Lesson 1

Task Structure

```
typedef struct{
    unsigned int Stack_Size;
    unsigned char priority;
    void (*p_TaskEntry)(void); //pointer to Tack C Function
    unsigned char AutoStart ;
    unsigned int _S_PSP_Task ;//Not Entered by the user
    unsigned int _E_PSP_Task ;//Not Entered by the user
    unsigned int* Current_PSP ;//Not Entered by the user
    char TaskName[30] ;
    enum{
        Suspend,
        Running,
        Waiting,
        ready
    }TaskState  ;//Not Entered by the user
struct{
        enum{
            Enable,
            Disable
        }Blocking;
        unsigned int Ticks_Count ;
    }TimingWaiting;
}Task_ref;
```

Scheduler.h

Global Functions

```
MYRTOS_errorID MYRTOS_init();
MYRTOS_errorID MYRTOS_CreateTask(Task_ref* Tref);
```

Error States

```
typedef enum{
    NoError,
    Ready_Queue_init_error,
    Task_exceeded_StackSize

}MYRTOS_errorID;
```

Scheduler.c

Os Structure

```
struct {
    Task_ref* OSTasks[100]; //Sch. Table
    unsigned int _S_MSP_Task ;
    unsigned int _E_MSP_Task ;
    unsigned int PSP_Task_Locator ;
    unsigned int NoOfActiveTasks ;
    Task_ref* CurrentTask ;
    Task_ref* NextTask ;
    enum{
        OSsuspend,
        OsRunning
    }OSmodeID;
}OS_Control;
```

تستخدم لتهيئة ***OS***
- تهئية ***MSP*** من خلال وظيفه ***MYRTOS_Create_MainStack***
- تحديد بداية أدرس ***PSP***
- إنشاء ***idle task*** وتهيئتها

```
MYRTOS_errorID MYRTOS_init()
{
    MYRTOS_errorID error = NoError ;

    //Update OS Mode (OSsuspend)
    OS_Control.OSmodeID = OSsuspend ;

    //Specify the MAIN Stack for OS
    MYRTOS_Create_MainStack();

    //Create OS Ready Queue
    if (FIFO_init(&Ready_QUEUE, Ready_QUEUE_FIFO, 100) !=FIFO_NO_ERROR)
    {
        error += Ready_Queue_init_error ;
    }

    //Configure IDLE TASK
    strcpy (MYRTOS_idleTask.TaskName, "idleTask");
    MYRTOS_idleTask.priority = 255 ;
    MYRTOS_idleTask.p_TaskEntry = MYRTOS_IdleTask ;
    MYRTOS_idleTask.Stack_Size = 300 ;

    error += MYRTOS_CreateTask(&MYRTOS_idleTask);

    return error ;

}
```

تستخدم هذه الوظيفه
- تحديد بداية ونهاية ***MSP***
- تحديد بدايه ***PSP***

```
void MYRTOS_Create_MainStack()
{
    OS_Control._S_MSP_Task = &_estack;
    OS_Control._E_MSP_Task = OS_Control._S_MSP_Task - MainStackSize;
    //Aligned 8 Bytes spaces between Main Task and PSP tasks
    OS_Control.PSP_Task_Locator = (OS_Control._E_MSP_Task - 8);

    //if (_E_MSP_Task <&_eheap) Error:excedded the availble stack size
}
```

تستخدم هذه الوظيفه
- لتحديد بدايه ونهايه ***PSP*** لل ***Task***
- تهيئة ال***stack*** الخاص بال***Task***
- إدراج ال***Task*** في ***Os_Task_Table***
- وضع ال***Task*** في حاله ***Suspend***

```
MYRTOS_errorID MYRTOS_CreateTask(Task_ref* Tref)
{
    MYRTOS_errorID error = NoError ;

    //Create Its OWN PSP stack
    //Check task stack size exceeded the PSP stack
    Tref->_S_PSP_Task = OS_Control.PSP_Task_Locator;
    Tref->_E_PSP_Task = Tref->_S_PSP_Task - Tref->Stack_Size ;

    if(Tref->_E_PSP_Task < (unsigned int)(&(_eheap)))
    {
        return Task_exceeded_StackSize ;
    }

    //Aligned 8 Bytes spaces between Task PSP and other
    OS_Control.PSP_Task_Locator = (Tref->_E_PSP_Task - 8);

    //Initialize PSP Task Stack
    MyRTOS_Create_TaskStack( Tref);

    //update sch Table
    OS_Control.OSTasks[OS_Control.NoOfActiveTasks]= Tref ;
    OS_Control.NoOfActiveTasks++ ;

    //Task State Update -> Suspend
    Tref->TaskState = Suspend ;

    return error ;
}
```

تستخدم هذه الوظيفه
- لتهيئة ال***stack*** الخاص بالوظيفه

```
void MyRTOS_Create_TaskStack(Task_ref* Tref)
{
    /*Task Frame
     * ======
     * XPSR
     * PC (Next Task Instruction which should be Run)
     * LR (return register which is saved in CPU while TASk1 running before TaskSwitching)
     * r12
     * r4
     * r3
     * r2
     * r1
     * r0
     *====
     *r5, r6 , r7 ,r8 ,r9, r10,r11 (Saved/Restore)Manual
     */
    Tref->Current_PSP = Tref->_S_PSP_Task ;

    Tref->Current_PSP-- ;
    *(Tref->Current_PSP) = 0x01000000;          //DUMMY_XPSR should T =1 to avoid BUS fault;//0x01000000

    Tref->Current_PSP-- ;
    *(Tref->Current_PSP) = (unsigned int)Tref->p_TaskEntry ; //PC

    Tref->Current_PSP-- ; //LR = 0xFFFFFFFD (EXC_RETURN)Return to thread with PSP
    *(Tref->Current_PSP)  = 0xFFFFFFFD ;

    for (int  j=0 ; j< 13 ; j++ )
    {
        Tref->Current_PSP-- ;
        *(Tref->Current_PSP)  = 0 ;

    }
```

إنشاء ***Fifo buffer*** لل***Task*** التي في وضع ***Ready***

```
FIFO_Buf_t Ready_QUEUE ;
```

إنشاء ***Table*** بعدد 100 لل***Tasks***

```
Task_ref* Ready_QUEUE_FIFO[100] ;
```

إنشاء ***Idle_Task***

```
Task_ref MYRTOS_idleTask ;
```

Lesson 1

```c
int main(void)
{

    MYRTOS_errorID error;
    //HW init (Initialize ClockTree, RestController)
    HW_init();

    if (MYRTOS_init() != NoError)
        while (1);

    Task1.Stack_Size = 1024 ;
    Task1.p_TaskEntry =task1;
    Task1.priority= 3 ;
    strcpy (Task1.TaskName, "task_1");

    error += MYRTOS_CreateTask(&Task1);

    MYRTOS_ActivateTask(&Task1);

    MYRTOS_STARTOS() ;

    while (1)
    {
    }
}
```

```c
MYRTOS_errorID MYRTOS_init()
{
    MYRTOS_errorID error = NoError ;

    //Update OS Mode (OSsuspend)
    OS_Control.OSmodeID = OSsuspend ;

    //Specify the MAIN Stack for OS
    MYRTOS_Create_MainStack();

    //Create OS Ready Queue
    if (FIFO_init(&Ready_QUEUE, Ready_QUEUE_FIFO, 100) !=FIFO_NO_ERROR)
    {
        error += Ready_Queue_init_error ;
    }

    //Configure IDLE TASK
    strcpy (MYRTOS_IdleTask.TaskName, "IdleTask");
    MYRTOS_IdleTask.priority = 255 ;
    MYRTOS_IdleTask.p_TaskEntry = MYRTOS_IdleTask ;
    MYRTOS_IdleTask.Stack_Size = 300 ;

    error += MYRTOS_CreateTask(&MYRTOS_IdleTask);

    return error ;
}
```

```c
extern int _estack ;
#define MainStackSize   3072

void MYRTOS_Create_MainStack()
{
    OS_Control._S_MSP_Task = &_estack;
    OS_Control._E_MSP_Task = OS_Control._S_MSP_Task - MainStackSize;
    //Aligned 8 Bytes-spaces between Main Task and PSP tasks
    OS_Control.PSP_Task_Locator = (OS_Control._E_MSP_Task - 8);

    //if (_E_MSP_Task <&_eheap) Error:exceeded the availble stack size
}
```

```c
Buffer_status FIFO_init (FIFO_Buf_t* fifo,element_type* buff , unsigned int length){

    if(!buff )
        return FIFO_NULL;

    fifo->base = buff ;
    fifo->head = fifo->base ;
    fifo->tail = fifo->base ;
    fifo->length = length;
    fifo->counter=0;

    return FIFO_NO_ERROR;

}
```

```c
MYRTOS_errorID MYRTOS_CreateTask(Task_ref* Tref)
{
    MYRTOS_errorID error = NoError ;

    //Create Its OWN PSP stack
    //Check task stack size exceeded the PSP stack
    Tref->_S_PSP_Task = OS_Control.PSP_Task_Locator;
    Tref->_E_PSP_Task = Tref->_S_PSP_Task - Tref->Stack_Size ;

    if(Tref->_E_PSP_Task < (unsigned int)(&(_eheap)))
    {
        return Task_exceeded_StacksSize ;
    }

    //Aligned 8 Bytes spaces between Task PSP and other
    OS_Control.PSP_Task_Locator = (Tref->_E_PSP_Task - 8);

    //Initialize PSP Task Stack
    MYRTOS_Create_TaskStack( Tref);

    //update sch Table
    OS_Control.OSTasks[OS_Control.NoOfActiveTasks]= Tref ;
    OS_Control.NoOfActiveTasks++ ;

    //Task State Update -> Suspend
    Tref->TaskState = Suspend ;

    return error ;
}
```

```c
void MYRTOS_Create_TaskStack(Task_ref* Tref)
{
    /*Task Frame
     * =====
     * XPSR
     * PC (Next Task Instruction which should be Run)
     * LR (return register which is saved in CPU while TASk1 running before TaskSwitching)
     * r12
     * r4
     * r3
     * r2
     * r1
     * r0
     *=====
     *r5, r6 , r7 ,r8 ,r9, r10,r11 (Saved/Restore)Manual
     */
    Tref->Current_PSP = Tref->_S_PSP_Task ;

    Tref->Current_PSP-- ;
    *(Tref->Current_PSP) = 0x01000000;          //DUMMY_XPSR should T =1 to avoid BUS fault;//0x01000000

    Tref->Current_PSP-- ;
    *(Tref->Current_PSP) = (unsigned int)Tref->p_TaskEntry ; //PC

    Tref->Current_PSP-- ; //LR = 0xFFFFFFFD (EXC_RETURN)Return to thread with PSP
    *(Tref->Current_PSP)  = 0xFFFFFFFD ;

    for (int  j=0 ; j< 13 ; j++ )
    {
        Tref->Current_PSP-- ;
        *(Tref->Current_PSP)  = 0 ;
    }
}
```

```c
void HW_init()
{
    //Initialize Clock Tree (RCC -> SysTick Timer & CPU) 8MHZ
    //init HW
    //Clock tree
    //RCC Default Values makes CPU Clock & SysTick Timer clock = 8 M HZ
    //  decrease PenSV  interrupt priority to be  smaller than or equal  SysTICK Timer
    //SysTICK have a priority 14
    __NVIC_SetPriority(PendSV_IRQn, 15) ;


}
```