VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY

HO CHI MINH CITY UNIVERSITY OF TECHNOLY

REPORT

LAB 4

Class: Microprocessors-Microcontrollers – CC01

Lecture: NGUYỄN THIÊN ÂN

No.	Full Name	ID Student
1.	Nguyễn Huy Tài	2110513

Ho Chi Minh City, November 12th 2024

Contents

1 Exercise 1: Initial scheduler

```
int main(void)

    /* structure of main flow */
    SCH_Init();
    SCH_Add_Task(turnREDLED, 10, 500, -1);
    SCH_Add_Task(turnGREENLED, 10, 200, -1);
    SCH_Add_Task(turnYELLOWLED, 10, 300, -1);

    SCH_Add_Task(turnRED, 10, 600, -1);
    SCH_Add_Task(turnGreen, 10, 700, -1);
    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
      SCH_Dispatch_Tasks();
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    }
    /* USER CODE END 3 */
}

/* USER CODE BEGIN 4 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    SCH_Update();
}
/* USER CODE END 4 */
```

Program 1: Structure of main flow

2 Exercise 2: SCH_Update function

```
void SCH_Update(void)
{
    if (head -> pTask != NULL)
    {
        if (head->Delay == 0)
        {
            head->RunMe +=1;
            if (head->Period)
            {
                head->Delay = head->Period;
            } else {
                SCH_Delete_Task(head->TaskID);
            }
        } else {
            head->Delay -= 1;
        }
    }
}
```

Program 2: Function of update sched

3 Exercise 3: Dispatch scheduler

```
void SCH_Dispatch_Tasks(void)
{
    sTask *temp = head;
    while (temp != NULL)
    {
        if (temp->RunMe > 0)
        {
            temp->RunMe = 0;
            temp->pTask();
            if (temp->Period == 0)
            {
                SCH_Delete_Task(temp->TaskID);
            }

            else
            {
                sTask *temp2 = (sTask *)malloc(sizeof(sTask
    ));
                deep_copy(temp2, temp);

                SCH_Delete_Task(temp->TaskID);
                SCH_Add_Task(temp2->pTask, temp2->Delay,
    temp2->Period, temp2->TaskID);
```

```
            }

//              break;
        }

        temp = temp->next;
    }
}
```

Program 3: Function of dispatch scheduler

4 Exercise 4: Add scheduler

```
unsigned char SCH_Add_Task(void (*pFunction)(), unsigned int DELAY, unsigned int PERIOD, int TaskID) {
    // add new task
    sTask *new_task = (sTask *)malloc(sizeof(sTask));
    if (count > SCH_MAX_TASKS) {
        Error_code_G = ERROR_SCH_TOO_MANY_TASKS;
        return SCH_MAX_TASKS;
    }

    if (id > SCH_MAX_TASKS) {
        id = 0;  // reset TaskID
    }

    new_task->pTask = pFunction;
    new_task->Delay = DELAY;
    new_task->Period = PERIOD;
    new_task->RunMe = 0;
    if (TaskID < 0)
    {
        new_task->TaskID = id++;
    } else {
        new_task->TaskID = TaskID;
    }
    new_task->next = NULL;
    new_task->prev = NULL;

    if (head == NULL) {
        head = new_task;
        tail = new_task;
        count++;
        return new_task->TaskID;
    }

    sTask *temp = head;
    while (temp != NULL && temp->Delay <= DELAY) {
        DELAY -= temp->Delay;
```

```
        temp = temp->next;
    }

    new_task->Delay = DELAY;
    if (temp != NULL) {
        new_task->next = temp;
        new_task->prev = temp->prev;

        if (temp->prev != NULL) {
            temp->prev->next = new_task;
        } else {
            head = new_task;
        }
        temp->prev = new_task;
        temp->Delay -= DELAY;

    } else {
        tail->next = new_task;
        new_task->prev = tail;
        tail = new_task;
    }

    count++;
    return new_task->TaskID;
}
```

Program 4: Add task function

5 Exercise 5: Delete scheduler

```
void SCH_Delete_Task(unsigned char TaskID)
{
    sTask *temp = head;

    while (temp != NULL && temp->TaskID != TaskID)
    {
        temp = temp->next;
    }

    if (temp == NULL)
    {
        Error_code_G = ERROR_SCH_CANNOT_DELETE_TASK;
        return;
    }

    if (temp->prev != NULL)
    {
        temp->prev->next = temp->next;
    }
```

```
    else
    {
        head = temp->next;
    }

    if (temp->next != NULL)
    {
        temp->next->prev = temp->prev;
    }
    else
    {
        tail = temp->prev;
    }

    free(temp);
    count--;
    return;
}
```

Program 5: Delete task functin

6 Link github

You can find the source code on my GitHub repository: **My GitHub Repository**.