Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №4
“Интерфейс I2C и матричная клавиатура”

Дисциплина: Проектирование вычислительных систем
Вариант 4

Выполнили: студенты группы P34131

Бусыгин Дмитрий Алексеевич

Гиря Максим Дмитриевич

Преподаватель:

Пинкевич Василий Юрьевич

Санкт-Петербург
2024

Задание:

- Разработать программу, которая использует интерфейс I2C для считывания нажатий кнопок клавиатуры стенда SDK-1.1.
- Подсистема опроса клавиатуры должна удовлетворять следующим требованиям:
  – реализуется защита от дребезга;
  – нажатие кнопки фиксируется сразу после того, как было обнаружено, что кнопка нажата (с учетом защиты от дребезга), а не в момент отпускания кнопки; если необходимо, долгое нажатие может фиксироваться отдельно;
  – кнопка, которая удерживается дольше, чем один цикл опроса, не считается повторно нажатой до тех пор, пока не будет отпущена (нет переповторов);
  – распознается и корректно обрабатывается множественное нажатие (при нажатии более чем одной кнопки считается, что ни одна кнопка не нажата, если это не противоречит требованиям к программе);
  – всем кнопкам назначаются коды от 1 до 12 (порядок на усмотрение исполнителей). Программа должна иметь два режима работы, переключение между которыми производится по нажатию кнопки на боковой панели стенда:
  – режим тестирования клавиатуры;
  – прикладной режим. Уведомление о смене режима выводится в UART. В режиме тестирования клавиатуры программа выводит в UART коды нажатых кнопок. В прикладном режиме программа обрабатывает нажатия кнопок и выполняет действия в соответствии с вариантом задания.

Массивы состояний клавиш, массив мелодий и переменные:

```
// Define the button states
char buttonStates[4][3] = {
    {0, 0, 0},
    {0, 0, 0},
    {0, 0, 0},
    {0, 0, 0}
};
// Define the previous button states
char previousButtonStates[4][3] = {
    {1, 1, 1},
    {1, 1, 1},
    {1, 1, 1},
    {1, 1, 1}
};
// Define the melody array
int melodies[5][5] = {
    {600, 700, 600, 700, 600},
    {500, 600, 500, 600, 500},
    {500, 700, 500, 700, 500},
    {400, 800, 700, 600, 500},
    {0} // Dynamic melody
};
int melodyStartFlag = 0;
int currentMelodyIndex = 0;
int currentNoteIndex = 0;
int toggleState = 0;
int dynamicMelodyNoteCount = 0;
int debugMode = 0;
```

Функция обработчик колбека TIM6:

```
// Function to update PWM frequency
void UpdatePWMFrequency(int frequency) {
    htim1.Instance->PSC = frequency;
    htim1.Instance->CCR1 = 500; // Set duty cycle
    char buffer[50];
    sprintf(buffer, "\n\rPWM Frequency Set: %d\n", frequency);
    HAL_UART_Transmit(&huart6, (uint8_t *)buffer, strlen(buffer), 10);
}
// Timer interrupt callback
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
    if (htim->Instance == TIM6) {
        htim1.Instance->CCR1 = 0; // Stop PWM
        if (melodyStartFlag) {
            UpdatePWMFrequency(melodies[currentMelodyIndex][currentNoteIndex++]);
        }
        if (currentNoteIndex >= 5) {
            melodyStartFlag = 0; // Stop melody
            currentNoteIndex = 0;
            HAL_UART_Transmit(&huart6, (uint8_t *)"\n\rMelody finished\n", strlen("\n\rMelody
finished\n"), 10);
        }
    }
}
```

Функция считывания состояний клавиш:

```
// Function to read button states
void ReadButtonStates() {
    for (int row = 0; row < 4; row++) {
        uint8_t rowMask = ~(1 << row); // Set active row
        HAL_I2C_Mem_Write(&hi2c1, 0xE2, 0x01, 1, &rowMask, 1, 10);
        HAL_I2C_Mem_Write(&hi2c1, 0xE2, 0x03, 1, &rowMask, 1, 10);
        uint8_t buttonData;
        HAL_I2C_Mem_Read(&hi2c1, 0xE2, 0x00, 1, &buttonData, 1, 10);
        buttonStates[row][0] = (buttonData >> 4) & 1;
        buttonStates[row][1] = (buttonData >> 5) & 1;
        buttonStates[row][2] = (buttonData >> 6) & 1;
    }
}
```

Функция смены состояний:

```
void switch_mode(){
	if(HAL_GetTick()-time<100)return;
	int state = HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_15);
	if(state == 0 && buttonState == 1) {
		if (debugMode){
			debugMode = 0;
			transmit("Debug mode on \n", strlen("Debug mode on \n"));
		}
		else{
			debugMode = 1;
			if(pmask)__set_PRIMASK(pmask);
			transmit("Debug mode off \n", strlen("Debug mode off \n"));
			ready = 1;
		}
		size = 0;
	}
	buttonState = state;
	time = HAL_GetTick();
}
```

Функции обработки сообщений:

```
// Function to handle button actions
void HandleMelodySelection(int col) {
    currentMelodyIndex = col;
    melodyStartFlag = 1;
}
void HandleToggleState() {
    toggleState = 1 - toggleState;
    if (!toggleState) {
        dynamicMelodyNoteCount = 0;
    }
}
```

```
void HandleDynamicMelody(int row, int col) {
    if (dynamicMelodyNoteCount >= 5) {
        return;
    }
    int note = 0;
    if (row == 2) {
        note = 500 + col * 100;
    } else if (row == 3) {
        note = 800 + col * 100;
    }
    if (note > 0) {
        melodies[4][dynamicMelodyNoteCount++] = note;
    }
}

       time = HAL_GetTick();
}


void ProcessButtonPress(int row, int col) {
    if (row == 0 && col < 3) {
        HandleMelodySelection(row * 3 + col);
    } else if (row == 1 && col == 2) {
        HandleToggleState();
    } else if (toggleState) {
        HandleDynamicMelody(row, col);
    }
}
void HandleButtonActions() {
    for (int row = 0; row < 4; row++) {
        for (int col = 0; col < 3; col++) {
            if (buttonStates[row][col] != previousButtonStates[row][col]) {
                char buffer[50];
                if (debugMode) {
                    if (!buttonStates[row][col]) {
                                    sprintf(buffer, "\rButton %d-%d Pressed\n", row + 1, col + 1);
                                } else {
                                    sprintf(buffer, "\rButton %d-%d Released\n", row + 1, col + 1);
                                }
                    HAL_UART_Transmit(&huart6, (uint8_t *)buffer, strlen(buffer), 10);
                }
                if (!debugMode && !buttonStates[row][col]) {
                    ProcessButtonPress(row, col);
                }
            }
            previousButtonStates[row][col] = buttonStates[row][col];
        }
    }
}
```

Код основной программы:

```
HAL_TIM_Base_Start_IT(&htim6);
HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);

while (1) {
    ReadButtonStates();
    HandleButtonActions();
}
```

Вывод:

В ходе лабораторной работы мы познакомились с интерфейсом I2C и устройством матричной клавиатуры