Міністерство освіти і науки України

Харківський національний університет імені В. Н. Каразіна

Кафедра електроніки та управляючих систем

**Практична робота №3**

«Розробка алгоритмів реального часу»

*з курсу «Архітектура комп'ютерів»*

*Виконав*:

Студент групи КІ-31

М. В. Андреєв

*Прийняв*:

Доцент кафедри електроніки

Та управляючих систем, к.т.н.

С. М. Рева

Харків, 2021 р.

***Мета роботи***

Отримання досвіду розробки алгоритмічних рішень, що будуються на основі системного аналізу задачі з використанням особливостей архітектури процесора, для якого створюється програма. Об'єктом аналізу є анімаційний процес — рухоме зображення, що програмно формується на світлодіодному дисплеї лабораторного макета.

***Завдання***

На основі системного аналізу анімаційного процесу та з урахуванням архітектурних особливостей мікроконтролера АТ89С51 розробити алгоритм програми, яка б забезпечувала відображення обраного варіанту анімації на світлодіодних дисплеях лабораторного макету ЛМ51А. Варіант анімації – 16, «Черв’ячок». Період зміни кадрів – 0.36 с.

***Порядок виконання роботи***

Проаналізував дану мені анімацію, зробив висновок, що в мене є чотири етапи руху «черв’ячка»: ініціалізація, коли він з’являється, коли він рухається по порту P1 та коли він поступово зникає по порту P3. Більш того, щоб розібратися з кадрами анімації, розбив GIF-файл на кадри і зрозумів, що перший світлодіод горить 2 кадри підряд, як і останній. Тобто, можна уявити, як цей «черв’ячок» стоїть на початку в «зібраному» стані, готуючись «зайти» на порт P1, потім «головою» переміщується на перший біт «униз», а далі знов «збирається» наступним кроком але це десь «за кадром». Потім знову рух головою і вже світиться другий біт, а перший ні, а потім вони вдвох світяться. Таким чином, «черв’ячок» «збирається в гармошку» і «пересувається» по бітам. Перші кадри це і є появлення «черв’ячка», а далі це вже є етап руху. При переході на інший порт треба перенести спочатку «голову» і не «загубити» «хвоста». Далі переміщується черв’ячок по порту P3 та закінчує свій рух приблизно так само, як і почав – також коли «голова» вже «за кадром», а тіло «черв’ячка» «гармошкою» і на останньому біті P3 знаходиться «хвіст», а потім коли «голова» далі рухається «за кадром», а «хвіст» ще на місці, а вже потім «зникає» з порту.

Тобто, ще раз:

* Етап ініціалізації. Пусті біти, ініціалізація таймерів, змінних, флагів, тощо.
* Етап появи, коли «черв’ячок» не повністю є на порту P1.
* Етап руху, коли і «голова», і «хвіст» є повністю на порту P1 та рухаються далі.
* Етап закінчення, коли вже «черв’ячок» не повністю є на порту Р3.

Так як кадри за умовою міняються кожні 0.36 с, то щоб перейти з першого на другий етап, повинно пройти 0.36 с, з другого на третій – 0.72 с (2 кадри), а четвертий починається як тільки «хвіст» дійде біту Р3.6, бо саме після цього «зникне» «голова».

У нашій моделі процесора є два таймери-лічильники Т/С0 та T/C1. Ці регістри мають в собі два регістри TL0/TL1 та TH0/TH1. У цих таймерів є 4 режими роботи. Нульовий режим має 13 біт, при переповненні якого викликається переривання. Перший режим схожий на нульовий, але має в собі 16 біт. Другий режим можна назвати самопрограмованим режимом, бо робота організована таким чином, що переповнення 8-бітного лічильника TL призводить не тільки до встановлення флага TF, але й автоматично перезавантажує в TL вміст старшого байта регістру TH, який попередньо було задано програмним шляхом. В третьому режимі Т/C0 и Т/C1 працюють порізаному. Цей режим не дуже популярний та більш складний за інші.

Щоб запрограмувати нам таймер, треба встановити потрібний нам режим через встановлення відповідних бітів у регістрі режимів роботи таймерів-лічильників TMOD. Після цього занести в регістри TL та TH потрібні числа, які ми розраховуємо для наших умов. Останній крок – ввімкнути потрібний таймер через біт у TCON – регістрі управління/статусу таймерів-лічильників.

Найбільше значення, яке може відрахувати таймер це 65 536 мкс. Мені для завдання треба 0.36 с. (360 000 мкс). Я можу запрограмувати таймер Т/С0 на 60 000 мкс на частоті 12 МГц:

1. Знаходимо період тактової частоти:
2. Тривалість машинного циклу:
3. Кількість необхідних машинних циклів:
4. Таку затримку може забезпечити нульовий та перший режим роботи таймера, але в цьому

випадку обираємо перший режим як більш зручний та з більшими можливостями.

1. Розрахуємо попередньо встановлене значення для першого режиму: 65 536 – 60 000= 5 536.
2. Обчислюємо TH0 і TL0: 5 536/ 256 = 21 + 5 в залишку, тобто: ТН0 = 21 (15h), ТL0 = 160 (A0h).

Таким чином я можу порахувати 0.06 с. Яким же чином отримати 0.36? Для цього я буду використовувати віртуальний таймер, який буде відраховувати 6 раз по 0.06 с., щоб отримати 0.36 с. Для цього я створив змінну VTIM = 6, і використовую команду DJNZ (декремент та умовний перехід), щоб відлічити 6 разів по 0.06 с у програмі обробці переривань. Коли ми дійшли нуля, виконуємо програму обробки, і першим же чином встановлюємо заново значення 6 у віртуальний таймер.

Для того щоб пересувати «черв’ячка» по портам, я використовую команди MOV, RL, ORL. Переписую значення «голови» або «хвоста» до регістру А, роблю здвиг та записую нове значення здвигнутої «частини тіла» до її змінної. «Черв’ячок» створюється через складання «голови» та «хвоста» до регістру А та записом його вмісту до потрібного порту. Далі йдуть різні модифікації різних флагів. Мої флаги та змінні:

HEAD (англ. голова) – змінна, що зберігає біт «голови»;

TAIL (англ. хвіст) – змінна, що зберігає біт «хвоста»;

FPART (англ. Flag Part – флаг частини) – флаг, що відповідає за те, яку саме «частину тіла» ми будемо міняти (0 – «голова», 1 – «хвіст»);

FMOVE (англ. Flag Move – флаг руху) – флаг, що зберігає в собі режим або появлення/зникнення «черв'ячка» (0), або рух «черв'ячка».

FMODE (англ. Flag Mode– флаг режиму) – флаг, що вже детальніше описує режим або появлення (1), або зникнення (0).

FPORT (англ. Flag Port – флаг порту) – флаг, який вказує в який порт Р1 (0), або Р3 (1) буде виконуватися запис з регістру А.

В самому початку FPART дорівнює 1. Здається, що це «хвіст», як сказано вище, але на першій ітерації він буде інвертований командою CPL, і це стане «голова». Це потрібно щоб кожного разу при виконанні програми переривань мінялася «частина», яка рухається.

Потім йде перевірка на те, чи рухається зараз «черв’як», чи він з’являється, чи зникає. На початку він з’являється, тому FMOVE = 0. А FMODE натомість, дорівнює 1. Тому першим кроком буде запис до порту Р1 значення «голови»: 1. Наступним кроком буде переміщатися «хвіст», але це не буде відображатися. «Хвіст» стане рівним 0 (замість 80h). Встановлюється флаг того, що ми починаємо рух «черв’ячка». При русі черв’ячка «пустий» «хвіст» буде заповнено через біт переносу С.

Далі в залежності від флагів, як було зазначено вище, використовуються операції побітового здвигу, чи то з бітом переносу, чи то без.

Якщо «хвіст» дійшов передостаннього біту порту Р1 (я кажу тут за «картинку», але всі операції у нас у регістрі А, тому правильніше сказати за біт регістра А), тоді спочатку голова переміщується у порт Р3, а хвіст залишається у порті Р1, потім ми пересуваємо «хвіст» далі униз, і якщо останній біт порту це «хвіст», тоді з бітом переносу зміщуємо «хвіст», та встановлюємо флаг, що тепер буде робота с портом Р3 і що далі буде поступово етап завершення.

Наступне переміщення «хвосту» з флагом FPORT = 1, очищує порт Р1 та встановлює режим появлення/зникнення «черв'ячка».

За всіма флагами попадаємо до етапу завершення (хоча тут ще продовжується рух по Р3). Тут вже важливо, що переміщення було з бітом переносу, щоб очищати всі змінні поступово. Через час, спочатку зникне (очиститься) «голова», а потім і «хвіст».

***Блок-схема***

