My RTOS

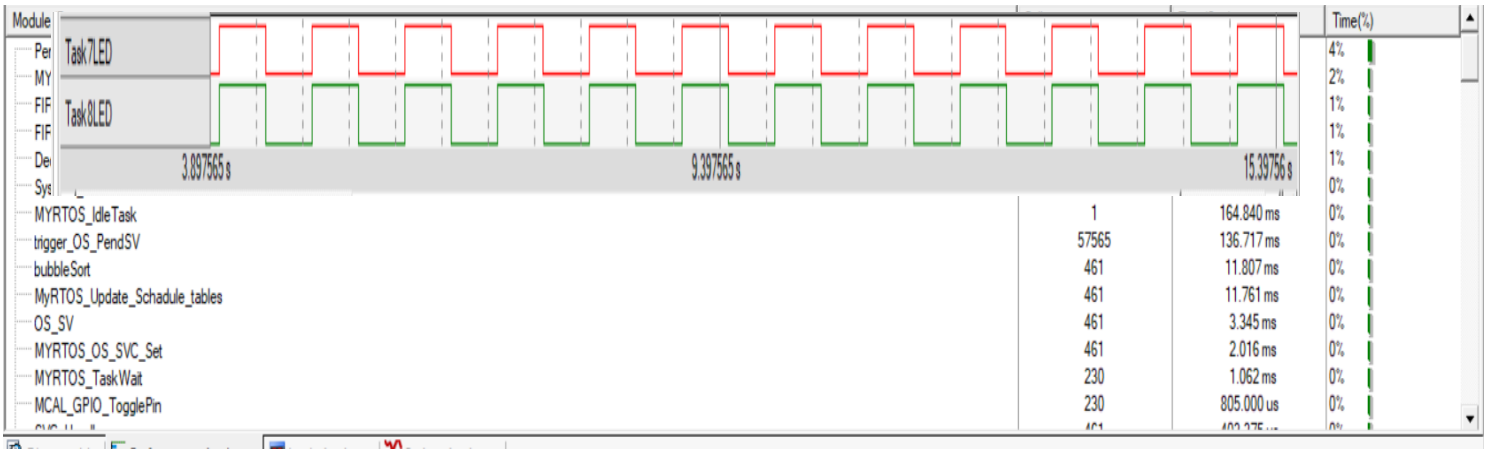
function and example

1:MYRTOS_CreateTask

2:MYRTOS_TaskWait

```
Task_Ref Task7,Task8;

void task7()
{
	while (1)
	{
		MCAL_GPIO_TogglePin(GPIOA, GPIO_PIN_6);
		Task7LED ^= 1;
		MYRTOS_TaskWait(500,&Task7);
	}
}

void task8()
{
	while (1)
	{
		MCAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7);
		Task8LED ^= 1;
		MYRTOS_TaskWait(500,&Task8);
	}
}

Task7.P_TaskEntry=task7;
Task7.Stack_Size=512;
Task7.Auto_Start=Task_Start;
strcpy(Task7.TaskName,"task7");
Task7.priority=3;
Task8.P_TaskEntry=task8;
Task8.Stack_Size=512;
Task8.Auto_Start=Task_Start;
strcpy(Task8.TaskName,"task8");
Task8.priority=3;

Error=MYRTOS_CreateTask(&Task7);
```

```
Error=MYRTOS_CreateTask(&Task8);
```



Task 7 and Task 8 have the same priority, they work in a round-robin range, and a delay of 500 ms is made, the OS consumes only 4% of the system

---

Synchronization between Task1 and Task 2 using Mutex

```
Mutex_Ref MUT1;
int flag=0;
MUT1.PayloadSize=1;
MUT1.Ppayload=&flag;
strcpy(MUT1.MutexName,"flag");
void task1()
{
	int pr_flag=0;
	while (1)
	{
		MYRTOS_AcquireMutex(&MUT1,&Task1);
		flag=MCAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0);
		if(flag != pr_flag)
		{
			MYRTOS_ReleaseMutex(&MUT1);
		}
		Task1LED ^= 1;
```

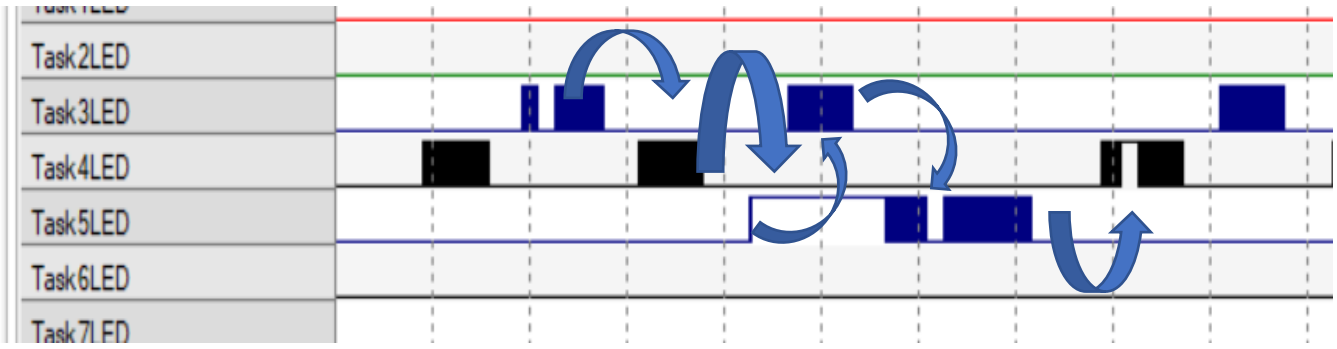
```
			pr_flag=flag;
			MYRTOS_TaskWait(500,&Task1);
		}
}
void task2()
{
		while (1)
		{
			MYRTOS_AcquireMutex(&MUT1,&Task2);
			MCAL_GPIO_TogglePin(GPIOA, GPIO_PIN_1);
			Task2LED ^= 1;
			MYRTOS_ReleaseMutex(&MUT1);
		}
}
```

Solution priority inversion used ceiling protocol

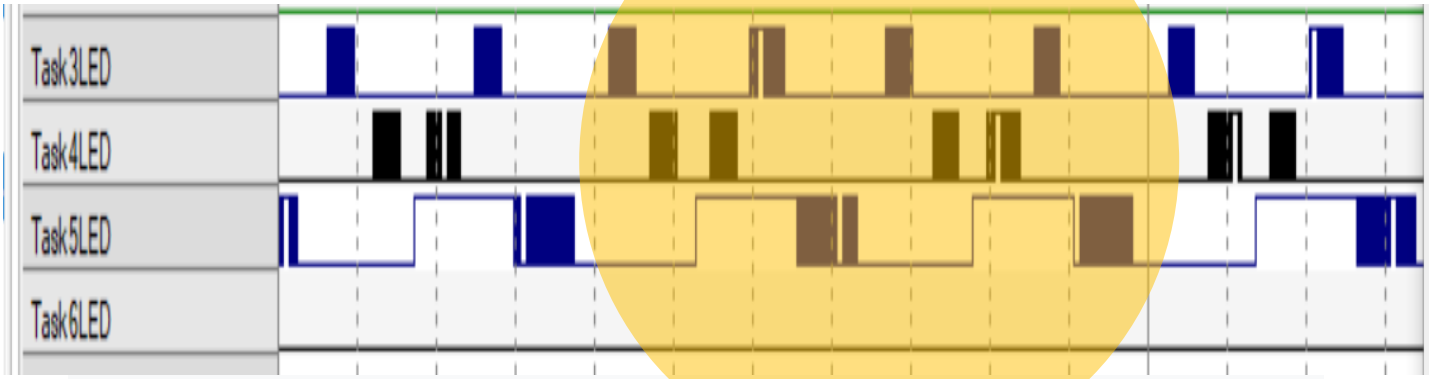
```
void task3()
{
	int count = 0;
	while(1)
	{
		Task3LED ^= 1;
		count++;
		if(count == 100)
		{
			MYRTOS_AcquireMutex(&MUT1,&Task3);
			MYRTOS_ActivateTask(&Task4);
		}
		if(count == 200)
		{
			count = 0;
			MYRTOS_ReleaseMutex(&MUT1);
		}
	}
}

void task4()
{
	int count = 0;
```

```
	while(1)
	{
		Task4LED ^= 1;
		count++;
		if(count == 100)
		{
			MYRTOS_ActivateTask(&Task5);
		}
		if(count == 200)
		{
			count = 0;
			MYRTOS_TerminateTask(&Task4);
		}
	}
}

void task5()
{
	int count = 0;
	while(1){
		Task5LED ^= 1;
		count++;
		if(count == 3)
		{
			MYRTOS_AcquireMutex(&MUT1,&Task5);
		}
		if(count == 200)
		{
			count = 0;
			MYRTOS_ReleaseMutex(&MUT1);
			MYRTOS_TerminateTask(&Task5);
		}
	}
}
```

after



before



Synchronization between Task1 and Task 2 using Semaphore

```
Semaphore_Ref Sem1;
Sem1.Ppayload=&flag;
strcpy(Sem1.SemaphoreName,"flag");
int flag=0;
void task6()
{
	int pr_flag=0;
	while (1)
	{
		MYRTOS_AcquireSemaphore(&Sem1, &Task6);
		flag=MCAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0);
		if(flag != pr_flag)
		{
			MYRTOS_ReleaseSemaphore(&Sem1);
		}
		Task6LED ^= 1;
		pr_flag=flag;
		MYRTOS_TaskWait(500,&Task6);
	}
}

void task7()
{
	while (1)
	{
		MYRTOS_AcquireSemaphore(&Sem1, &Task7);
		MCAL_GPIO_TogglePin(GPIOA, GPIO_PIN_1);
```

```
            Task7LED ^= 1;
            MYRTOS_ReleaseSemaphore(&Sem1);
        }

}
```