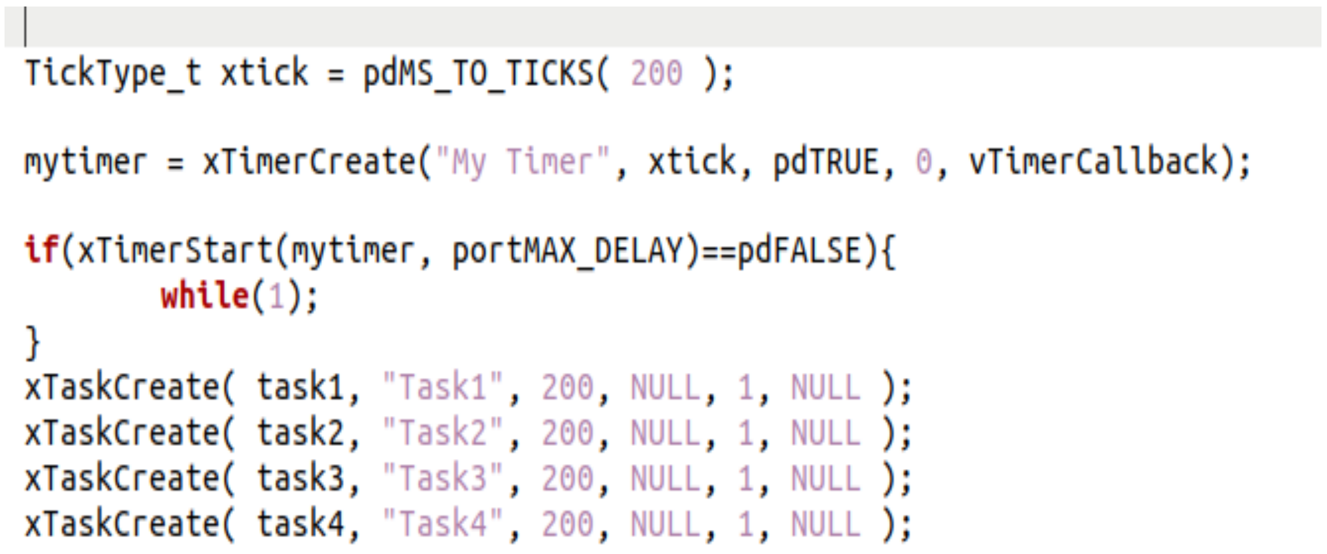
Abhishek Kumar Simra
PRN:-220940130002

Program to set Sofrware timer

This program objective is to create a software timer that will make the timer callback function and schedule the task.

SOLUTION:

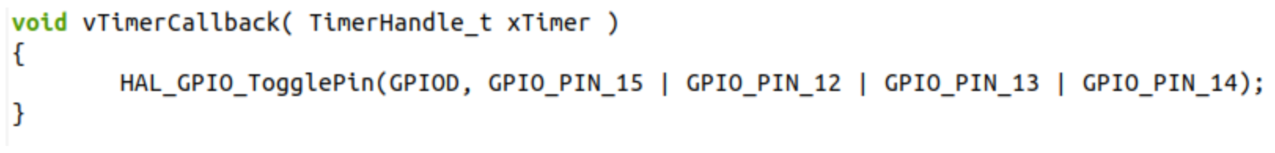


```
TickType_t xtick = pdMS_TO_TICKS( 200 );

mytimer = xTimerCreate("My Timer", xtick, pdTRUE, 0, vTimerCallback);

if(xTimerStart(mytimer, portMAX_DELAY)==pdFALSE){
        while(1);
}
xTaskCreate( task1, "Task1", 200, NULL, 1, NULL );
xTaskCreate( task2, "Task2", 200, NULL, 1, NULL );
xTaskCreate( task3, "Task3", 200, NULL, 1, NULL );
xTaskCreate( task4, "Task4", 200, NULL, 1, NULL );
```

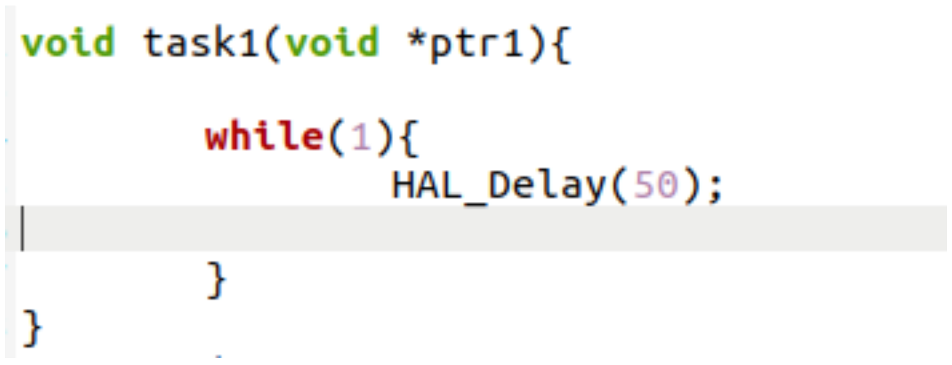
Here we have created the software timer, the software timer takes the pasrameters as timer name as first parameter,the period of timer in form of ticks, then to restart or auto-reload timer, then the pid of timer is entered which is a void pointer and can be used by other task to use the timers, then at last the callback function is declared.



```
void vTimerCallback( TimerHandle_t xTimer )
{
        HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_15 | GPIO_PIN_12 | GPIO_PIN_13 | GPIO_PIN_14);
}
```

Here is the timer callback function which calls the timer and toggle leds to show every tick of the timer

Task1



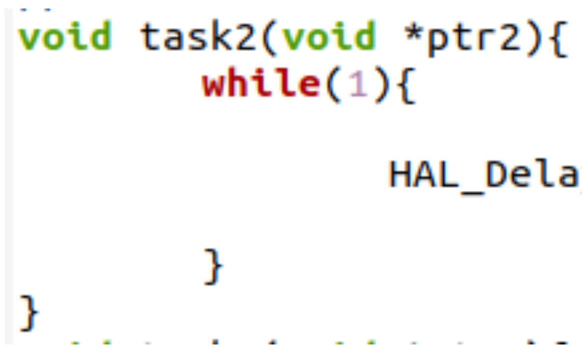
```
void task1(void *ptr1){

        while(1){
                HAL_Delay(50);

        }
}
```

the task1 is a simple function with a HAL_Delay in the while loop which is using the software timer as its clock

Task2



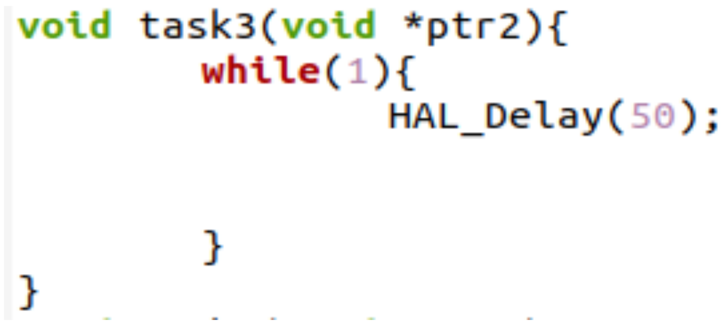
```
void task2(void *ptr2){
        while(1){

                HAL_Delay(50);

        }
}
```

the task2 is also a simple function same as task1 with a HAL_Delay in the while loop which is using the software timer as its clock

Task3

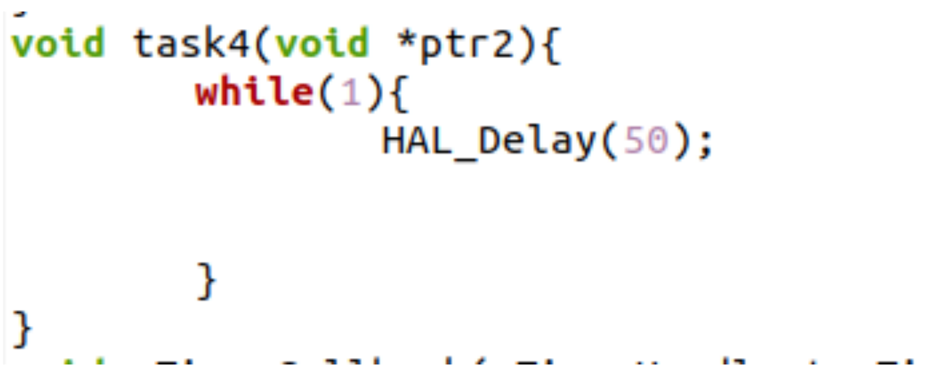


```
void task3(void *ptr2){
        while(1){
                HAL_Delay(50);


        }
}
```

the task3 is also a simple function same as task1 and task2 with a HAL_Delay in the while loop which is using the software timer as its clock

Task4

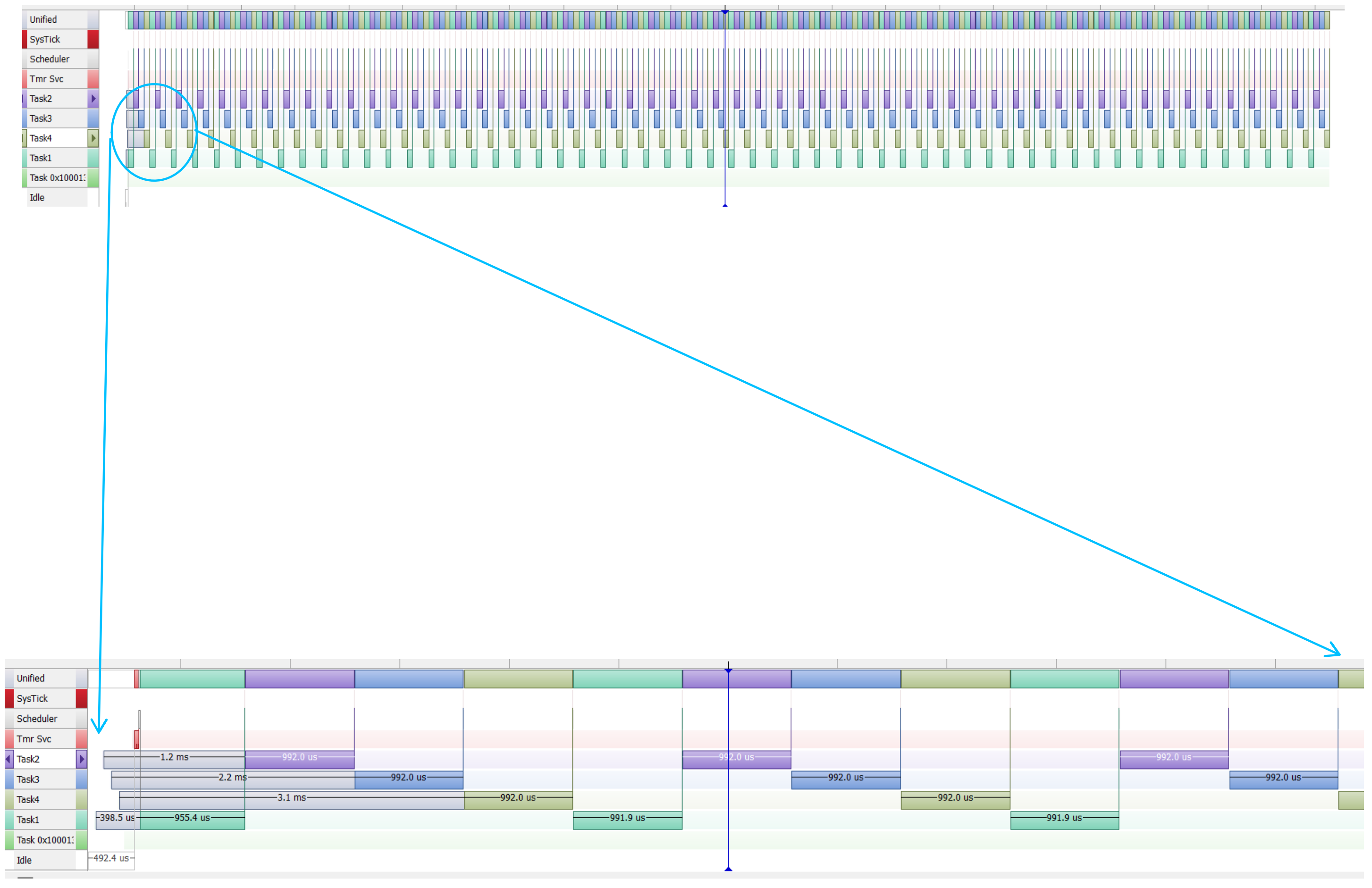


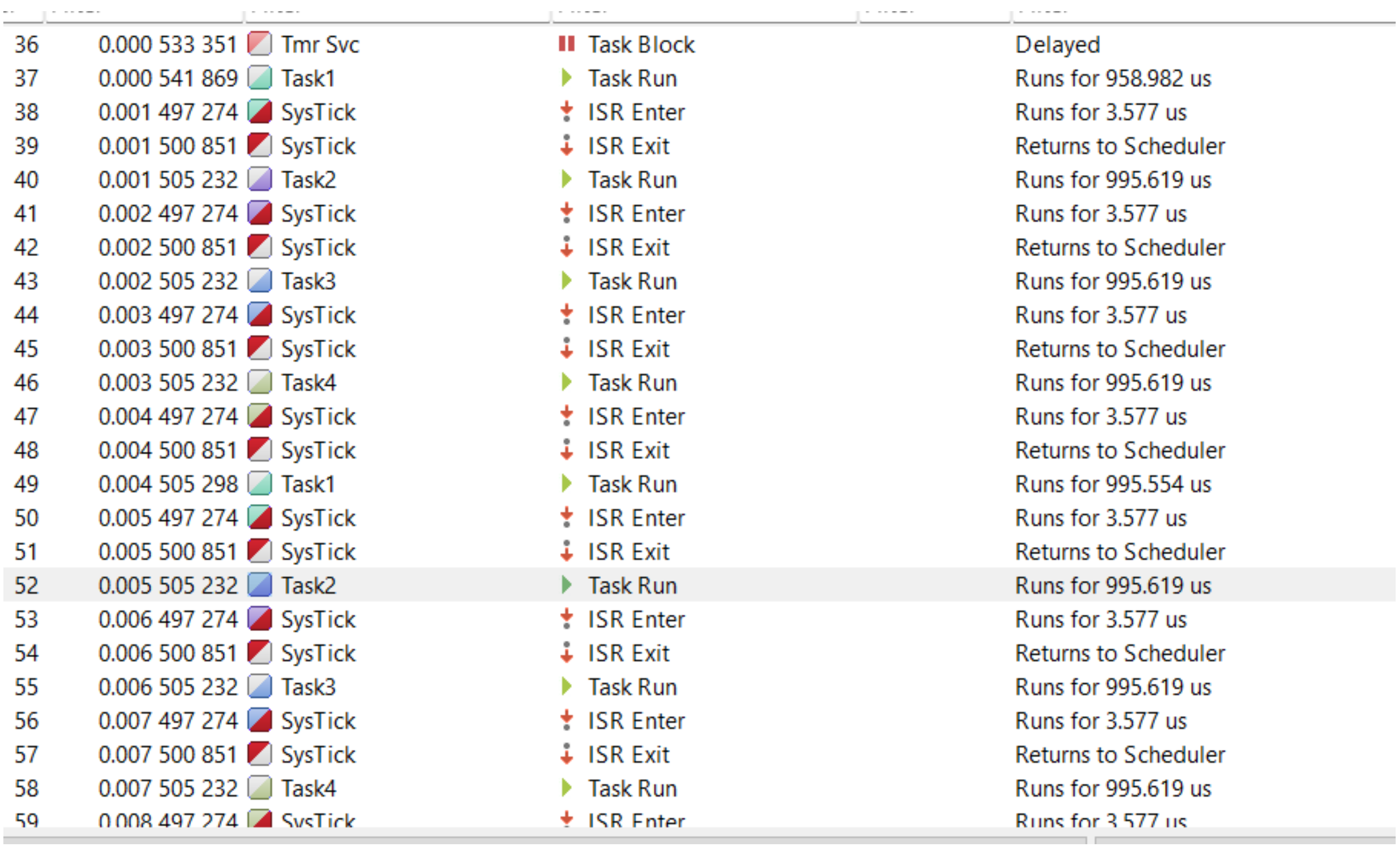
```
void task4(void *ptr2){
        while(1){
                HAL_Delay(50);


        }
}
```

the task4 is also a simple function same as task1 and task2 with a HAL_Delay in the while loop which is using the software timer as its clock

TRACE





36	0.000 533 351	Tmr Svc	Task Block	Delayed
37	0.000 541 869	Task1	Task Run	Runs for 958.982 us
38	0.001 497 274	SysTick	ISR Enter	Runs for 3.577 us
39	0.001 500 851	SysTick	ISR Exit	Returns to Scheduler
40	0.001 505 232	Task2	Task Run	Runs for 995.619 us
41	0.002 497 274	SysTick	ISR Enter	Runs for 3.577 us
42	0.002 500 851	SysTick	ISR Exit	Returns to Scheduler
43	0.002 505 232	Task3	Task Run	Runs for 995.619 us
44	0.003 497 274	SysTick	ISR Enter	Runs for 3.577 us
45	0.003 500 851	SysTick	ISR Exit	Returns to Scheduler
46	0.003 505 232	Task4	Task Run	Runs for 995.619 us
47	0.004 497 274	SysTick	ISR Enter	Runs for 3.577 us
48	0.004 500 851	SysTick	ISR Exit	Returns to Scheduler
49	0.004 505 298	Task1	Task Run	Runs for 995.554 us
50	0.005 497 274	SysTick	ISR Enter	Runs for 3.577 us
51	0.005 500 851	SysTick	ISR Exit	Returns to Scheduler
52	0.005 505 232	Task2	Task Run	Runs for 995.619 us
53	0.006 497 274	SysTick	ISR Enter	Runs for 3.577 us
54	0.006 500 851	SysTick	ISR Exit	Returns to Scheduler
55	0.006 505 232	Task3	Task Run	Runs for 995.619 us
56	0.007 497 274	SysTick	ISR Enter	Runs for 3.577 us
57	0.007 500 851	SysTick	ISR Exit	Returns to Scheduler
58	0.007 505 232	Task4	Task Run	Runs for 995.619 us
59	0.008 497 274	SysTick	ISR Enter	Runs for 3.577 us

the trace shows how all the tasks where using the software timer to switch controller between them they the using the systick as the software timer because the systicks where acting as the interrupts for the task to schedule between them as the tasks where having a HAL_Delay API in while loop and the Hal_delay function let the task keep the priority but when systick comes the controll is shifted to the next highest priority task, and like this the switching between tasks there taking place.