaz de medir la taza de error y mostrar el diagrama de estados tanto en el transmisor como en el receptor. En la siguiente figura se ilustra el proceso.



Las especificaciones son las siguientes:
Generar una señal digital d(t) compuesta de 1024 valores digitales binarios(“1” y “0”) producidos aleatoriamente con una distribución normal (gaussiana) y con una velocidad de datos de 1000 bps. Emplear la función randm() para obtener valores aleatorios que luego sean convertidos a 1’s y 0’s.

```
% Nd = número de valores digitales binarios
Nd = 1024;
dt = round(rand(1, Nd));
```

Formar pares secuenciales de bits que produzcan las fases correspondientes. Los valores de las fases pueden definirse libremente. La siguiente figura ilustra el diagrama de estados de la portadora para 4 posibles fases.



Para este caso se definieron las mismas fases mostradas en el diagrama de estados anterior quedando de la siguiente manera:

```
%                              X    Y
%  | d1(t),d0(t) | Phase | Si | Sq |                Sq
%  |-------------------------------|              1 + (0,1)
%  |    (0,0)    |  0    | 1  | 0  |                |
%  |-------------------------------|                |
%  |    (0,1)    | pi/2  | 0  | 1  |         (1,0)  |   (0,0)
%  |-------------------------------|       ----+-----+-----+--- Si
%  |    (1,0)    |  pi   | -1 | 0  |          -1    |0    1
%  |-------------------------------|                |
%  |    (1,1)    | 3pi/2 |  0 | -1 |                |
%  |-------------------------------|             -1 + (1,1)
```

Los pares secuenciales “Si(fase)” y “Sq (cuadratura)” se definieron de acuerdo a la tabla anterior:
Generar a través del codificador los valores correspondientes de las señales Si(t) y Sq(t) para generar las fases previamente definidas.

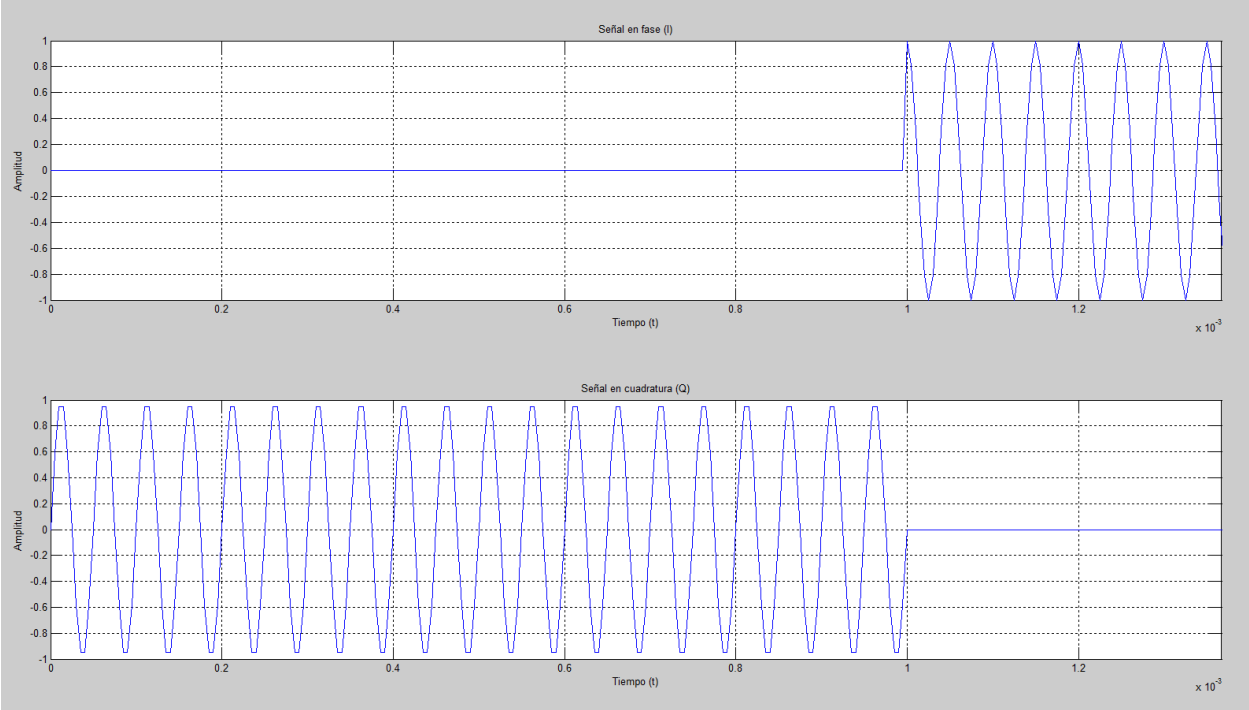
```
for i=1 : Ct
          % d0(t)                    d1(t)
    if((dt(indx1(i)) == 0) && (dt(indx2(i))==0))
    Si(i) = As*cos(sym(0));
    Sq(i) = As*sin(sym(0));
    end
    if((dt(indx1(i)) == 0) && (dt(indx2(i))==1))
    Si(i) = As*cos(sym(pi/2));
    Sq(i) = As*sin(sym(pi/2));
    end
    if((dt(indx1(i)) == 1) && (dt(indx2(i))==0))
    Si(i) = As*cos(sym(pi));
    Sq(i) = As*sin(sym(pi));
    end
    if((dt(indx1(i)) == 1) && (dt(indx2(i))==1))
    Si(i) = As*cos(sym((3*pi)/2));
    Sq(i) = As*sin(sym((3*pi)/2));
    end

end
```

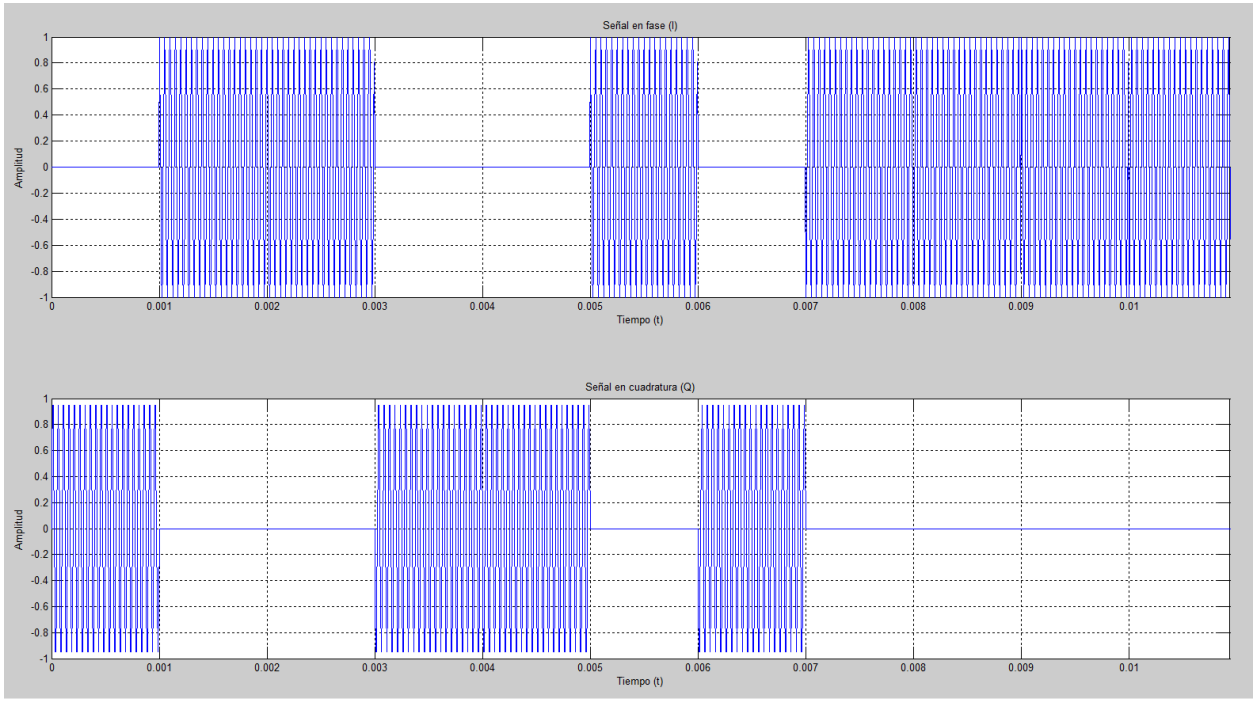
Transmitir al canal de comunicación la señal equivalente pasa-baja formada por la adición de Si(t) y Sq(t).
Configuración de la señal:

```
% Nd = número de valores digitales binarios
Nd = 1024;
dt = round(rand(1, Nd));

% velocidad de datos de 1000bps = 1KHz
Vd = 1000; % 1000 Hz
% Tb = Tiempo de bit
Tb = 1/Vd; % 1 ms
% Definición de portadora a 10KHz
% Frecuencia de portadora Fp
```

```
Fp = 20*Vd;
Tp= 1/Fp;
Fs = 10*Fp; %Frecuencia de muestreo para la señal portadora

Ts = 1/Fs; %
% Nm = número de muestras por bit
Nm = Tb/Ts;
% Tiempo total de 1024 bits
t = 0:(Tb/Nm):((Nd*Tp*Nm)-((Tp/Nm)*Nm));
w = 2*pi*Fp;
```

Gráficas de las señales generadas Si(t) y Sq(t) mostradas de manera independiente.



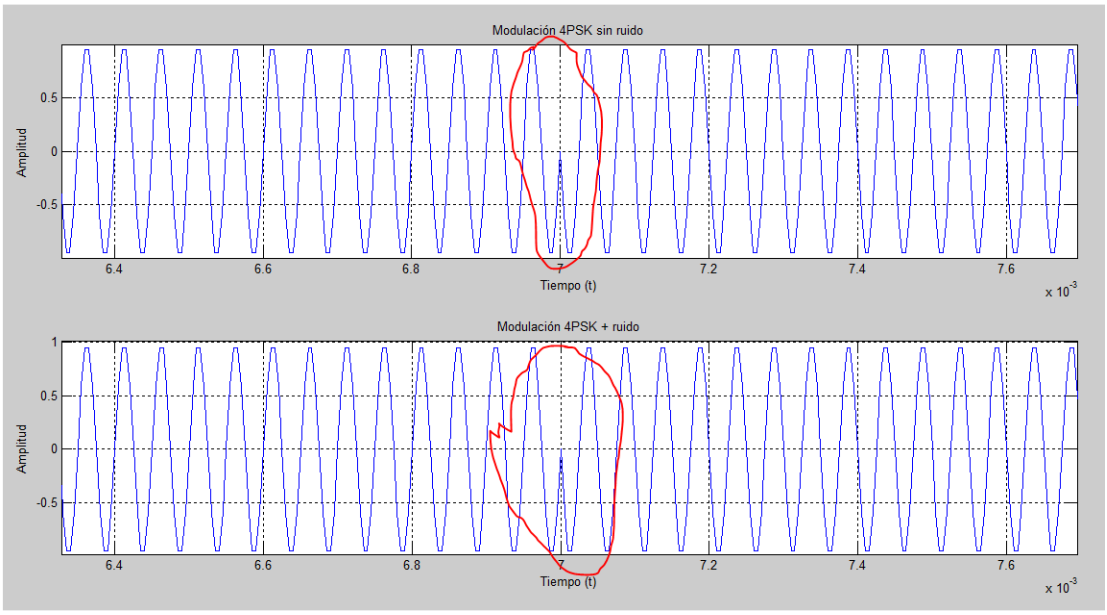


Señal con ruido:

```
TxSt = SiXt.*cos(w.*ft) - SqXt.*sin(w.*ft);
```

Agregar a la señal transmitida anterior el ruido n(t), el cual posee una distribución gaussiana normalizada randn().

Comparativo de la modulación sin ruido vs con ruido, con un nivel de ruido de amplitud '0'



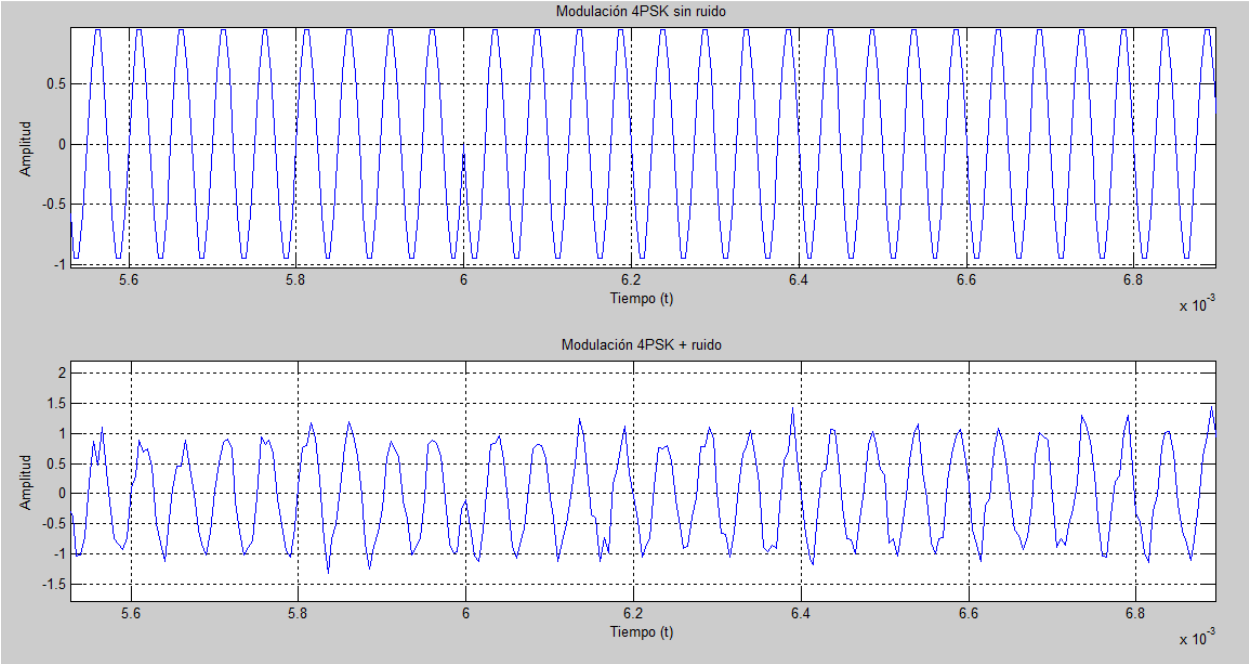
El círculo rojo en la señal muestra el cambio de fase.

Variar la potencia del ruido de acuerdo a lo que se indique posteriormente.

```
% El ruido debe afectar de manera independiente a cada parte de la señal
% equivalente pasa-bajas, es decir, las señales de ruido que se le añade
```

```
% a Si(t) y Sq(t) son independientes.

Si_n = SiXt + ntSi;
Sq_n = SqXt + ntSq;
```

Comparativo de la modulación sin ruido vs con ruido, con un nivel de ruido de amplitud '0.13422'



El receptor debe ser capaz de producir la información digital binaria a partir del equivalente pasa-bajas de la señal con ruido que recibe.
Antes de producir la información digital, primero filtramos la señal recibida con un filtro pasa-bajas para eliminar el ruido añadido en el canal de transmisión.

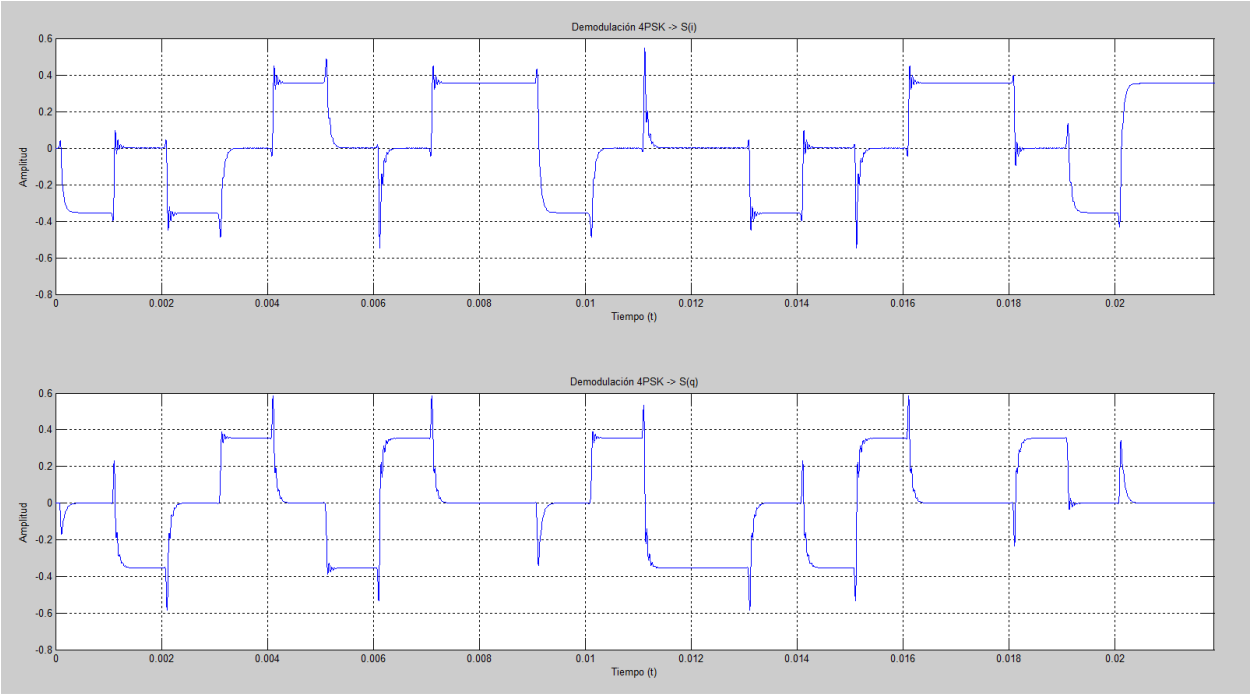
```
Fc = Fp;
[b, a] = butter(10, Fc/(Fs/2));
% Quitamos el ruido que se agrega en el canal
DataRxNL = filter(b, a, TxSt);
```

Aplicamos las propiedades ortogonales de la función senoidal y cosenoidal para obtener Si(t) y Sq(t) de la señal recibida y filtrada.

```
RxSq = DataRxNL.*cos(w.*ft);
RxSi = DataRxNL.*sin(w.*ft);
```

Filtramos las señales de Si y Sq para eliminar las componentes de las frecuencias no deseadas y así poder decodificar nuestras señales de interés.

```
FSiRx = filter(b, a, RxSi);
FSqRx = filter(b, a, RxSq);
```



Para codificar las señales recibidas se toman 20 muestras tomando como referencia el punto medio de cada bit considerando el tiempo de bit.

```
% Número de muestras a evaluar de la señal demodulada y filtrada.
% 10% del tamaño total del bit
Samp = Nm * 0.1;
% Punto medio del bit recibido a codificar
Pm = Nm/2;
% Vector de datos recibidos para Si & Sq en la recepción
RxSiData = zeros(1 , Ct);
RxSqData = zeros(1 , Ct);

for k = 1 : Ct

    tmpSIRx_Data = FSiRx((((k*Nm)-Pm) - (Samp/2)): (((k*Nm)-Pm) + (Samp/2)));
    tmpSQRx_Data = FSqRx((((k*Nm)-Pm) - (Samp/2)): (((k*Nm)-Pm) + (Samp/2)));
    tmpAvgSI = sum(tmpSIRx_Data)/Samp;
    tmpAvgSQ = sum(tmpSQRx_Data)/Samp;

    % decodificación del "0"
    if((tmpAvgSI < 0.1) && (tmpAvgSI > -0.1))
        RxSiData(k) = 0;
    else
        % Decodificación del "1"
        if((tmpAvgSI < 0.7) && (tmpAvgSI > 0.35))
            RxSiData(k) = 1;
        else
            % Decodificación del "-1"
            if((tmpAvgSI < -0.25) && (tmpAvgSI > -0.7))
                RxSiData(k) = -1;
            end
        end
```

```
    end

    % decodificación del "0"
    if((tmpAvgSQ < 0.1) && (tmpAvgSQ > -0.1))
        RxSqData(k) = 0;
    else
        % Decodificación del "1"
        if((tmpAvgSQ < 0.7) && (tmpAvgSQ > 0.35))
            RxSqData(k) = 1;
        else
            % Decodificación del "-1"
            if((tmpAvgSQ < -0.25) && (tmpAvgSQ > -0.7))
                RxSqData(k) = -1;
            end
        end
    end
end
```

Una vez que se determina si la señal recibida corresponde a un ‘0’, ‘1’, o ‘-1’ se comienzan a generar los pares de bits en los cuales se dividió la señal original de acuerdo a su fase.

```
if((RxSiData(k) == 1)&& (RxSqData(k)== 0))
    RxSiData(k) = 0;
    RxSqData(k) = 0;
end
if((RxSiData(k) == 0)&& (RxSqData(k)== 1))
    RxSiData(k) = 0;
    RxSqData(k) = 1;
end
if((RxSiData(k) == -1)&& (RxSqData(k)== 0))
    RxSiData(k) = 1;
    RxSqData(k) = 0;
end
if((RxSiData(k) == 0)&& (RxSqData(k)== -1))
    RxSiData(k) = 1;
    RxSqData(k) = 1;
end
```

Reordenar los pares de bits en una serie de 1024 bits de acuerdo a la longitud de la señal transmitida originalmente.

```
% Reordenamiento de los datos recibidos de paralelo a serie
for m = 1 : Ct
RxVector(indx1(m)) = RxSiData(m);
RxVector(indx2(m)) = RxSqData(m);

end
```

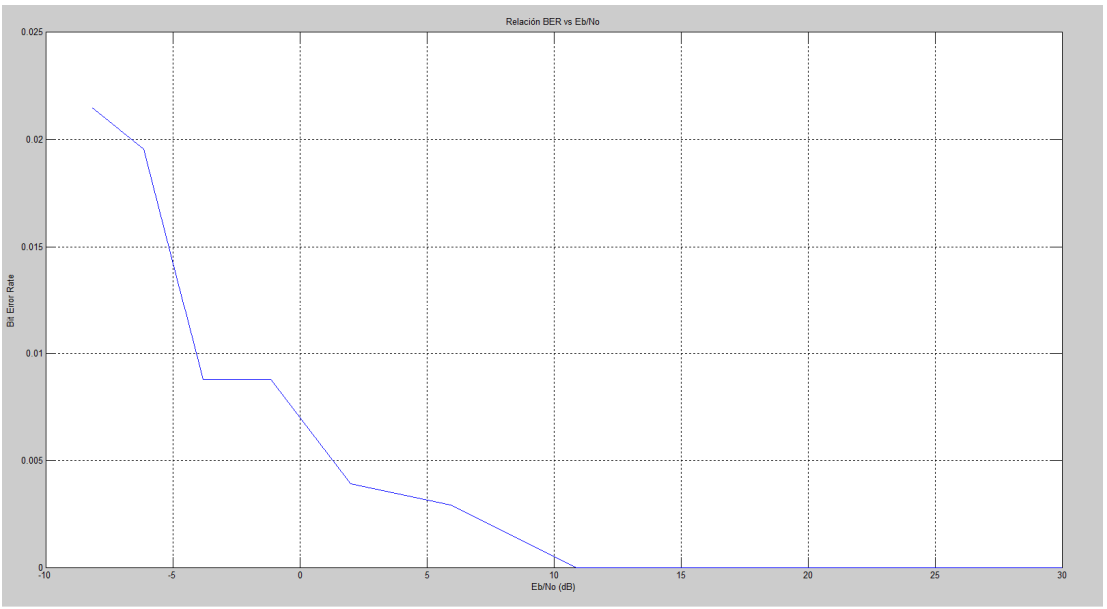
La taza de Error debe determinarse por medio del cociente entre el número de bits reconocidos erróneamente en el receptor en relación al número total de bits recibidos.

```
% Cálculo del número de bits recibidos erroneamente
TE = 0;
for l=1:Nd
    if(dt(l) ~= RxVector(l))
        TE = TE + 1;
    end
end
% cálculo del Bit Error Rate
```

```
% BER = EB/NB donde EB(TE): es el número de bis recibidos incorrectamente y
NB(Nd):
% es el número total de bits transmitidos
TE2(R) = TE;
BER(R) = TE/Nd;

% Para graficar BER vs Eb/No necesitamos obtener Eb y No.
% Eb = ((A^2)*T)/2 donde Eb = Energia del bit y No es ka densidad espectral
% de potencia del ruido
% No = 2*(var^2)
% Tb = velocidad de transmision de los datos T
v = ((var(ntSi) + var(ntSq))/2)^2;
NR(R) = (var(ntSi) + var(ntSq))/2;
No = 2*v;
A = 1;

Eb = ((A^2)*Tb)/2;
EbNo = Eb/No;

dB(R) = 10*log10(EbNo);

end
```

Realizar varias ejecuciones del sistema utilizando diferentes valores de la razón Eb/No expresada en dB: desde un valor de 0 hasta 30 dB. Para cada ejecución calcular la Taza de Error obtenida (BER) vs Eb/No expresada en dBs.

```
No. Errores ----  B.E.R.  ------ Ruido ----- Eb/No (dB)
     0       ---- 0.00000  ----  0.00000  ---- Inf
     0       ---- 0.00000  ----  0.00050  ---- 30
     0       ---- 0.00000  ----  0.00200  ---- 18
     0       ---- 0.00000  ----  0.00451  ---- 11
     1       ---- 0.00098  ----  0.00799  ----  6
     5       ---- 0.00488  ----  0.01248  ----  2
    16       ---- 0.01563  ----  0.01795  ---- -1
    17       ---- 0.01660  ----  0.02462  ---- -4
    22       ---- 0.02148  ----  0.03189  ---- -6
    25       ---- 0.02441  ----  0.04056  ---- -8
fx >>
```



Comparativo BER vs Eb/No

Graficar el diagrama de estados (fases) de la señal equivalente pasa-bajas generada en el transmisor y en la obtenida a la entrada del codificador para diferentes niveles de ruido (solo para algunos de ellos).

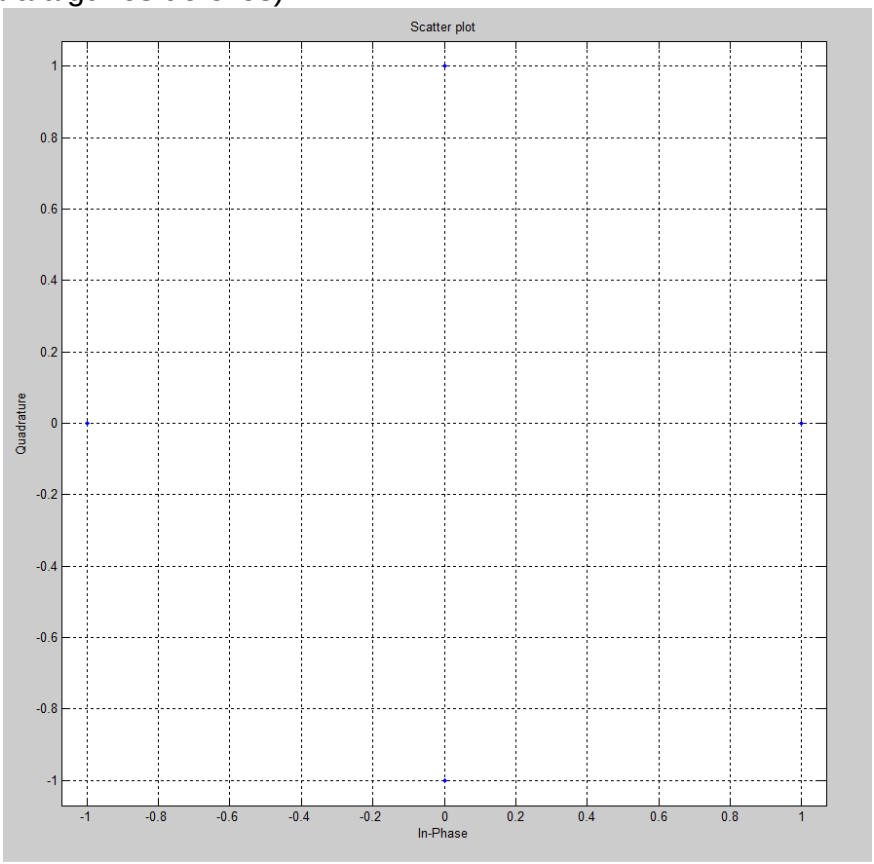


Diagrama de fases con nivel de ruido en ‘0’

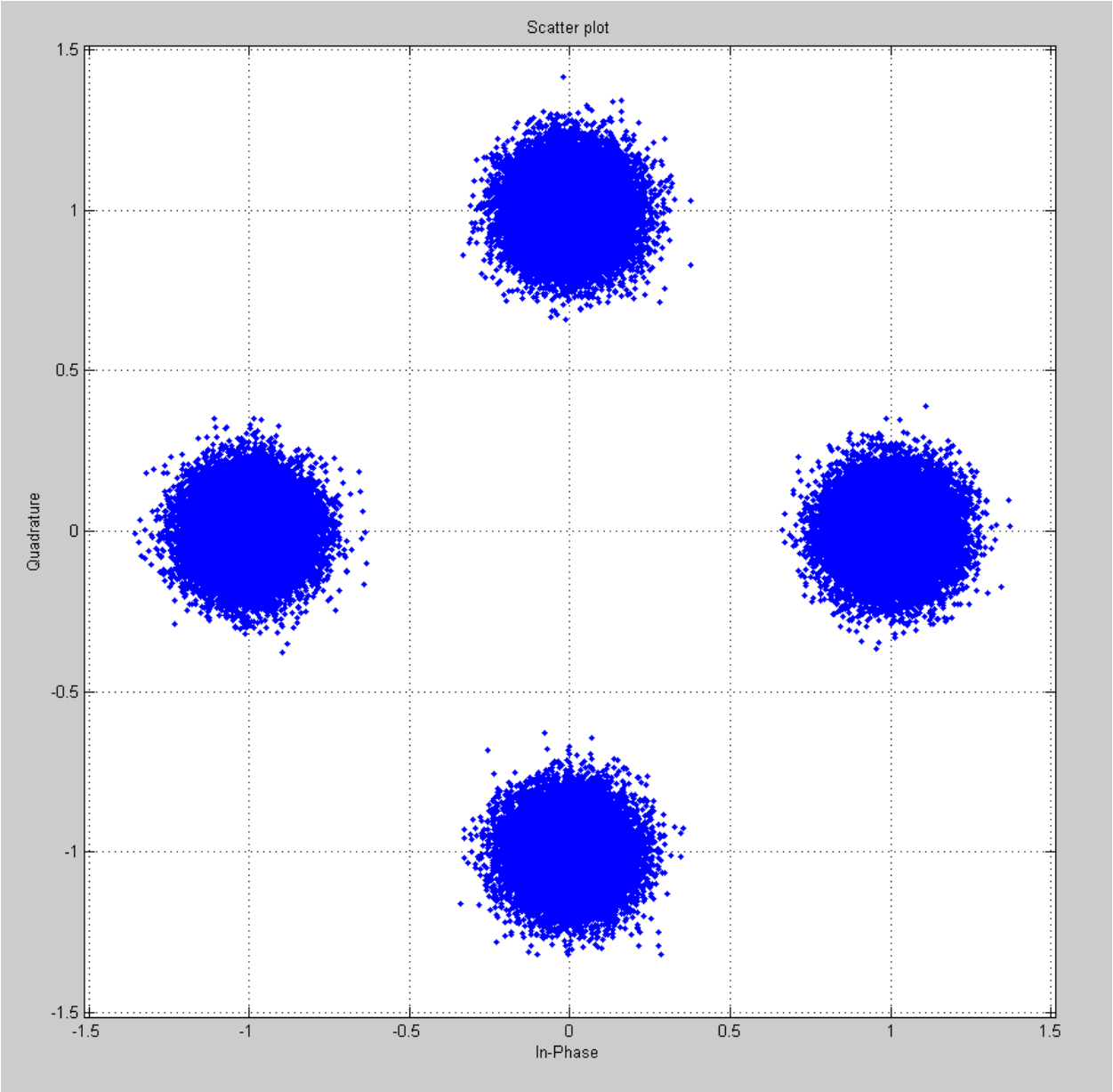


Diagrama de fases con un nivel de ruido al 50% de la amplitud de la señal

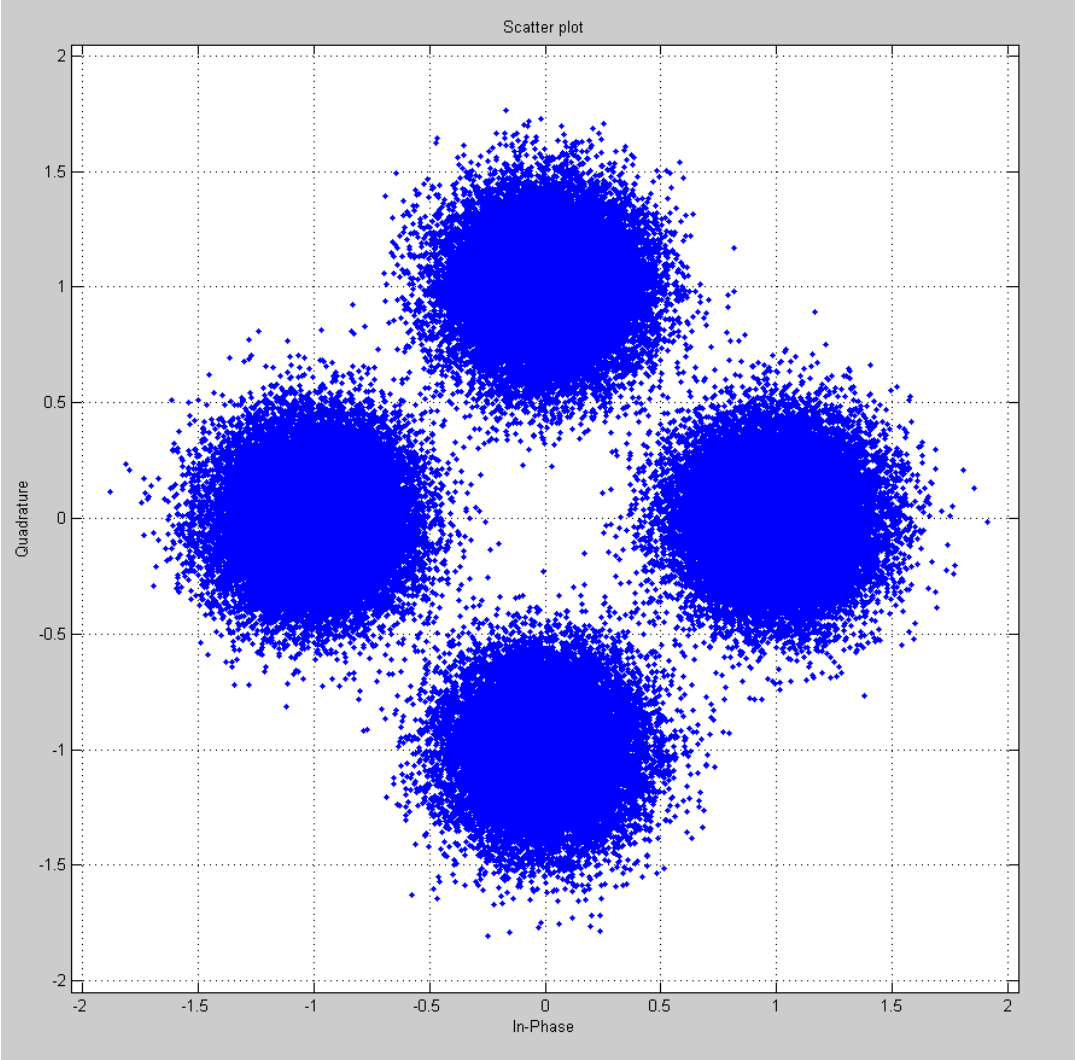


Diagrama de fases con un nivel de ruido al 100% de la amplitud de la señal

Código fuente:

```
close all %cierra imagenes
clear %limpia workspace
clc;
set(0, 'DefaultFigureCreateFcn', 'zoom xon');
% Realizar un proceso de modulación digital pasa-banda 4-PSK
representado
% por su equivalente pasa-bajas. El sistema debe considerar la
adición de
% ruido en la transmisión, debe de ser capaz de medir la taza de
error y
% mostrar el diagrama de estados tanto en el transmisor como en
el receptor
```

```
% 1 - Generar una señal digital d(t) compuesta de 1024 valores
digitales
% binarios (“1” y “0”) producidos aleatoriamente con una
distribución normal
% (gaussiana) y con una velocidad de datos de 1000 bps. Emplear
la función
% randn() para obtener valores aleatorios que luego sean
convertidos
% a 1’s y 0’s.

% Nd = número de valores digitales binarios
Nd = 1024;
dt = round(rand(1, Nd));

% velocidad de datos de 1000bps = 1KHz
Vd = 1000; % 1000 Hz
% Tb = Tiempo de bit
Tb = 1/Vd; % 1 ms
% Definición de portadora a 10KHz
% Frecuencia de portadora Fp
Fp = 20*Vd;
Tp= 1/Fp;
Fs = 10*Fp; %Frecuencia de muestreo para la señal portadora

Ts = 1/Fs; %
% Nm = número de muestras por bit
Nm = Tb/Ts;
% Tiempo total de 1024 bits
t = 0:(Tb/Nm):((Nd*Tp*Nm)-((Tp/Nm)*Nm));
w = 2*pi*Fp;

Ct = Nd/2; % Tiempo después de la compresión o codificación
Ct=Coded time

As = 1; % Amplitud de la señal
% Formar pares secuenciales de bits que produzcan las fases
% correspondientes. Los valores de las fases pueden definirse
libremente.
Si = zeros(1, Ct);
Sq = zeros(1, Ct);
Ph = zeros(1, Ct);
indx1 = 1:2:Nd;
indx2 = 2:2:Nd;

% Generar a través del codificador los valores correspondientes
de las
```

```
% señales Si(t) y Sq(t) para generar las fases previamente
definidas

%====================CODIFICADOR=================================
===========
%                              X    Y
%  | d1(t),d0(t) | Phase | Si | Sq |                Sq
%  |-------------------------------|               1 + (0,1)
%  |    (0,0)    |  0    | 1  | 0  |                 |
%  |-------------------------------|                 |
%  |    (0,1)    | pi/2  | 0  | 1  |         (1,0)   |   (0,0)
%  |-------------------------------|       ----+-----+-----+---
Si
%  |    (1,0)    |  pi   | -1 | 0  |           -1      |0    1
%  |-------------------------------|                 |
%  |    (1,1)    | 3pi/2 |  0 | -1 |                 |
%  |-------------------------------|              -1 + (1,1)

for i=1 : Ct
          % d0(t)                    d1(t)
    if((dt(indx1(i)) == 0) && (dt(indx2(i))==0))
    Si(i) = As*cos(sym(0));
    Sq(i) = As*sin(sym(0));
    end
    if((dt(indx1(i)) == 0) && (dt(indx2(i))==1))
    Si(i) = As*cos(sym(pi/2));
    Sq(i) = As*sin(sym(pi/2));
    end
    if((dt(indx1(i)) == 1) && (dt(indx2(i))==0))
    Si(i) = As*cos(sym(pi));
    Sq(i) = As*sin(sym(pi));
    end
    if((dt(indx1(i)) == 1) && (dt(indx2(i))==1))
    Si(i) = As*cos(sym((3*pi)/2));
    Sq(i) = As*sin(sym((3*pi)/2));
    end

end
%============================================================
===========
% Como Si & Sq tienen una longitud de 512, se extenderá para
coincidir con
% t ya que son 100 muestras por bit
SiXt = zeros(1, Ct*Nm);
SqXt = zeros(1, Ct*Nm);
for i=1:Ct
   for j=1:Nm
```

```
        SiXt((Nm*(i-1))+j) = Si(i);
        SqXt((Nm*(i-1))+j) = Sq(i);
    end
end

% Transmitir al canal de comunicación la señal equivalente pasa-
baja
% formada por la adición de Si(t) y Sq(t).
ft = t(1:Nm*Ct);
TxSt = SiXt.*cos(w.*ft) - SqXt.*sin(w.*ft);
figure;
subplot(2,1,1);
plot(ft, SiXt.*cos(w.*ft)); grid on;
title('Señal en fase (I)');
xlabel('Tiempo (t)');
ylabel('Amplitud');
subplot(2,1,2);
plot(ft, SqXt.*sin(w.*ft)); grid on;
title('Señal en cuadratura (Q)');
xlabel('Tiempo (t)');
ylabel('Amplitud');

BkUp = TxSt;
%=============================================================
===========
% Agregar a la señal transmitida anterior el ruido n(t), el cual
posee una
% distribución gaussiana normalizada (randn()). Variar la
potencia del
% ruido de acuerdo a lo que se indique posteriormente.

AN = 0:0.02237:1; % Amplitud del ruido
BER = zeros(1, 10);
dB =  zeros(1, 10);
TE2 = zeros(1, 10);
NR = zeros(1, 10);
for R = 1: 10

figure;
subplot(2,1,1);
plot(ft, BkUp); grid on;
title('Modulación 4PSK sin ruido');
xlabel('Tiempo (t)');
ylabel('Amplitud');

ntSi = AN(R)*randn(1, (Ct*Nm));
```

```
ntSq = AN(R)*randn(1, (Ct*Nm));

% El ruido debe afectar de manera independiente a cada parte de
la señal
% equivalente pasa-bajas, es decir, las señales de ruido que se
le añade
% a Si(t) y Sq(t) son independientes.

Si_n = SiXt + ntSi;
Sq_n = SqXt + ntSq;

% Señal equivalente pasa-bajas + ruido
TxSt = Si_n.*cos(w.*ft) - Sq_n.*sin(w.*ft);

subplot(2,1,2);
plot(ft, TxSt); grid on;
title('Modulación 4PSK + ruido');
xlabel('Tiempo (t)');
ylabel('Amplitud');

% Diagrama de fases de la señal en el transmisor
DiagFTx = [SiXt', SqXt'];
scatterplot(DiagFTx);
grid on;

% Diagrama de fases de la señal en el receptor con ruido
DiagFRx = [Si_n', Sq_n'];
scatterplot(DiagFRx);
grid on;
%=============================================================
===========
% El receptor debe ser capaz de producir la información digital
binaria a
% partir del equivalente pasa-bajas de la señal con ruido que
recibe.
%b = [0.0145 0.0306 0.0725 0.1245 0.1665 0.1826 0.1665 0.1245
0.0725 0.0306 0.0145];
%a = 1;

Fc = Fp;
[b, a] = butter(10, Fc/(Fs/2));
% Quitamos el ruido que se agrega en el canal
DataRxNL = filter(b, a, TxSt);

RxSi = zeros(1, Ct*Nm);
RxSq = zeros(1, Ct*Nm);
```

```
RxSq = DataRxNL.*cos(w.*ft);
RxSi = DataRxNL.*sin(w.*ft);

FSiRx = filter(b, a, RxSi);
FSqRx = filter(b, a, RxSq);

figure;
subplot(2,1,1);
plot(ft, FSiRx); grid on;
title('Demodulación 4PSK -> S(i)');
xlabel('Tiempo (t)');
ylabel('Amplitud');

subplot(2,1,2);
plot(ft, FSqRx); grid on;
title('Demodulación 4PSK -> S(q)');
xlabel('Tiempo (t)');
ylabel('Amplitud');

%=========================CODIFICADOR EN EL
RECEPTOR======================
% El receptor debe ser capaz de producir la información digital
binaria a
% partir del equivalente pasa-bajas de la señal con ruido que
recibe

% Número de muestras a evaluar de la señal demodulada y
filtrada.
% 10% del tamaño total del bit
Samp = Nm * 0.1;
% Punto medio del bit recibido a codificar
Pm = Nm/2;
% Vector de datos recibidos para Si & Sq en la recepción
RxSiData = zeros(1 , Ct);
RxSqData = zeros(1 , Ct);

for k = 1 : Ct

    tmpSIRx_Data = FSiRx((((k*Nm)-Pm) - (Samp/2)): (((k*Nm)-Pm)
+ (Samp/2)));
    tmpSQRx_Data = FSqRx((((k*Nm)-Pm) - (Samp/2)): (((k*Nm)-Pm)
+ (Samp/2)));
    tmpAvgSI = sum(tmpSIRx_Data)/Samp;
    tmpAvgSQ = sum(tmpSQRx_Data)/Samp;

    % decodificación del "0"
```

```
    if((tmpAvgSI < 0.1) && (tmpAvgSI > -0.1))
        RxSiData(k) = 0;
    else
        % Decodificación del "1"
        if((tmpAvgSI < 0.7) && (tmpAvgSI > 0.35))
            RxSiData(k) = 1;
        else
            % Decodificación del "-1"
            if((tmpAvgSI < -0.25) && (tmpAvgSI > -0.7))
                RxSiData(k) = -1;
            end
        end
    end

    % decodificación del "0"
    if((tmpAvgSQ < 0.1) && (tmpAvgSQ > -0.1))
        RxSqData(k) = 0;
    else
        % Decodificación del "1"
        if((tmpAvgSQ < 0.7) && (tmpAvgSQ > 0.35))
            RxSqData(k) = 1;
        else
            % Decodificación del "-1"
            if((tmpAvgSQ < -0.25) && (tmpAvgSQ > -0.7))
                RxSqData(k) = -1;
            end
        end
    end
if((RxSiData(k) == 1)&& (RxSqData(k)== 0))
    RxSiData(k) = 0;
    RxSqData(k) = 0;
end
if((RxSiData(k) == 0)&& (RxSqData(k)== 1))
    RxSiData(k) = 0;
    RxSqData(k) = 1;
end
if((RxSiData(k) == -1)&& (RxSqData(k)== 0))
    RxSiData(k) = 1;
    RxSqData(k) = 0;
end
if((RxSiData(k) == 0)&& (RxSqData(k)== -1))
    RxSiData(k) = 1;
    RxSqData(k) = 1;
end
end
RxVector = zeros(1 , Nd);
% Reordenamiento de los datos recibidos de paralelo a serie
```

```
for m = 1 : Ct

RxVector(indx1(m)) = RxSiData(m);
RxVector(indx2(m)) = RxSqData(m);

end

% Cálculo del número de bits recibidos erroneamente
TE = 0;
for l=1:Nd
    if(dt(l) ~= RxVector(l))
        TE = TE + 1;
    end
end
% calculo del Bit Error Rate
% BER = EB/NB donde EB(TE): es el número de bis recibidos
incorrectamente y NB(Nd):
% es el número total de bits transmitidos
TE2(R) = TE;
BER(R) = TE/Nd;

% Para graficar BER vs Eb/No necesitamos obtener Eb y No.
% Eb = ((A^2)*T)/2 donde Eb = Energia del bit y No es ka
densidad espectral
% de potencia del ruido
% No = 2*(var^2)
% Tb = velocidad de transmision de los datos T
v = ((var(ntSi) + var(ntSq))/2)^2;
NR(R) = (var(ntSi) + var(ntSq))/2;
No = 2*v;
A = 1;

Eb = ((A^2)*Tb)/2;
EbNo = Eb/No;

dB(R) = 10*log10(EbNo);

end

fprintf('No. Errores ----  B.E.R.  ------ Ruido ----- Eb/No (dB)
\n');
for fp = 1: R
    fprintf('   %2.0f       ----', TE2(fp) );
    fprintf(' %1.5f  ----', BER(fp) );
    fprintf('  %1.5f  ----', NR(fp) );
```

```
    fprintf(' %2.0f \n', dB(fp) );
end

figure; plot(dB, BER);
title('Relación BER vs Eb/No');
ylabel('Bit Error Rate');
xlabel('Eb/No (dB)');
grid on;
```

Conclusiones:

Este proyecto fue complicado al inicio ya que al momento de filtrar la señal recibida esta se distorsionaba y no era posible reconstruir la señal original. Al agregar una portadora fue la solución para ese problema. Hubo varios elementos que contribuyeron a mejorar la recepción y decodificación dependiendo de su configuración, como la amplitud de la señal transmitida, el nivel de ruido que se agregó al pasar por el canal de transmisión, el filtro utilizado para eliminar el ruido agregado en el canal, y los comparadores de nivel al momento de determinar si el bit recibido era ‘0’, ‘1’ o ‘-1’.

C. REPORTE DE K64 AUDIO PLAYER V2.0 Y AMPLIFICADOR DE AUDIO

ITESO

Development of Printed Circuit Boards PCB

K64 AUDIO PLAYER V2.0 Y AMPLIFICADOR DE AUDIO

Juan Esteban Bernardo Aldana

The aim of this document is to describe the design flow process of the PCB development of an audio amplifier and the K64 Audio Board.
I also included some tricks and tips learned during the process and while learning the tools capabilities, especially in Cadence Allegro.
This report includes a summary with the entire process followed in order to make the projects mentioned before, starting at creating your EBOM, padstacks, footprints to routing, placement and manufacturing order.

Contents

PCB Design flow 118

ELECTRICAL DESIGN 118
MECHANICAL DESIGN 118
COMPONENTS SELECTION (BOM) 119
FOOTPRINTS 119
PADSTACKS 120
PACKAGE SYMBOLS 121
ELECTRICAL SYMBOLS 122
SCHEMATIC DRAWING 126
DESIGN RULES CHECK (DRC) 127
NETLIST CREATION 127

Cadence PCB Editor 129

BOARD SETTINGS 129
BOARD WIZARD 130
IMPORT NETLIST 135
DESIGN FOR ASSEMBLY (DFA) 137
PLACEMENT 138
CONSTRAINTS 138
Routing 139
IMPEDANCE 140
PLANES/SHAPES 140
MANUFACTURING 141
TESTS 144
CONCLUSIONS 145
LESSONS LEARNED 145
Delivery status ..... **Error! Bookmark not defined.**

3. PCB Design flow

3.1. Electrical design

Electrical design is the first part of the development process. It is needed to define all the elements and components to be used in our design for our target project.

3.2. Mechanical design

It is also important to have well defined the container for our board and be aware of the dimensions of the final project.

3.3. Components selection (BOM)

Based on electrical requirements, it is necessary to create the components list or BOM (Bill of Materials). It is recommendable to select all components from the same provider if possible, verify availability, cost, price per unit, minimum quantity and package.
Well known providers are:
Digikey, Mouser and Newark among others.
www.**digikey**.com
www.**mouser**.mx
http://mexico.newark.com/
Figure 1 show the BOM used for the Audio Amplifier project

Component	Value	Qty	DigiKey Part Number	Manufacturer Part number	Manufacturer	Description	Available Qty	Price per unit USD	Minimum Qty
Capacitor	1.0uF	4	478-8302-1-ND	F931E105MAA	AVX Corporation	CAP TANT 1UF 25V 20% 1206	25,817	0.32	1
Capacitor	470uF	4	493-6225-1-ND	UUR1E471MNL1GS	Nichicon	CAP ALUM 470UF 20% 25V SMD	4,179	0.68	1
Capacitor	0.1uF	3	490-1524-1-ND	GRM188R71E104KA01D	Murata Electronics	CAP CER 0.1UF 25V X7R 0603	3,551,728	0.1	1
Capacitor	10uF	1	490-5523-1-ND	GRM21BR61E106KA73L	Murata Electronics	CAP CER 10UF 25V X5R 0805	681,601	0.19	1
Capacitor	0.22uF	2	490-3290-1-ND	GRM188R71E224KA88D	Murata Electronics	CAP CER 0.22UF 25V X7R 0603	435,399	0.14	1
Inductor	22uH	2	587-2411-1-ND	NRS5020T220MMGJ	Taiyo Yuden	FIXED IND 22UH 1A 312 MOHM SMD	3,930	0.45	1
Resistor	4.7k	2	P4.7KJCT-ND	ERJ-2GEJ472X	Panasonic Electronic	RES SMD 4.7K OHM 5% 1/10W 0402	12,247,383	0.1	1
Capacitor	0.68uF	2	399-8114-1-ND	C0805C684K5RACTU	Kemet	CAP CER 0.68UF 50V X7R 0805	81,489	0.26	1
TPA3125D2	NA	1	296-24090-5-ND	TPA3125D2N	Texas Instruments	IC AMP AUDIO PWR 10W STER 20DIP	475	2.79	1
Terminal Block	2 PIN	2	277-1667-ND	1935161	Phoenix Contact	TERM BLOCK PCB 2POS 5.0MM GREEN	24,058	0.37	1

Figure 1- (BOM for audio amplifier)

Using BOM as reference, it is important to download all datasheets from manufacturer's webpage (recommended). They're important not only to know their electrical characteristics but their respective dimensions.

3.4. Footprints

A footprint or land pattern is the arrangement of pads (in surface-mount technology) or through-holes (in through-hole technology) used to physically attach and electrically connect a component to a printed circuit board. The land pattern on a circuit board matches the arrangement of leads on a component.
To calculate the required land patterns for the Audio amplifier project, the LP Calculator tool was used. Figure 2 shows some important parameters introduced in this tool, like pitch, component length, width and pin count that will be used to calculate the appropriate land pattern for this type of components and packages as shown in Figure 3.

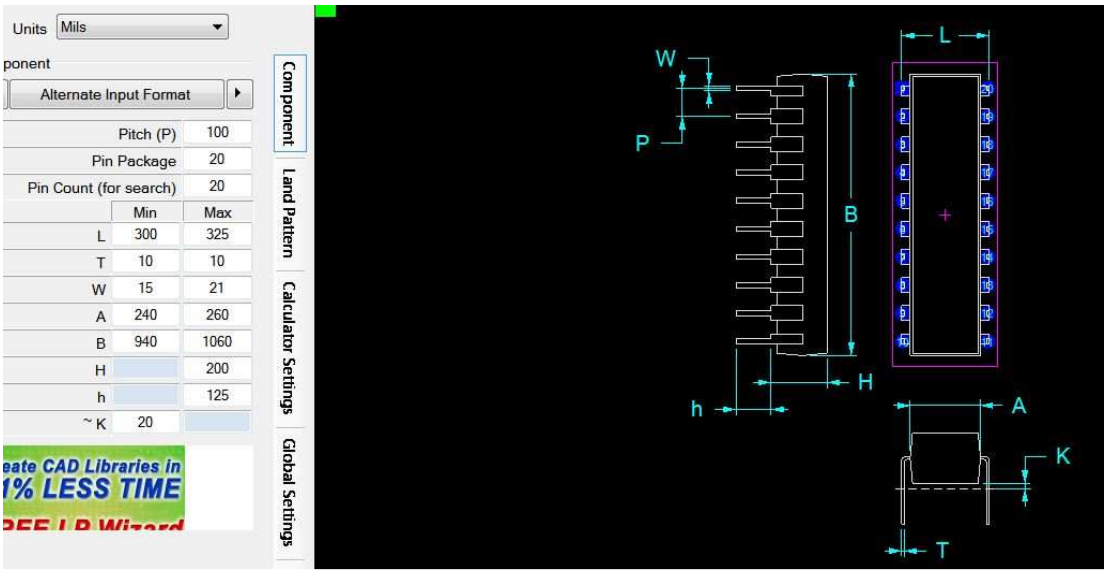


Figure 2 – (Component parameters)

The tool also offers the possibility to introduce parameter units in Mils, millimeters or inches and make respective conversions if needed.

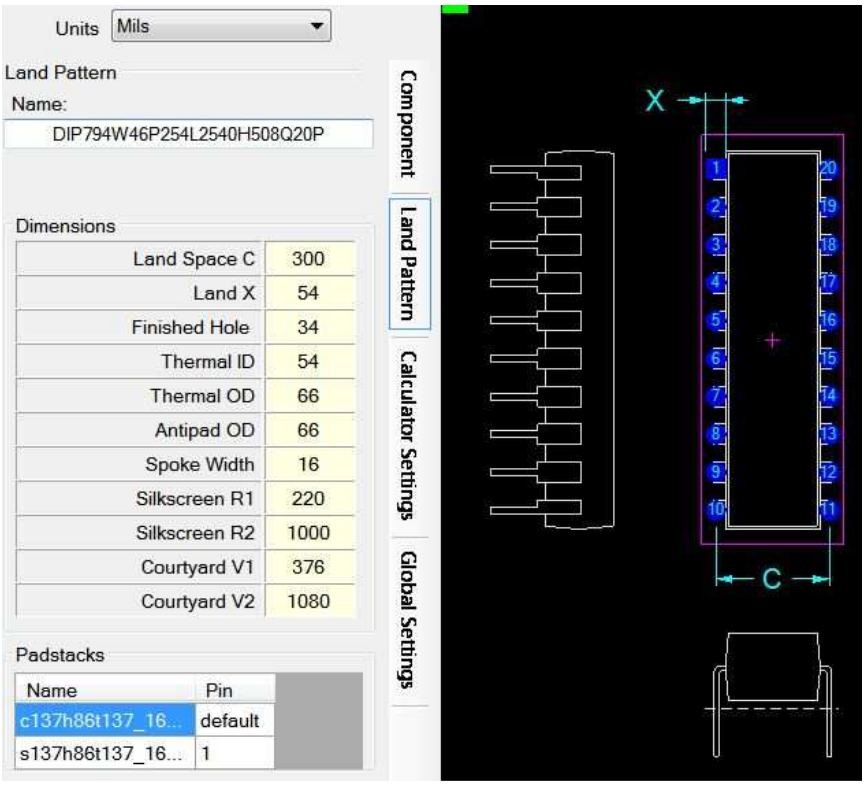


Figure 3 – (Land Pattern calculated)

3.5. Padstacks

The creation of padstacks, was performed using the information calculated with the LP Calculator, especially the data generated in land pattern tab.

Additional aspects to consider in the creation of padstacks:

- Assign a specific folder to save all the padstacks of your project
- Use a naming convention or descriptive names that help you quickly identify the type of pad you are creating.
- Reuse padstacks as much as possible of existent designs.
- Mounted holes are not plated and are defined as mechanical symbols

The padstack name o54x54t.pad, describes an octagonal pad with a width 54 and height 54 through hole
Following padstacks were imported in Audio Amplifier from K64-Audio design:
c025s.pad
c030sxp.pad
c035-035n.pad
c043-043n.pad
c060-040t.pad
c063-063n.pad
c067-043t.pad

Following padstacks were created specifically for Audio Amplifier:
o54x54t.pad
r23x25s.pad
r48x57s.pad
r70x50s.pad
s54x54t.pad

3.6. Package Symbols

Package symbols were created using Package wizard in allegro.

Additional aspects to consider in the creation of package symbols:

- Use specific folder to save your package designs
- Use descriptive names that helps you quickly identify the package type of your symbol
- Make sure your padpath points to the correct pad folder container previously established in the padstacks section
- Reuse as much as possible existing symbols previously tested and make sure of the existence of its correspondent padstacks

In Audio amplifier project only the following component packages were created and all others were imported from K64 Audio project:
FIXED IND 22UH 1A 312 MOHM SMD

- IND_22uH_SMD.dra
- IND_22uH_SMD.psm

CAP ALUM 470UF 20% 25V SMD

- CAP_C470uF.dra
- CAP_C470uF.psm

IC AMP AUDIO PWR 10W STER 20DIP

- TPA3125D2.dra
- TPA3125D2.psm

TERM BLOCK PCB 2POS 5.0MM GREEN

- TBlock2pin_H10v2.dra
- TBlock2pin_H10v2.psm

3.7. Electrical Symbols

Allegro contains several electrical symbols that can be reused in your project. In Audio Amplifier all electrical symbols were reused following the procedure described below.
Open Orcad Capture and create a new library. Save it as something intuitive.
File-> New Library

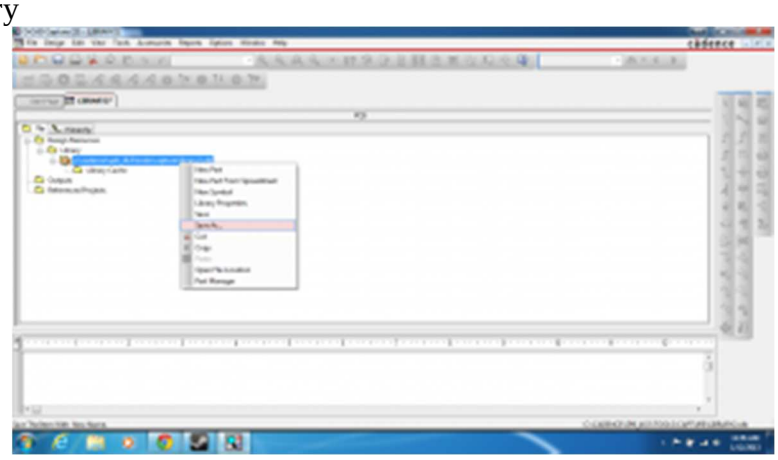


Figure 4 – (New Library creation)

Open the library with the symbol you want to copy.
Right Click->Copy the symbol.

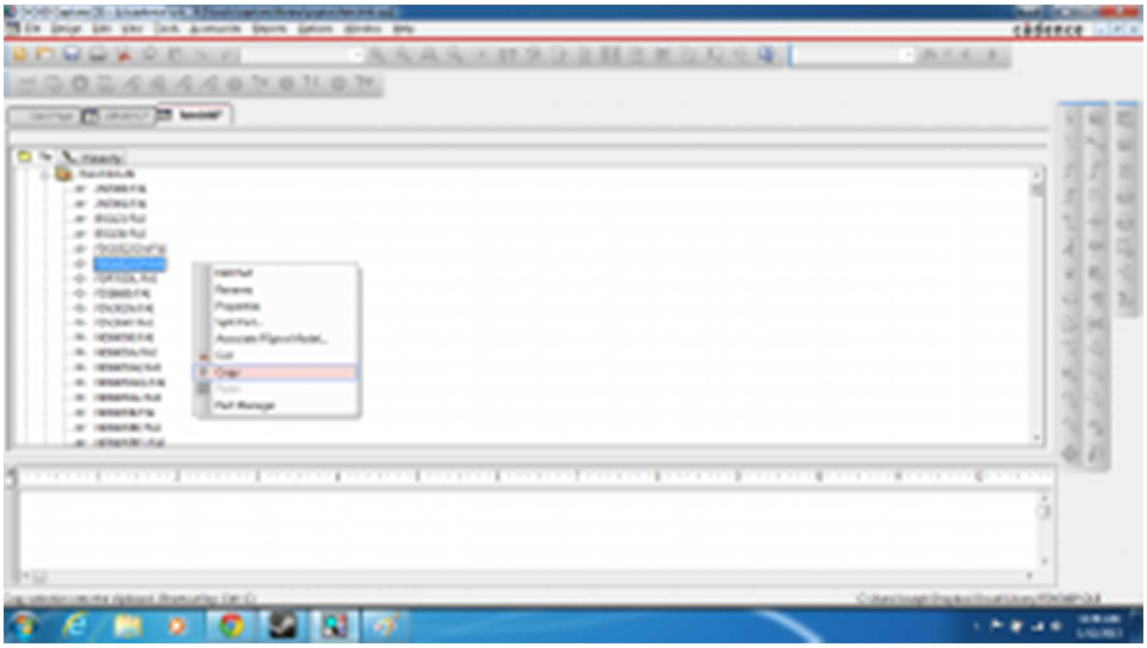


Figure 5 – (New Symbol creation)

Right Click->Past on your library.
Right Click->Rename the part.
Right Click->Edit part

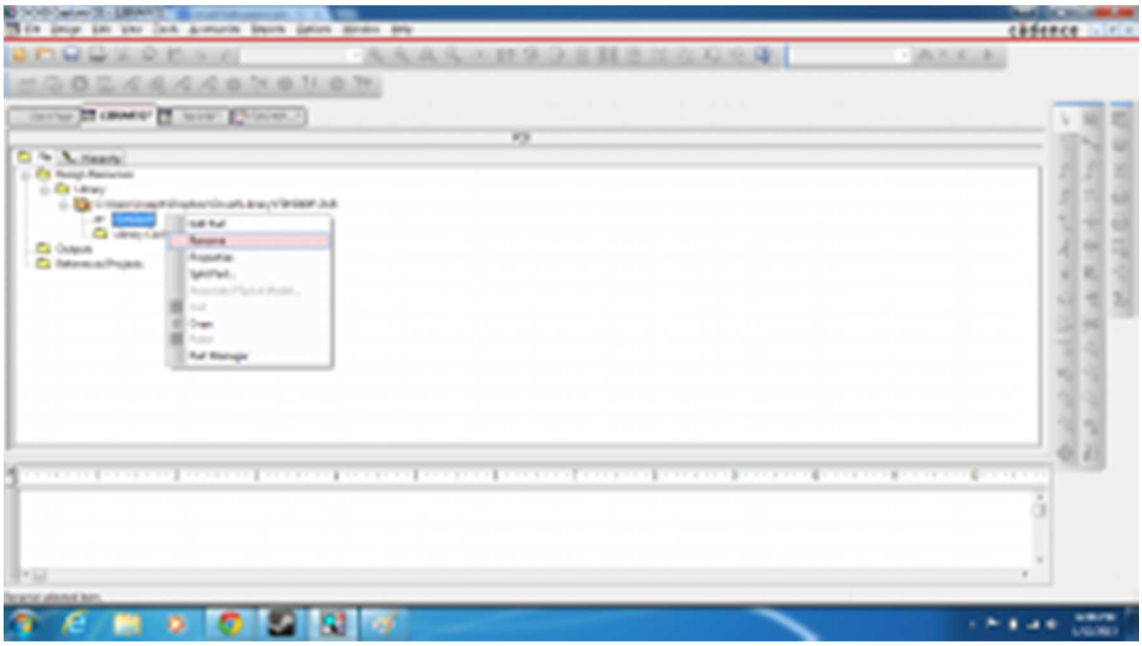


Figure 6 – (Rename part)

Options->Package Properties
Change Reference to Q

Change PCB footprint name to its correspondent package symbol name previously defined in Package symbols section.
Orcad should be able to locate your footprint if you enter its name in “PCB Footprint.” If not, you may have to add your directory in the psmpath defined in Allegro PCB Designer tool.

Finally change the part name and your symbol is set up.

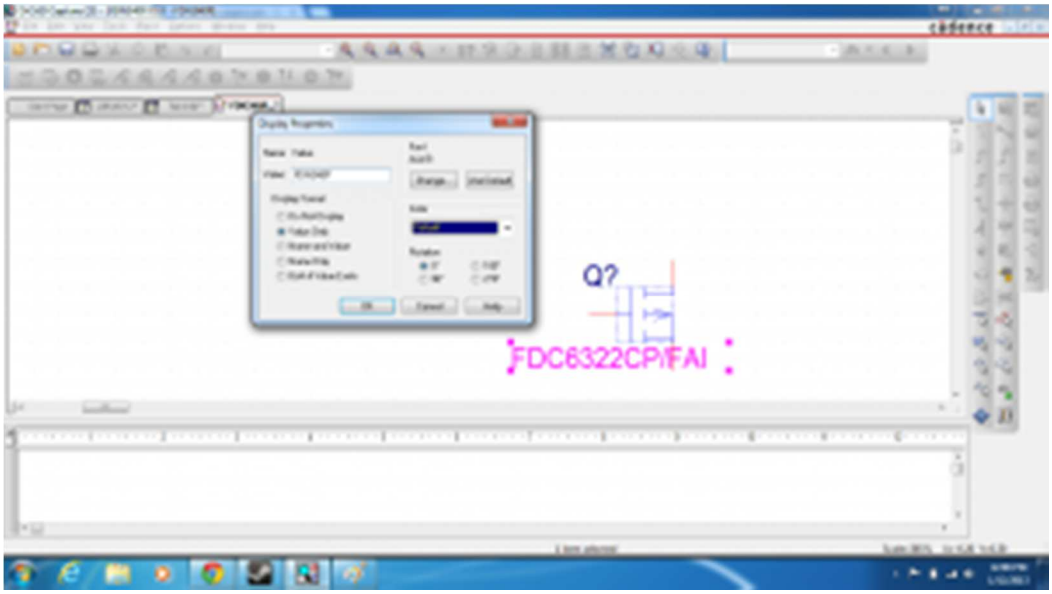


Figure 7 – (Part name modification)

Once the symbol is created, it can be located by loading the library in the Place Part menu.

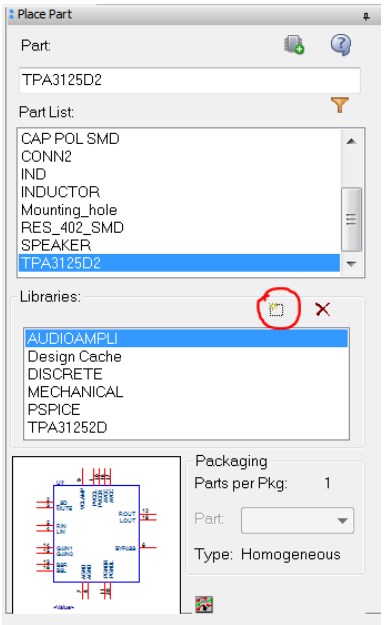


Figure 8 – (Loading symbols library)

The modified symbols already placed in the schematic worksheet have to be refreshed in Design Cache folder by right clicking on the modified component and selecting Update Cache option

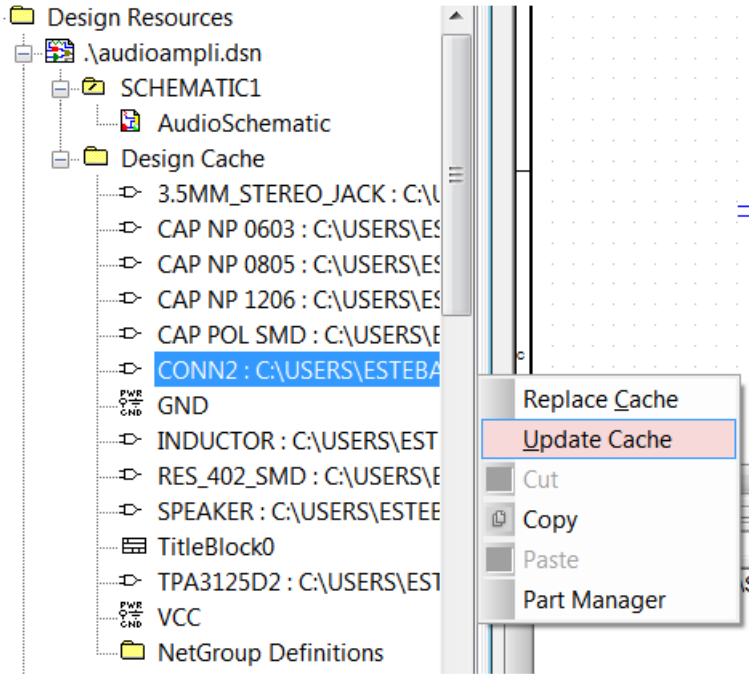


Figure 9 – (Design Cache update)

To create IC TPA3125D2, right click on your created library and select New Part From Spreadsheet option.

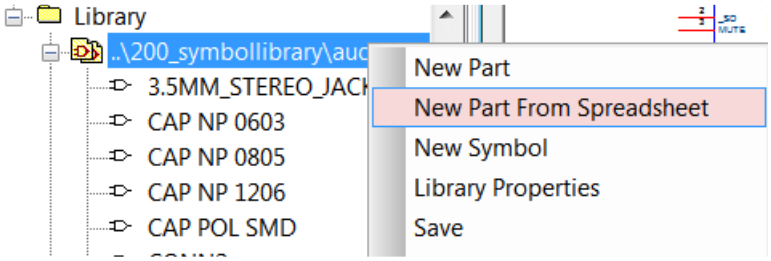


Figure 10 – (Create new for ICs)

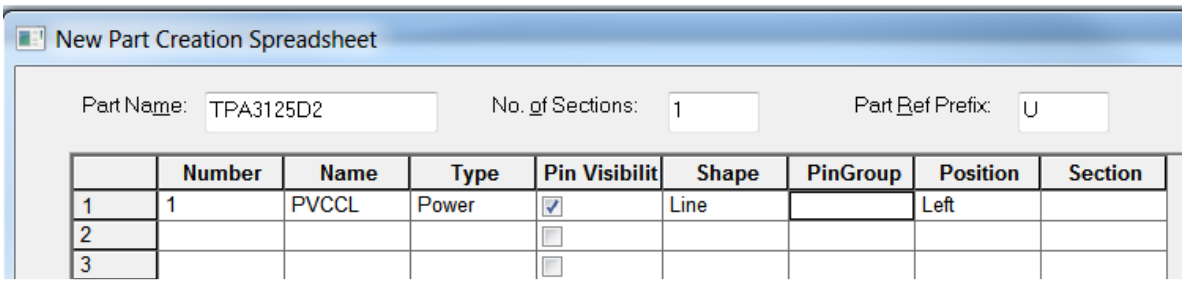


Figure 11 – (New part spreadsheet window)

Set Part Name, Number, Name, Type Pin Visibility, Shape & Position.
Names can be obtained from Terminal functions table in datasheet document

Positions are considered as follows:
Inputs -> Left & Outputs ->Right

Table 1. TERMINAL FUNCTIONS

TERMINAL		I/O	DESCRIPTION
NAME	20-PIN (DIP)		
PVCCL	1		Power supply for left channel H-bridge.
$\overline{SD}$	2	I	Shutdown signal for IC (low = outputs disabled, high = operational). TTL logic levels with compliance to AVCC.
MUTE	3	I	Mute signal for quick disable/enable of outputs (high = outputs switch at 50% duty cycle; low = outputs enabled). TTL logic levels with compliance to AVCC.
LIN	4	I	Audio input for left channel.
RIN	5	I	Audio input for right channel.

Figure 12 – (Datasheet section)

3.8. Schematic Drawing

Schematic is created by double clicking on symbol name in Place Part menu with the appropriate library loaded.

Select Place->Wire option to connect all components. It is also possible to add Net Alias to interconnect components by setting the same alias name to the desired nets by doing Select->Net Alias option.

The schematic created for Audio Amplifier is shown in image below.

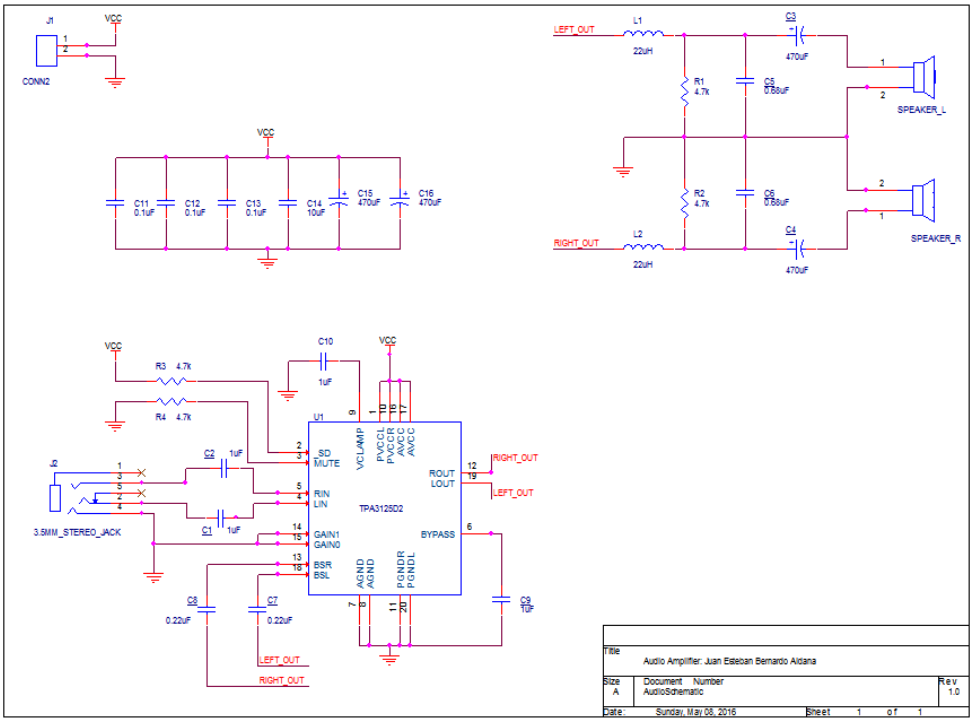


Figure 13 – (Audio amplifier schematic)

3.9. Design Rules Check (DRC)

It's very important to run DRC check before netlist creation to make sure everything is properly connected and in place. No errors should be reported in DRC log results.
To run DRC check, select the schematic design and click on Tools->Design Rules Check.

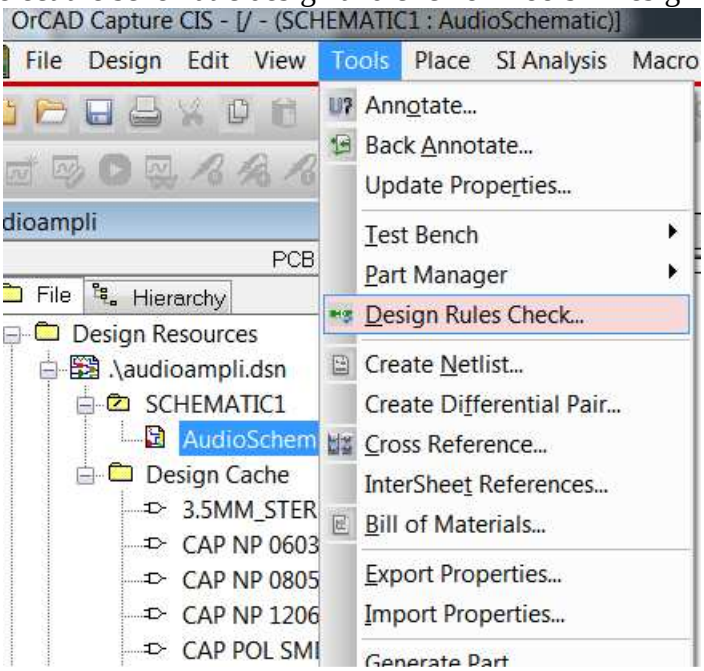


Figure 14 – (Design Rules Check option)

Verify log results report and there are no nets in the schematic circled in green.

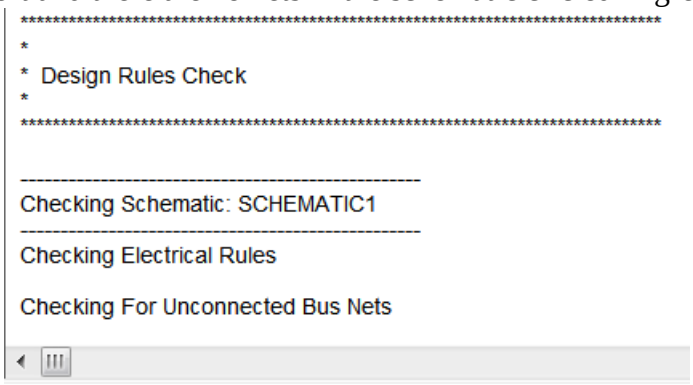


Figure 15 – (DRC report log)

3.10. Netlist creation

Once the DRC results have no errors, we can continue creating our netlist. The netlist is created to export, to our board design the connectivity of our just created schematic design.
In order to create the netlist, select the schematic design and click on Tools->Create Netlist option menu.

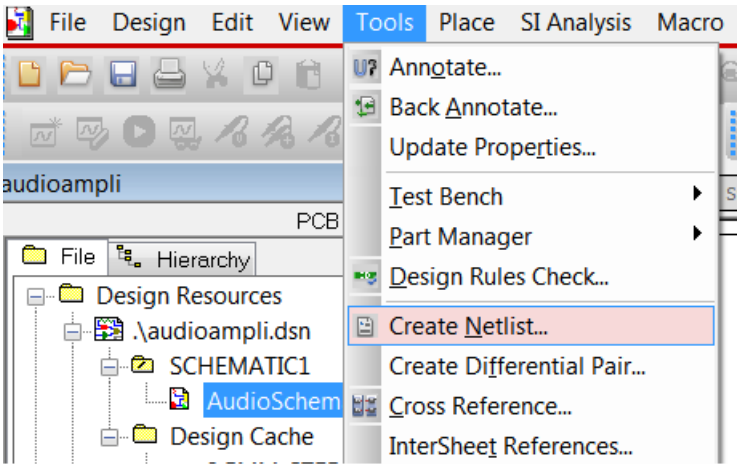


Figure 16 – (Netlist creation)

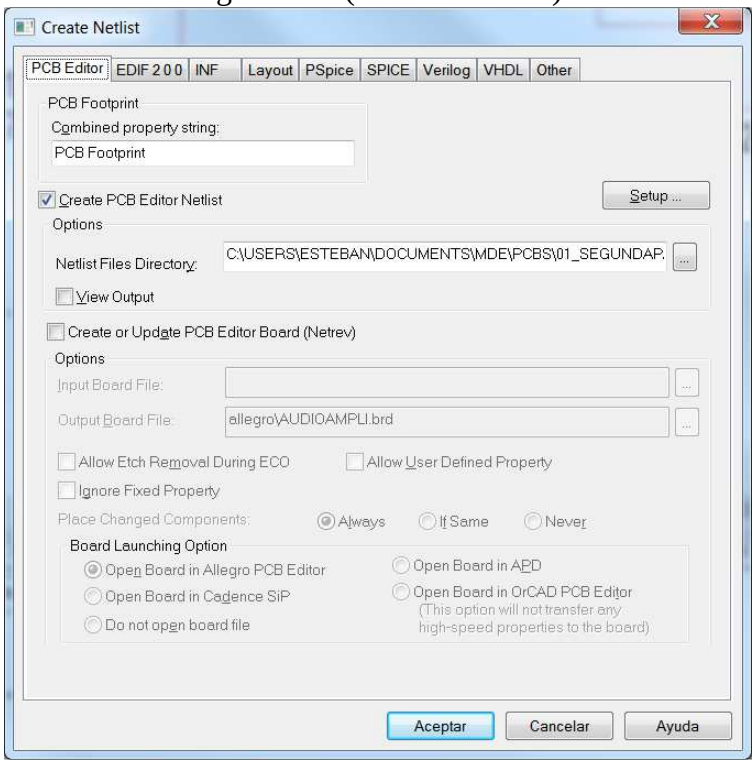


Figure 17 – (Create Netlist window)

Create a folder exclusively for your netlist files and set its location in the Netlist Files Directory section as shown in figure 17.
The files listed below will be created after clicking on Accept button.
pstchip.dat
pstxnet.dat
pstxprt.dat

4. Cadence PCB Editor

In this section, we will see mostly examples of the K64 Audio board based on the NXP K64 MCU and its design flow.

K64 Audio Board was a predefined project with specific requirements and inputs previously defined and designed like footprints, padstacks, symbols and schematic to integrate the final product Audio Board.

The following sections describe the design flow utilized to create the final board.

Once the netlist is created as specified in section Netlist Creation, and as new developers, it was a good practice to use the Board wizard in PCB Editor.

4.1. Board Settings

First of all, it is necessary to configure the board settings and paths for imported libraries. It is very important to perform this step since during importing netlist, our PCB Editor might not find all padstacks, symbols and footprints associated to the schematic design.

Open the User Preferences Editor window, and go to the Library folder inside the Paths folder
User Preferences->Paths->Library

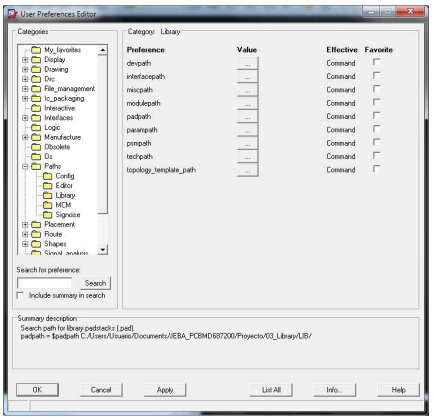


Figure 18 – (Path settings)

On the library category section, select the library directories of your project. In K64 Audio board project, only padpath and psmpath were set.

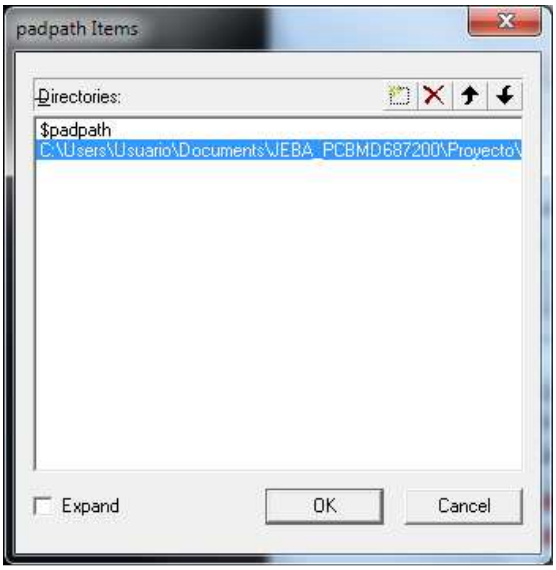


Figure 19 –(padpath directory configuration)

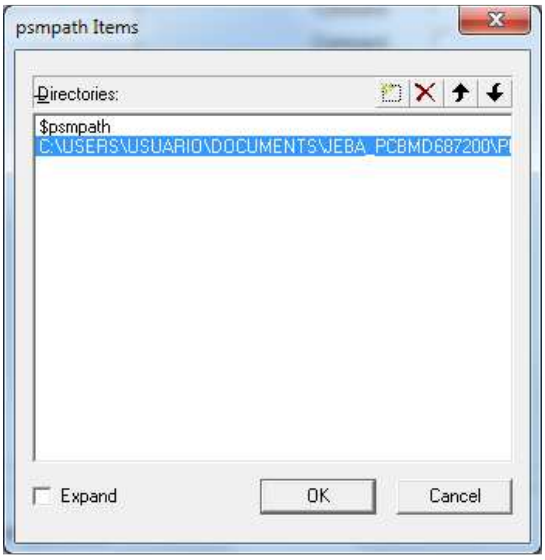


Figure 20 – (psmpath directory configuration)

4.2. Board Wizard

As it was previously mentioned, for a beginner is a good practice to use the board wizard to configure the initial work area of the PCB design.

In PCB Editor, click on new Drawing, then select the directory you are using for your work, set the Drawing Name and select Board (wizard) in Drawing Type option then click OK.

Figure 21 – (New Drawing)

In Board Template window, no board template was loaded for K64 Audio design. However for a different design, there might be a predefined template which can be loaded in this section.

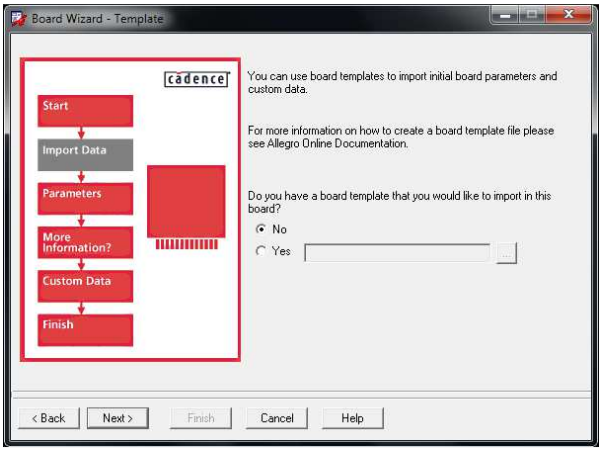


Figure 22 – (Board template selection)

There was not need to load any Tech or Parameter files in the next window, since they are loaded and configured in later sections.

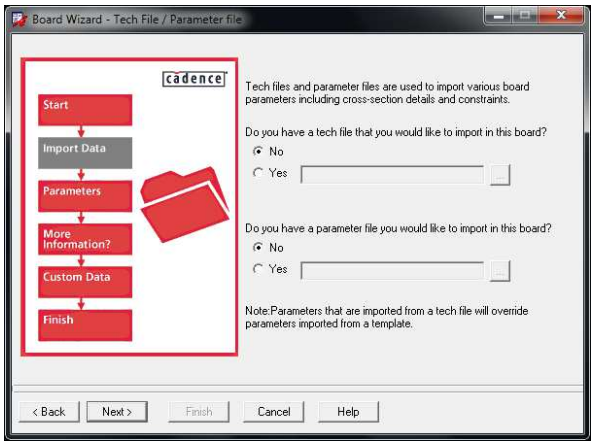


Figure 23 – (Tech/Parameter files selection)

Board Symbol section, is checked with the No option since outline was defined in a future procedure. This option is checked as Yes, just in case a predefined outline for a specific project is required.

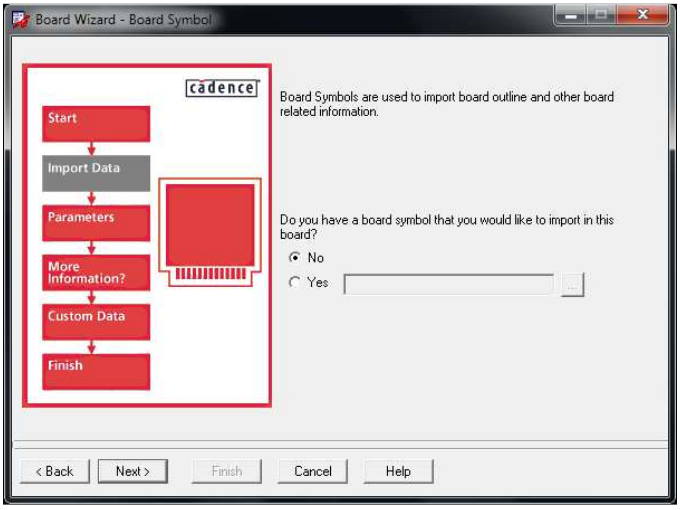


Figure 24 – (Board Symbol /Outline selection)

General Parameters are configured as described in the window below:

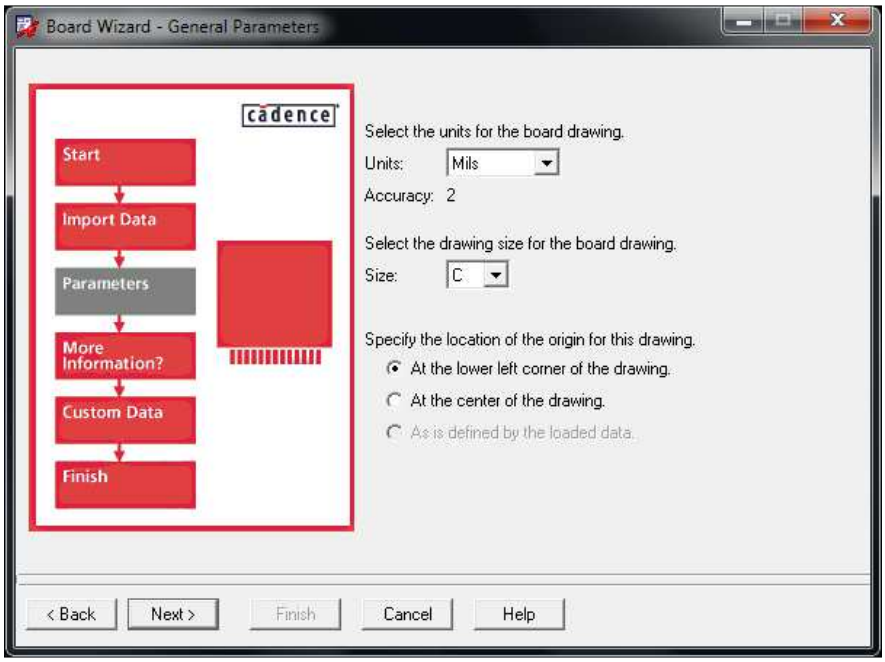


Figure 25 – (General parameters)

General Parameters Continued. It requires configuring the non-etching grid spacing. Four layer etch were specified in the requirements. Check Generate default artwork films. This helps to get initially configured the default views for Bottom and Top standard views.

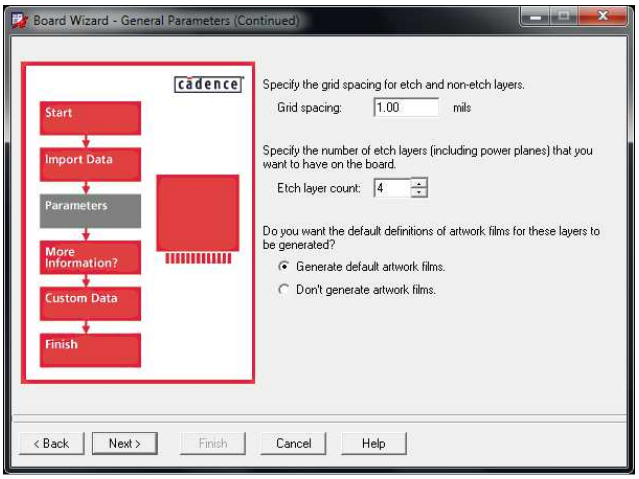


Figure 26 – (More General Parameters)

In order to assign a plane to each plane to be used in the design, the Etch Cross-section details window is helpful to define them.

For this project, it was required to use Top and Bottom layers as Routing planes and internal Layer2 and Layer3 as Power planes.

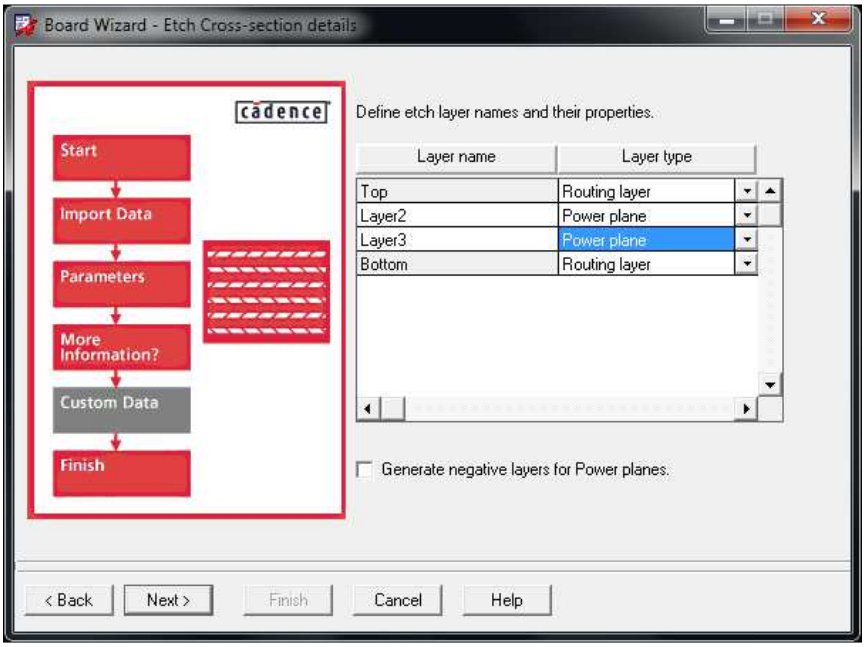


Figure 27 – (Etch Cross-section details)

In the Board Wizard Padstack browser, we set the default pad to be used for vias creation, for now C020-014t pad was selected.

If padpath directory for the library containing C020-014t pad was not properly set, you may not be able to see the specified pad in the padstack browser.

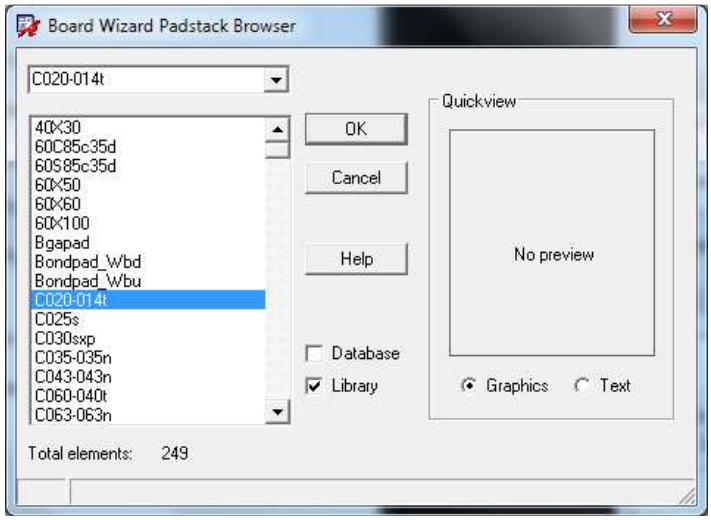


Figure 28 – (Padstack Browser selection)

Spacing Constraints are set to 5.0 mils by default. However those parameters can be modified in the constraint manager later.

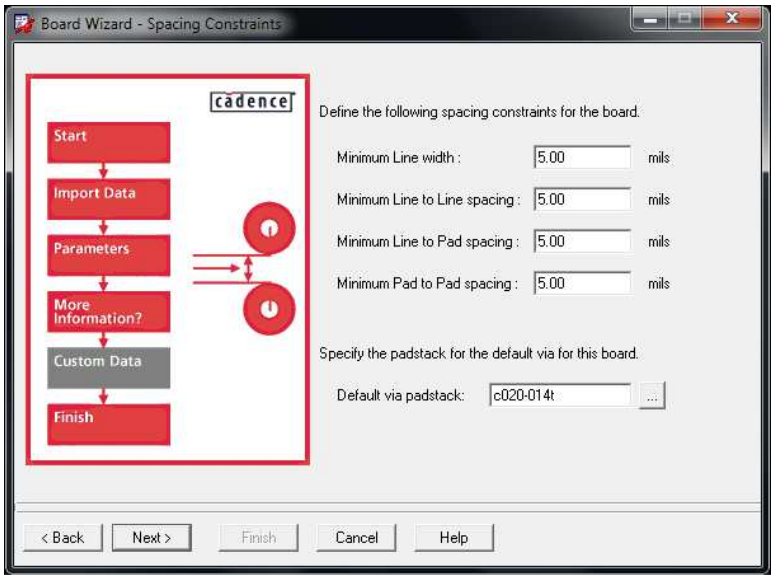


Figure 29 – (Spacing Constraints window)

Board definition Parameters are set in the window shown below. For K64 Audio board, the specified Width and Height are 2500 and 1000 mils respectively.

Route keeping distance is the limit distance to make any routing.
Package keeping distance is de limit distance to place any component.

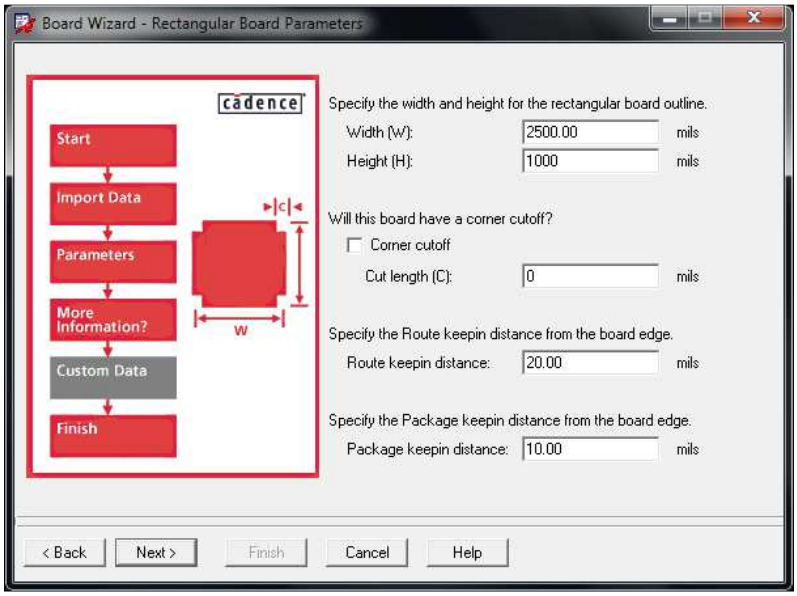


Figure 30 – (Board Parameters)

4.3. Import Netlist

To import the netlist previously created click on File->Import->Logic

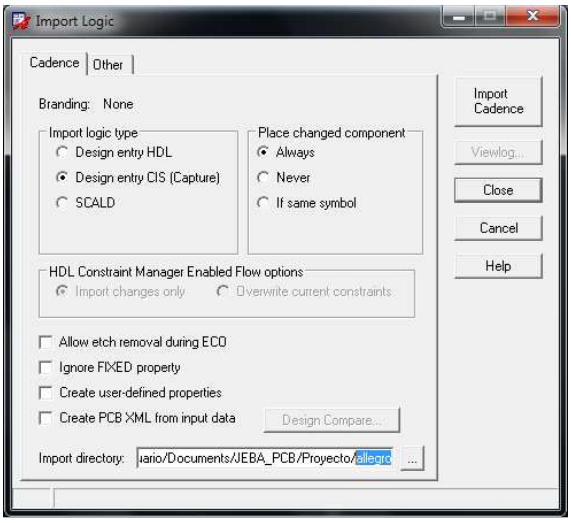


Figure 31 – (Import Netlist)

If all library paths were set correctly, when you click on Import Cadence button will be successful. Make sure you select correctly your netlist folder to browse.

After importing the netlist, it is necessary to place all components recently imported by clicking on Setup->Quickplace option. Unplaced symbols count is visible above the buttons section.

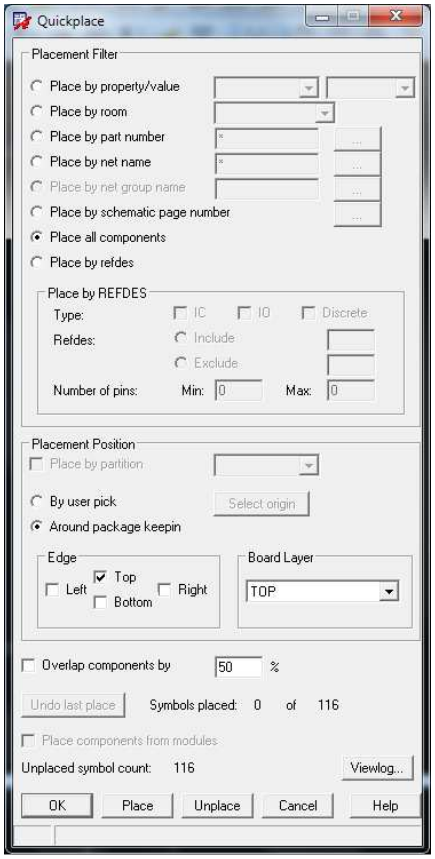


Figure 32 – (Quickplace option menu)

After Click OK button, unplaced symbol count shall be 0

4.4. Design for Assembly (DFA)

Design for assembly (DFA) is a process by which products are designed with ease of assembly in mind. If a product contains fewer parts it will take less time to assemble, thereby reducing assembly costs. In addition, if the parts are provided with features which make it easier to grasp, move, orient and insert them, this will also reduce assembly time and assembly costs. The reduction of the number of parts in an assembly has the added benefit of generally reducing the total cost of parts in the assembly. This is usually where the major cost benefits of the application of design for assembly occur.

The K64 Audio board has its own predefined DFA table which was provided as an input for the project. The procedure followed to import that DFA table is described below.

Go to File->Open and load the High_Density.dfa file then click Ok button

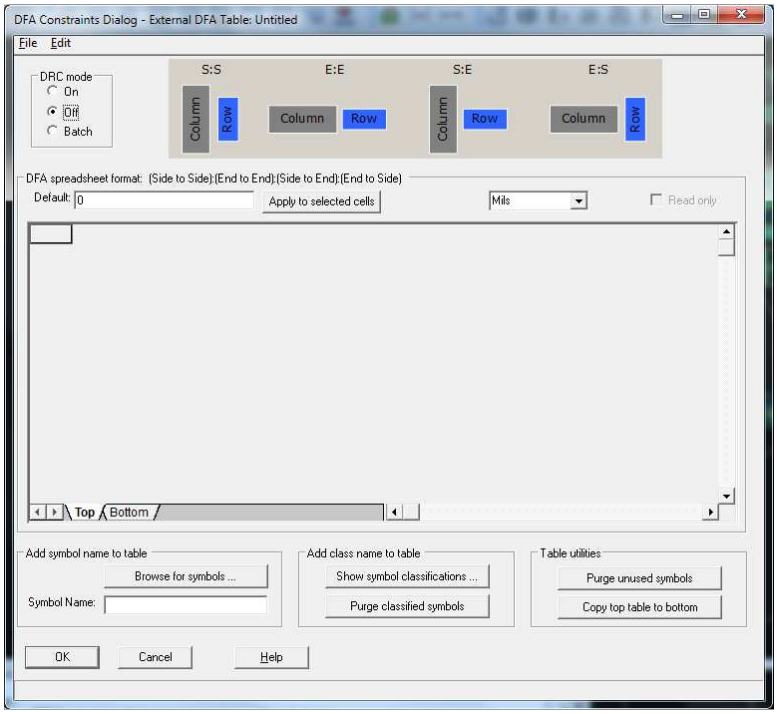


Figure 33 – (DFA Table import window)

4.5. Placement

Placement was performed considering specifications defined in the document K64_Audio_LDG.doc section 8 Component Placement Requirements.
Additional considerations no listed are:

Power supply shall be placed as far as possible from audio codec

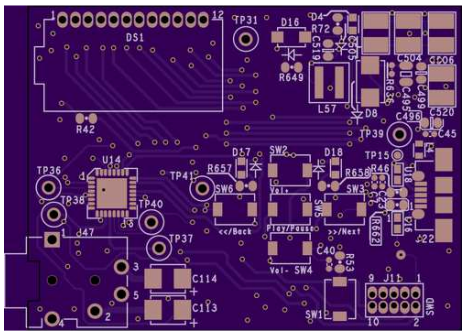


Figure 34 – (Final placement TOP)

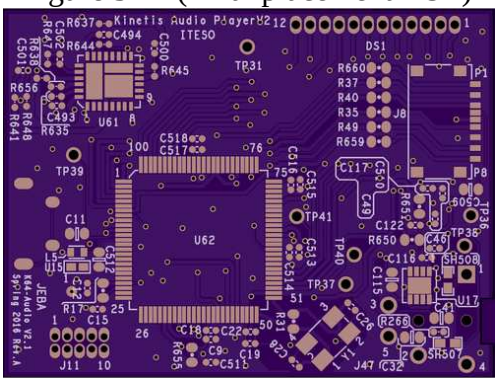


Figure 35 – (Final placement BOTTOM)

4.6. Constraints

Constraint manager allow us to configure the rules to follow during our routing and placement. For the K64 Audio board the constraints were provided as input and were made for this specific project. It is not possible to import the same constraints for a different project since the netlist may differ.

Constraint Manager lets you define, view, and validate constraints at each step in the design flow, from design capture (in Allegro® Design Entry HDL or System Connectivity Manager) to floorplanning (in Allegro® PCB SI L, XL, and GXL to design realization (in Allegro® PCB L,

XL, GXL, and OrCAD PCB Editor). You can also use Constraint Manager with SigXplorer to explore circuit topologies and derive electrical constraint sets which can include custom constraints, custom measurements, and custom stimulus.

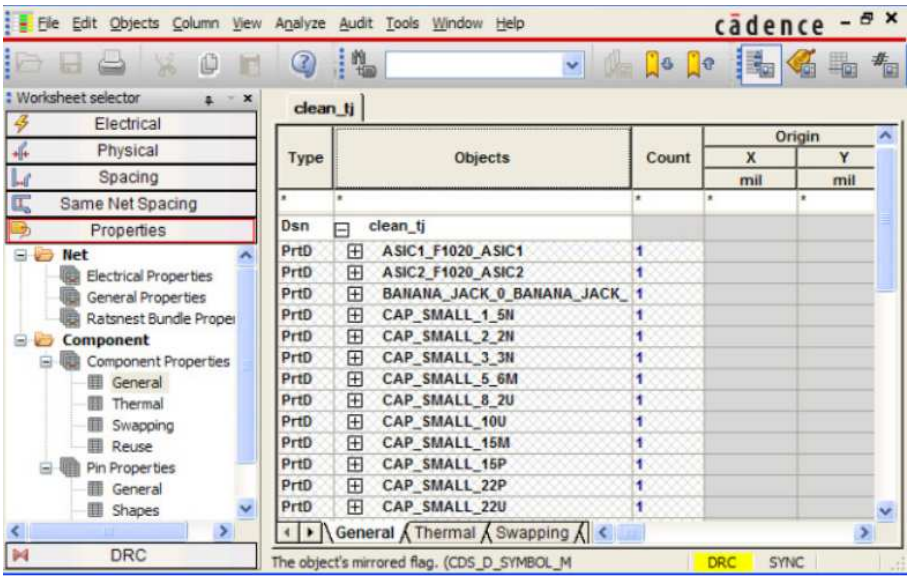


Figure 36 – (Constraint Manager window)

The constraint rules imported for K64 Board were defined in the file K64_AUDIO_0v1.dcf
It is important to keep all DRC active, so that way we can make sure we are following all the constraints rules.

4.6.1 Routing

Routing was performed following General Routing Requirements described in Section 10 of the requirements document K64_Audio_LDG.doc

During routing it is also recommended to activate DRCs for all views so it can help to avoid rework and waste of time fixing DRCs warnings or errors.

To verify the current status of DRCs go to Display->Status

It's also helpful set colors to the nets during routing and placement, it will make easier to connect all the components.

For assigning color to a net follow the next procedure:
Select Assign Color icon and on Options menu select the desired color, after that, select the net you want to assign the option color selected.

If net selection becomes difficult using mouse, you can optionally type in commands window. Net "Usb_Dm" highlighted in this case the desired net to be colored is Usb_Dm

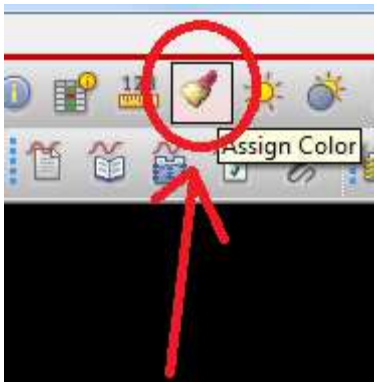


Figure 37 – (Assign color icon)

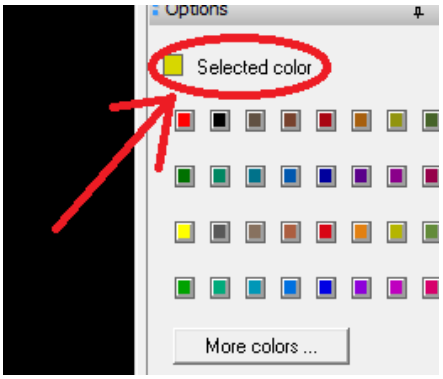


Figure 38 – (Selected color option)

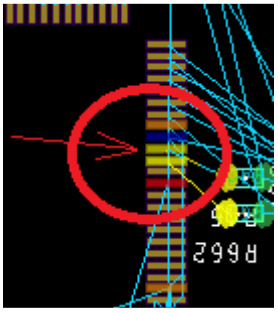


Figure 39 – (PAD colored)

4.7. Impedance

Requirements for impedance are described in K64_Audio_LDG.doc and some parameters can be defined in the constraint manager. Nevertheless for non controlled or unspecified impedances, the Saturn PCB design tool was really helpful to calculate impedances specifically for speakers 8 Ohm impedance. However, impedance calculated for 8 Ohm speaker was not possible due to space limitations and constraint max net width. A multiple of that impedance was set instead.

4.8. Planes/Shapes

To create a Ground plane, follow the next steps:

1- In the options menu, select the etch you want for a Ground plane.
2- Select the shape icon and select the area you want the shape.
3- Click on Select shape icon and select the shape you just created.
4- Verify your view is in the correct plane before set the GND plane.

5- Right click on the created shape and select assign plane. Make sure the options menu has selected GND plane.

K54_Audio board has a XTAL that will be isolated to avoid EMC failures and suggested for high speed layouts:

1- Click on create void icon
2- Select the area and correct etch you want the void
3- Click on polygon shape and surround the shape into the limit of the recent created void
4- Right click on polygon shape created and assign GND plane.

4.9. Manufacturing

Two manufacturer options were used, Oshpark and ITESO.
For Oshpark manufacturing specification details visit:
www.oshpark.com/guidelines
www.oshpark.com/design-tools/cadence-allegro
http://docs.oshpark.com/design-tools/cadence-allegro/
Gerber files required by Oshpark need to be packaged in a zip file containing the next file description and file extensions:
BOTTOM.GBL
K64AUDIO-1-4.XLN -> Drills
LAYER2.G2L
LAYER3.G3L
OUTLINE.GKO -> Outline
SMB.GBS -> Solder mask bottom
SMT.GTS -> Solder mask top
SSB.GBO -> Silkscreen bottom
SST.GTO -> Silkscreen top
TOP.GTL

To manufacture the board by using the ITESO tools, the board design must contain no more than 2 layers. Gerber files required are:
JEBA_BOTTOM.art
JEBA_NC_DRILL.drl -> Drills
JEBA_OUTLINE.art ->Outline
JEBA_TOP.art

It's strongly suggested not to add shapes, ground or power shapes, since the tool creates the net first and then creates the shapes or planes. This may cause shorts or nets deleted if shapes are too close to the net.

Images below show the generated gerbers files and schematics for the Audio amplifier and the K64_Audio board:

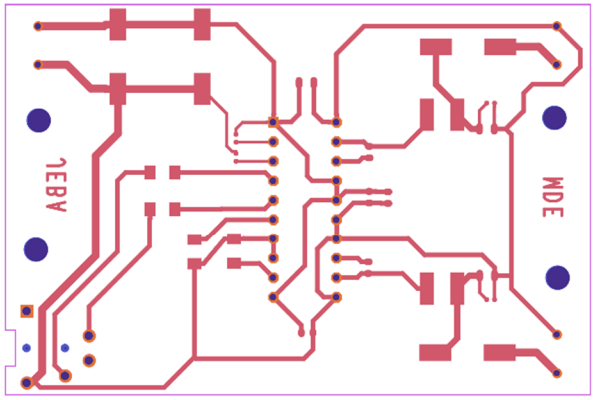


Figure 40 – (Audio amplifier Gerber files together)

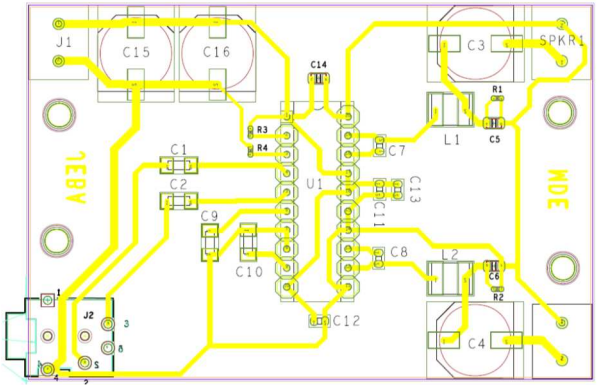


Figure 41 – (Silkscreen assembly)

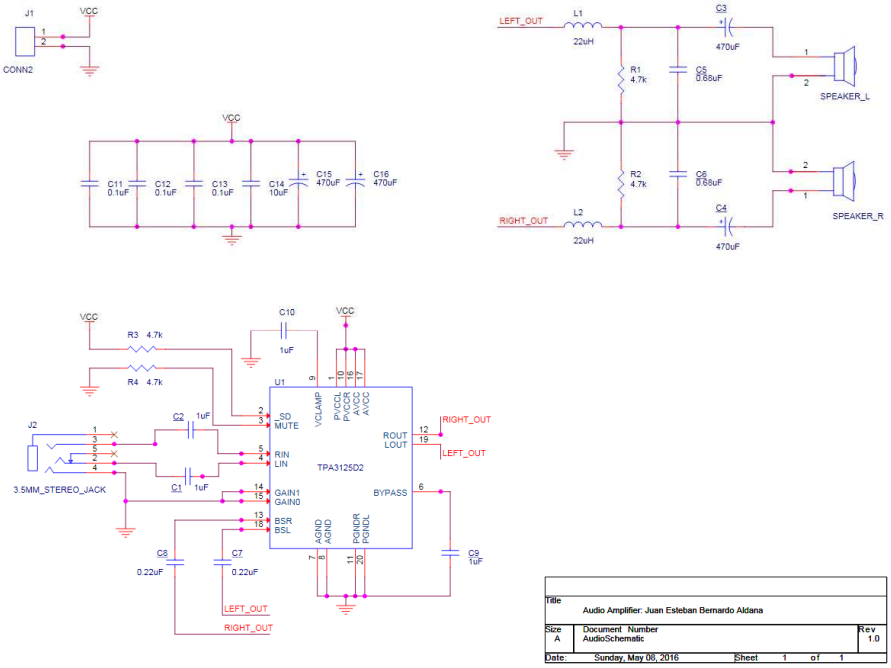


Figure 42 – (Audio amplifier schematic)

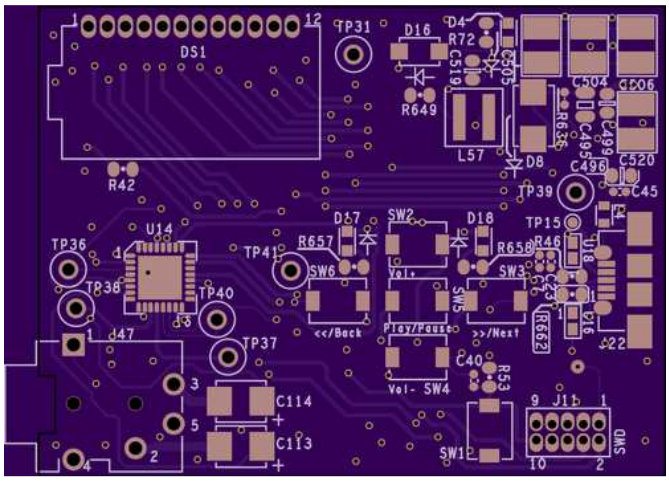


Figure 43 – (K63 Audio board top)

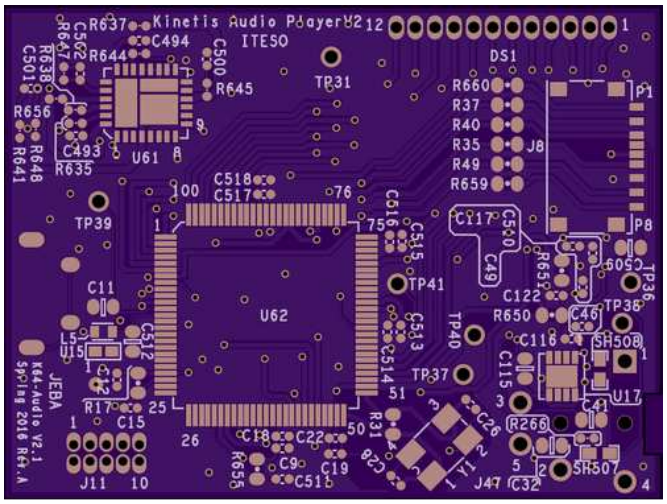


Figure 44 – (K64 Audio Bottom)

4.10. Tests

Only Audio amplifier was tested since K64 Audio board is stuck in customs office in Mexico City.

Results:
Input: 1Khz signal 520mVp-p
Output: 1Khz + 90° Phase signal 5.11Vp-p

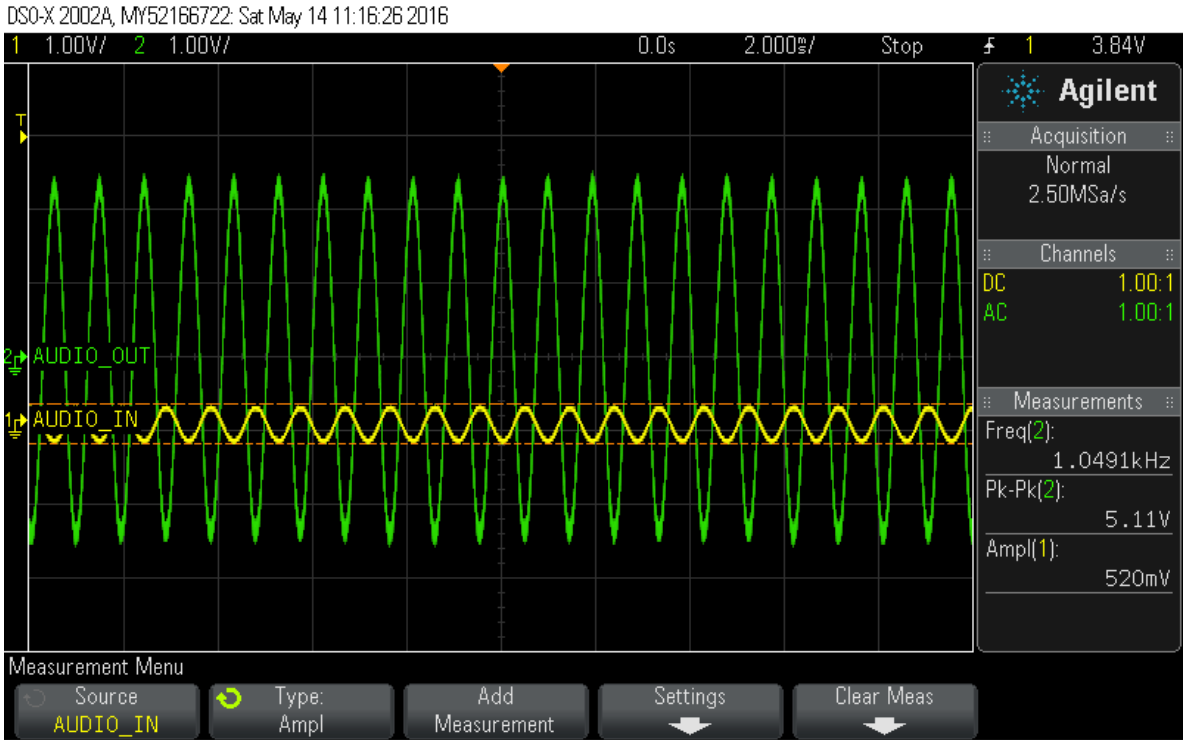


Figure 45 – (Audio amplifier test results)

4.11. Conclusions

As expected, now I know the basic elements to consider during a PCB development that makes me more confident at the moment of creating my own designs or projects and send them to manufacture.

4.12. Lessons learned

Do not select USPS priority Mail option from Oshpark.