VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
Faculty of Computer Science and Engineering

CC02 — Lab Report

Microprocessor - Microcontroller
Lab 4

Supervisors: Nguyen Thien An
Students: Vu Trinh Thanh Binh 2252085

Ho Chi Minh City, November 25, 2024

Contents

1 Exercise

The GitHub link for the lab schematics is at here or in this link: https://github.com/thanhbinh0710/VXL.git.

1.1 Proteus Schematic

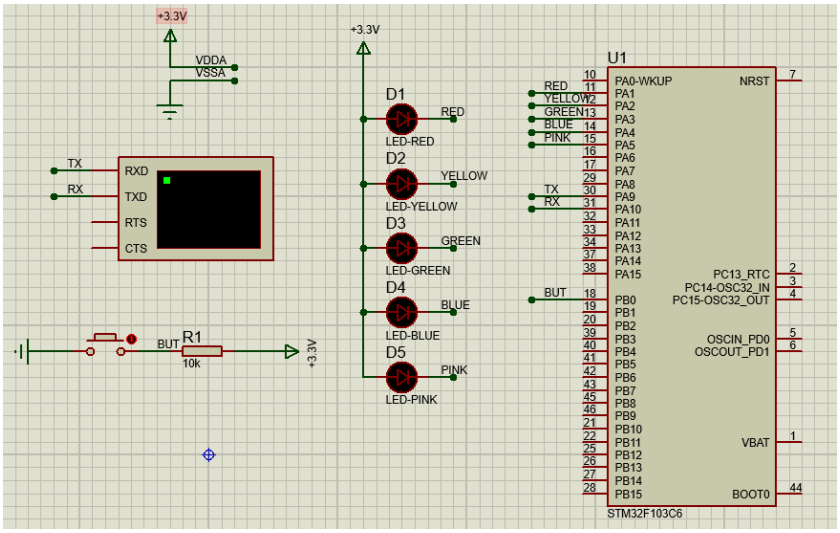

Figure 1: *Schematic*

1.2 Scheduler Initialization

```
Task_List Tlist;
uint32_t Tcounter = 0; //Task_counter
ERROR_CODE errorCode = NO_ERROR;
ERROR_CODE lastError = NO_ERROR;

char str[50];

void SCH_INIT(){
  Tlist.head = NULL;
  Tlist.size = 0;
  Tcounter = 0;
  errorCode = NO_ERROR;
}
```

1.3 Scheduler Update Function

```
void SCH_Update(){
  timestamp += 10;
  if (Tlist.size && Tlist.head->Delay > 0){
    Tlist.head->Delay--;
  }
}
```

1.4 Scheduler Delete Function

```
int deleteTask (uint32_t ID){
  if (Tlist.size == 0) return 1;
  Task_Control *curr = Tlist.head;
  Task_Control *prev = NULL;
  while (curr != NULL){
    if (ID == curr->TaskID){
      //found task
      if (prev == NULL){
        //delete 1st task in list
        Tlist.head = curr->nextTask;
      }else{
        prev->nextTask = curr->nextTask;
      }
      if (curr->nextTask != NULL){
        curr->nextTask->Delay += curr->Delay;
      }
      free(curr);
      Tlist.size--;
      return 0;
    }
    prev = curr;
    curr = curr->nextTask;
  }
  return 1;
}

RETURN_CODE SCH_Delete_Task(const unsigned char TaskID){
  RETURN_CODE returnCode;
  if (deleteTask(TaskID)){
    returnCode = ERROR_SCH_CANNOT_DELETE_TASK;
  }else{
    returnCode = NO_ERROR;
  }
  return returnCode;
}
```

1.5 Scheduler Add Task Function

```
int addTask(Task_Control *task){
 if (Tlist.size >= SCH_MAX_TASKS){
   return 1;
 }
 if (Tlist.size == 0){
   Tlist.head = task;
   Tlist.size++;
   return 0;
 }
 Task_Control *curr = Tlist.head;
 Task_Control *prev = NULL;
 while (curr != NULL && task->Delay >= curr->Delay){
   task->Delay -= curr->Delay;
   prev = curr;
   curr = curr->nextTask;
 }
 task->nextTask = curr;
 if (prev != NULL){
   prev->nextTask = task;
 }else{
   Tlist.head = task;
 }
 if (curr != NULL){
   curr->Delay -= task->Delay;
 }
 Tlist.size++;
 return 0;
}

unsigned char SCH_Add_Task(void (*func_ptr)(), unsigned int DELAY,
    unsigned int PERIOD){
 Task_Control *task = (Task_Control*)malloc(sizeof(Task_Control));
 task->Delay = DELAY;
 task->Period = PERIOD;
 task->TaskID = (++Tcounter)%256;
 task->Task_ptr = func_ptr;
 task->nextTask = NULL;
 addTask(task);
 return task->TaskID;
}
```

1.6 Scheduler Dispatcher Funtion

```
void SCH_Dispatcher(){
  while (Tlist.size != 0 && Tlist.head -> Delay == 0){
    uint32_t timeOut = timestamp;
    sprintf(str, "TaskID: %ld timeout at: %ld ms\r\n", Tlist.head->TaskID,timeOut)
    ;
    displayMSG(str);
    (*Tlist.head->Task_ptr)();
    if (Tlist.head->Period) SCH_Add_Task(Tlist.head->Task_ptr, Tlist.head->Period,
    Tlist.head->Period);
    SCH_Delete_Task(Tlist.head->TaskID);
  }
  SCH_Report_Status();
}
```

1.7 Scheduler Report Status Function

```
int errorCounter = 0;
void SCH_Report_Status(){
#ifdef REPORT_ERROR
  static int lastError = NO_ERROR;
  static int errorCounter = 0;
  if (errorCode != lastError){
    lastError = errorCode;
    if (errorCode != NO_ERROR){
      errorCounter = 60000;
      char errorMSG[100];
      snprintf(errorMSG, sizeof(errorMSG), "Error Code: %d\r\n", errorCode);
      displayMSG(errorMSG);
    }else{
      errorCounter =0;
      char noErrorMessage[] = "No Error\r\n";
      displayMSG(noErrorMessage);
    }
  }
  if (errorCounter > 0){
    if (--errorCounter == 0){
      errorCode = NO_ERROR;
    }
  }
#endif
}
```

1.8 Perform Led Blinking by using Scheduler

```
SCH_Add_Task(toggleRed,50,50);
SCH_Add_Task(toggleYellow,51,100);
SCH_Add_Task(toggleGreen,52,150);
SCH_Add_Task(toggleBlue,53,200);
SCH_Add_Task(togglePink,54,250);
while (1)
{
  SCH_Dispatcher();
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
SCH_Update();
  }
```

References