МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни

“Мікропроцесорні пристрої”

для студентів спеціальності
141 – ЕЛЕКТРОЕНЕРГЕТИКА, ЕЛЕКТРОТЕХНІКА ТА ЕЛЕКТРОМЕХАНІКА
усіх форм навчання

2017

Методичні вказівки до лабораторних робіт з дисципліни «Мікропроцесорні пристрої» для студентів спеціальності 141 – ЕЛЕКТРОЕНЕРГЕТИКА, ЕЛЕКТРОТЕХНІКА ТА ЕЛЕКТРОМЕХАНІКА усіх форм навчання. /Укл: В.В. Осадчий, О.С. Назарова. - Запоріжжя: ЗНТУ, 2017. – 54 с.

Укладачі:
В.В. Осадчий, к.т.н., доцент
О.С. Назарова, к.т.н., доцент

Рецензент: В.І. Бондаренко, к.т.н., доцент

Відповідальний за випуск: В.І. Бондаренко, к.т.н., доцент

Затверджено
на засіданні кафедри
Електропривода і автоматизації
промислових установок
протокол № 7 від 02.03.2017 р.

Затверджено
на засіданні НМК ЕТФ
протокол № 3 від 20.03.2017 р.

ЗМІСТ

ПЕРЕДМОВА

Методичні вказівки складаються з двох розділів і одного додатку і мають рекомендації до виконання лабораторних робіт з дисципліни МІКРОПРОЦЕСОРНІ ПРИСТРОЇ у відповідності до навчальних планів ОКР бакалаврів.

В першому розділі наведені короткі відомості про інструментальне програмне забезпечення для розробки та налагодження програм, а також методичні вказівки щодо їх написання. У другому розділі наведено перелік лабораторних робіт, їх короткий зміст та рекомендації з їх виконання.

У додатку подано перелік команд мікроконтролера Intel 8051, який включає назву, мнемокод та опис дій, що виконуються.

Для студентів спеціальності 141 – ЕЛЕКТРОЕНЕРГЕТИКА, ЕЛЕКТРОТЕХНІКА ТА ЕЛЕКТРОМЕХАНІКА усіх форм навчання.

1 РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

1.1 Засоби ProView для налагодження взаємодії з об'єктами керування.

Для прискорення процесу розробки керуючих програм існують спеціальні програмні засоби, що називаються інструментальним програмним забезпеченням. Одним з таких засобів є Proview від Franklin Software. Вказане інструментальне програмне забезпечення (ПЗ) дозволяє набирати, редагувати та зберігати вихідний текст програми, перетворювати його в машинний код, здійснювати перевірку працездатності програми, як окремих її частин, так і вцілому.

Інтегровані програмні емулятори апаратних засобів дозволяють імітувати роботу портів і таймерів мікроконтролера, оброблювати зовнішні переривання й переривання від таймерів.

Інструментальне ПЗ Proview фірми Franklin Software Inc. містить декілька вікон, що призначені для налагодження взаємодії мікроконтролера з об'єктами керування. У вікні Main Registers (рисунок 1.1) відображається вміст лічильника команд PC, акумулятора ACC, слова стану процесу PSW, показчика стеку SP, показчика даних DPTR, допоміжного акумулятора B, біту позики/перенесення C, регістру масок переривань IE; регістру блокувань преривань EA, показчика банку регістрів RB, регістрів загального призначення R0-R7; портів P0-P3, регістру керування/статусу таймера TCON, регистрів таймерів THL0-THL2; регістру керування потужністю PCON.

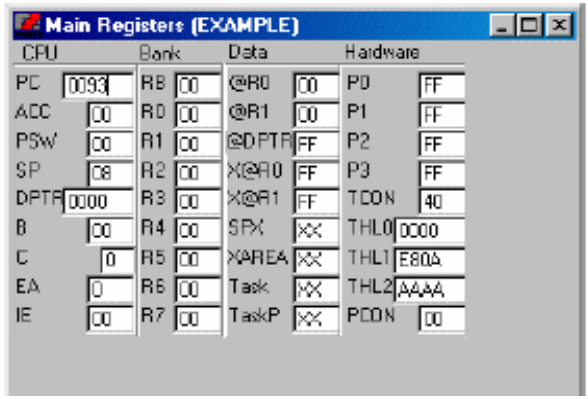


Рисунок 1.1 – Вікно регістрів

Крім того, через пункт Hardware меню View (рисунок 1.2) можна відкрити вікна паралельних портів контролера переривань, таймерів і послідовного порту.

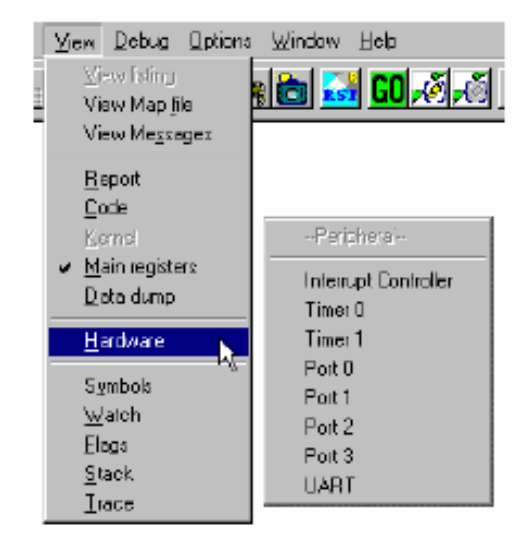


Рисунок 1..2 – Меню View

У інформаційному вікні контролера переривань «Interrupt Controller» (рисунок 1.3) вказується статус і пріоритет всіх джерел переривань: «Enable» - наявність переривання, «Not Enable» - відсутність переривання.

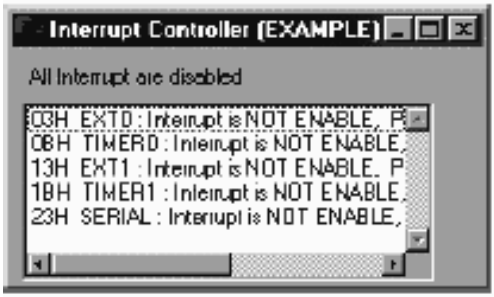
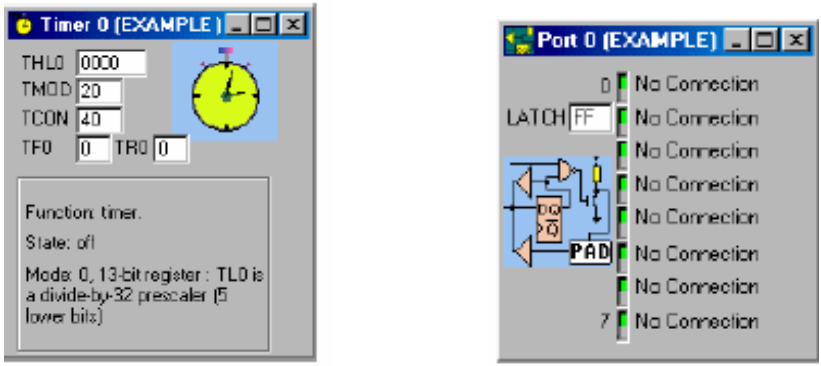


Рисунок 1.3 – Вікно контролера переривань

В рядку стану (рисунок 1.4) при виконанні програми у покроковому режимі або в режимі анімації показується поточний реальний час.

Рисунок 1.4 – Рядок стану



а) б)

а - вікно стану таймеру; б - вікно стану паралельного порту

Рисунок 1.5 – Вікна Proview таймера та паралельного порту

У вікнах таймерів (рисунок 1.5) відображається вміст 16-бітного таймера/лічильника THLx, регістру режиму роботи таймера/лічильника TMOD, регістра керування/статусу таймера TCON, флаг переповнення таймера TFx, біт керування таймера TRx. В цьому вікні також вказується стан і режим роботи таймера.

У вікнах паралельних портів (рисунок 1.5, б) відображається вміст регістру-фіксатора LATCH і окремих ліній порту. Стан ліній порту може бути змінено за допомогою миші.

У вікні послідовного порту UART (рисунок 1.6) відображаються дані буфера передавача Buffer, режим роботи і швидкість. У буфер приймача Input може бути введена послідовність байтів. Інформація може бути представлена у символьному вигляді ASCII або у шістнадцятковій формі HEX. Буфер очищується за допомогою кнопки Reset Buffer.

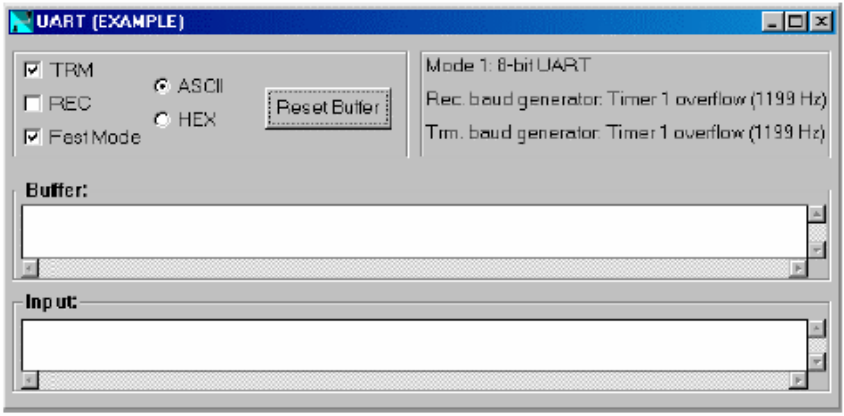


Рисунок 1.6 – Вікно послідовного порту

При відлагодженні програми (рисунок 1.7) часто виникає необхідність призупинити її виконання на певній команді.

Breakpoints – це маркери у вашій програмі, які примушують відлагоджувальник зупинити її виконання у «реальному масштабі часу». Ви можете встановлювати маркери на будь-яких командах програми (рисунок 1.8, 1.9).

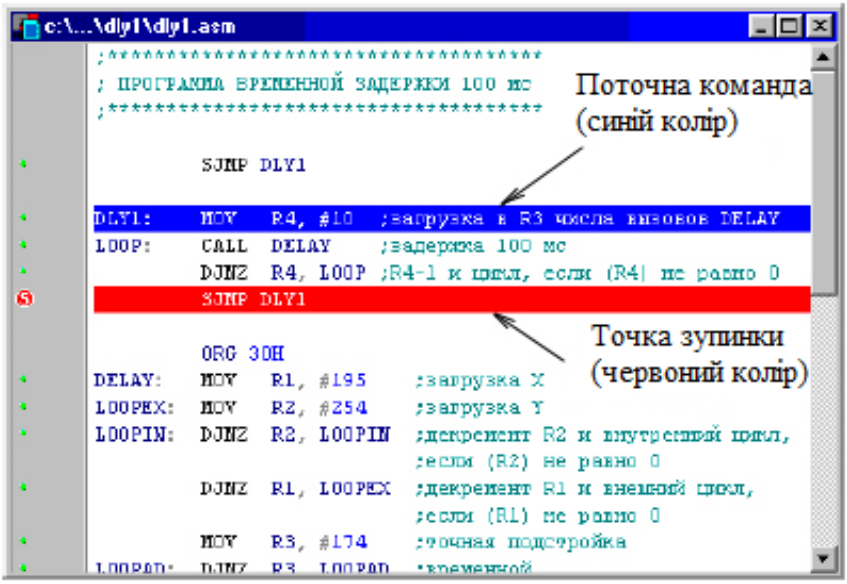
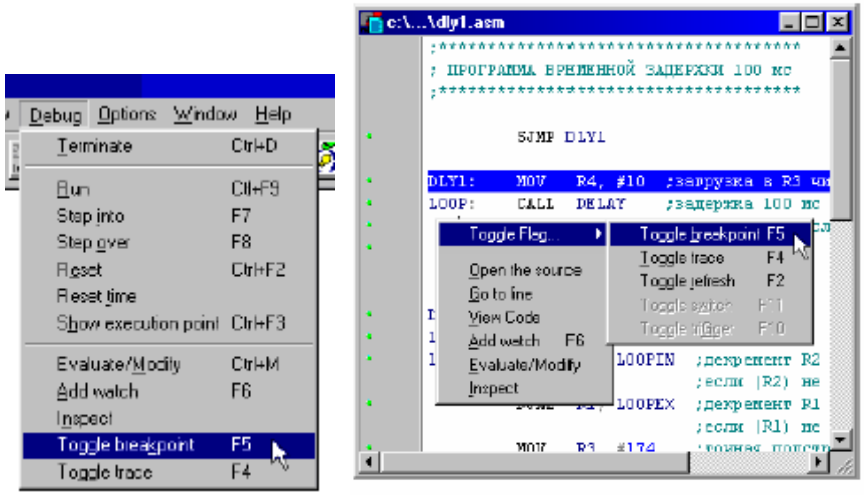


Рисунок 1.7 – Вікно відлагодження



а) б)

а) з головного меню; б) з контекстного меню

Рисунок 1.8 – Встановлення Breakpoint

1.2 Послідовність написання та відлагодження програми.

1.2.1 Ознайомитися з теоретичними відомостями та прикладами з тематики запропонованого завдання.

1.2.2. Скласти блок-схему виконання завдання:

- виділити із умов завдання вихідні дані;
- визначити послідовність виконання дій для вирішення задачі;
- визначити регістри, в яких будуть зберігатися необхідні дані, вигадати для них коротке та інформативне ім'я, а також чіткий описуючий коментар;
- за необхідності визначити перелік дій, які будуть (якщо буде необхідність) повторюватися в ході виконання завдання і вирішити, що саме для цього використати: цикл (декілька циклів) або підпрограму.

1.2.3 Згідно складеної блок-схеми написати текст програми з коментарями до кожної команди (групи команд), які реалізують виконання однієї з умовних частин усієї програми. Бажано уникати коментарів, що дублюють дію окремої команди, тобто дані, які відомі або можуть бути взяті з опису ситеми команд. Коментар повинен роз'яснювати з якою метою використовується команда (група команд) у конкретній програмі.

Наприклад:

```
не бажано (погано) INC R1      ;R1+1→R1
бажано (добре )    INC R1      ;збільшення вказівника
                               ;елементів масиву
```

Процес створення програми бажано розбити на умовні частини:

- складання основного переліку дій для виконання завдання;
- введення вихідних даних;
- опис дій, які виконуються циклічно;
- підпрограми, які будуть викликатися під час виконання основної програми.

Перевірити наявність міток у тексті програми і команд переходу за мітками, а також команд виклику підпрограм і виходу з підпрограм.

Наприклад:

```
org 0h
     mov r0, #30h ; адреса початкового елемента першого
                    масиву
     mov r7, #8h  ; кількість елементів першого масиву
     call count_null; виклик п/п підрахунку
                    кількості елементів масиву, що
                    дорівнюють нулю
     mov b, a     ; збереження результату виконання
                    підпрограми
     mov r0, #40h ; адреса початкового елемента другого
                    масиву
     mov r7, #8h  ; кількість елементів другого масиву
     call count_null ; виклик п/п підрахунку кількості
                    елементів масиву, що дорівнюють нулю
     cjne a, b, m_ne; порівняння результатів підрахунків
     mov 20h, #0h ; запис у комірку 20h числа 0h, якщо
                    кількість елементів, що дорівнюють
                    нулю, в обох масивах однакова
      jmp m_end ;
m_ne:
     jc m1 ;
     mov 20h, #2h ; запис у комірку 20h числа 2h, якщо
                    у другому масиві кількість елементів,
                    що дорівнюють нулю, більша, ніж у
                    першому
     jmp m_end

m1:
     mov 20h, #1h ; запис у комірку 20h числа 1h, якщо
                    у першому масиві кількість елементів,
                    що дорівнюють нулю, більша ніж у
                    другому

m_end:
     jmp m_ne;
     count_null:  ; п/п підрахунку кількості елементів
                    масиву, що дорівнюють нулю
     mov r6, #0h  ;встановлення початкового стану
                    лічильника
cnt_m1:           ; початок циклу
     mov a, @r0   ;
```

```
     jnz cnt_m2    ; перехід, якщо елемент масиву не
                   дорівнює нулю
     inc r6        ; збільшення вмісту лічильника на 1
cnt_m2:            ;
     inc r0        ; збільшення "вказівника" елементів
                   масиву на 1
     djnz r7, cnt_m1     ; перевірка завершення циклу
     mov a, r6     ; запис результату підрахунку у
                   акумулятор
     ret
END
```

1.2.4 Перевірити правильність програми, виконавши такі дії:

Набрати у вікні вводу програми одну з умовних частин всієї програми;

Зберегти файл з цим текстом програми
(File / Save as / [им'я файлу латинськими літерами]. asm);

Запустити проект для відлагодження (Project / Build all).

За наявності у тексті програми синтаксичних помилок, у вікні «Message» з'явиться виділене червоним кольором повідомлення «ERROR», яке вказує на рядок, що містить помилку.

Запустити проект для перевірки працездатності цієї частини програми (Debug / Start). При запуску програми доступні такі функції емулятора (таблиця 1.1).

Вікно «Main Registers» дозволяє перевіряти стан портів та регістрів та порівнювати його з бажаним.

У вікні «Code» відображуються адреси, коди, мнемоніки програми.

Перевірити стан і режим роботи таймерів можна додатково відкривши відповідні вікна (View / Hardware / Timer 0 та/або Timer 1, Timer 2).

За необхідності не програмного керування окремими лініями паралельних портів їх стан може бути змінений за допомогою миші у вікнах паралельних портів (View / Hardware / Port 0 та/або Port 1, Potr 2, Port 3).

Таблиця 1.1 - Функції емулятора

Граф. зображення	Назва	Функції
	«Animate»	Анімоване виконання програми – при запуску програми у відповідності до порядку виконання команд вони будуть виділені рядком синього кольору
	«Step into»	Покрокове виконання програми – при натисканні на цю кнопку буде виконана тільки одна команда
 	«Run» / «Stop»	Запуск/зупинка програми – після запуску програми ця ж кнопка використовується для зупинки виконання програми, при повторному натисканні програма буде запущена з того місця, на якому була зупинена.
	«Reset»	Повернення у початкове положення – при наступному запуску виконанння програми почнеться з першої команди.

1.2.5 Після того як відлагоджена одна частина програми, слід поступово доповнювати її іншими частинами до повного тексту програми і повної працездатності.

2 ЛАБОРАТОРНІ РОБОТИ

2.1 Лабораторна робота №1

Паралельне введення-виведення інформації

Мета: ознайомитися з емулятором семисегментного індикатора та особливостями його взаємодії з програмним забезпеченням Franklin Software.

2.1.1 Короткі теоретичні відомості.

Семисегментний індикатор (ССІ) - це один з простіших індикаторів, який використовується для виведення інформації з мікроконтролера (МК). Він підключається до паралельного порту таким чином, що кожному сегменту відповідає певний біт порту (рисунок 2.1).

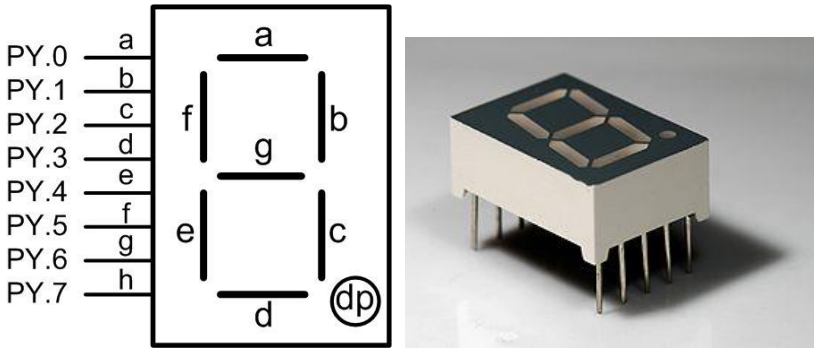


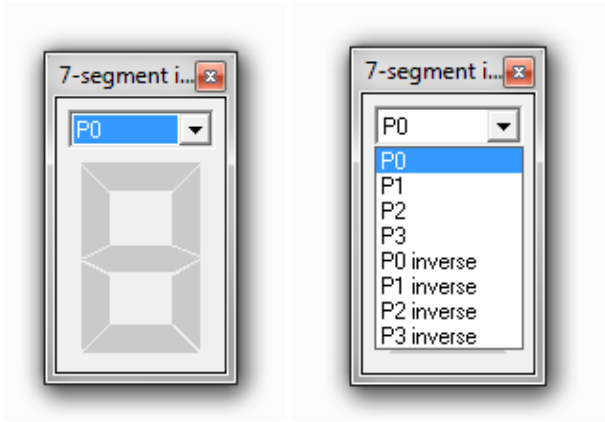
Рисунок 2.1 - Семисегментий індикатор

Наприклад, для того, щоб на семисегментному індикаторі з'явилась цифра «5», треба в один із чотирьох портів МК вивести двійкове число «01101101b» (таблиця 2.1).

Таблиця 2.1 – Відповідність сегментів індикатора для виведення цифри «5»

Біт порту	PY.7	PY.6	PY.5	PY.4	PY.3	PY.2	PY.1	PY.0
Сегмент індикатора	DP	G	F	E	D	C	B	A
Бажаний стан	0	1	1	0	1	1	0	1

Для сполучення Proview Franklin Software з емулятором семисегментного індикатора треба запустити файл 7segment_1.exe. Після цього відкриється вікно, зовнішній вигляд якого показано на рисунку 2.2, а. Зі списку, що відкривається вибирається потрібний порт (рисунок 2.2, б).



а б

а – зовнішній вигляд; б – список можливих для використання портів

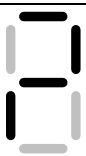
Рисунок 2.2 – Вікно емулятора CCI

Після запуску програми індикації, буде здійснено виведення необхідних знаків на індикатор, при цьому сегменти будуть змінювати колір.

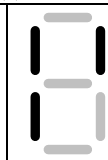
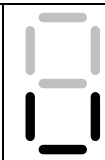
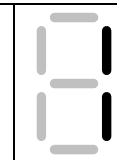
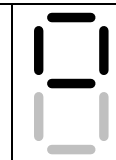
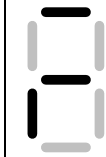
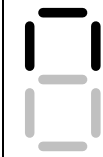
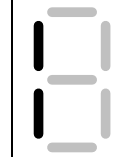
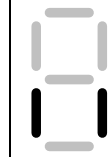
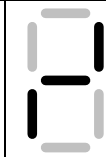
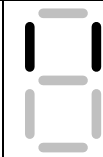
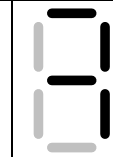
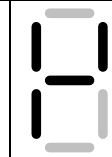
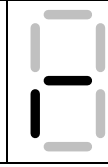
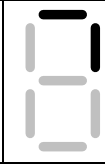
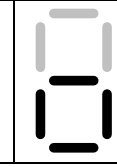
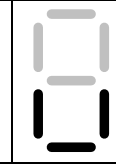
2.1.2 Завдання.

Написати програму для індикації знаків на семисегментному індикаторі згідно варіанта індивідуального завдання (таблиця 2.2), використавши для цього порти P0 і P1.

Таблиця 2.2 – Дані для виконання індивідуального завдання

№	Символи		№	Символи		№	Символи	
	P0	P1		P0	P1		P0	P1
1.			11.			21.		
2.			12.			22.		
3.			13.			23.		
4.			14.			24.		
5.			15.			25.		
6.			16.			26.		

Продовження таблиці 2.2

7.			17.			27.		
8.			18.			28.		
9.			19.			29.		
10.			20.			30.		

2.1.3 Зміст звіту з лабораторної роботи.

Звіт з лабораторної роботи повинен містити титульний лист, тему, мету лабораторної роботи, умову індивідуального завдання, текст програми з коментарями до кожної команди, рисунок, отриманий шляхом Print Screen з результатами лабораторної роботи, висновки з лабораторної роботи.

2.1.4 Контрольні запитання.

1. Призначення семисегментного індикатора.
2. Підключення семисегментного індикатора.
3. Процедура виведення інформації на семисегментний індикатор.

2.2 Лабораторна робота №2

Організація часових затримок програмним методом

Мета: ознайомитися з принципом роботи портів МК родини х51, навчитися робити часові затримки програмним методом.

2.2.1 Короткі теоретичні відомості.

Порти введення-виведення інформації

Всі чотири порти МК51 призначені для введення або виведення інформації побайтно. Порти 1 і 2 мають приблизно таку ж структуру, як і порт 3. Кожний порт містить керовані регістр-фіксатор, вхідний буфер і вхідний драйвер.

Вихідні драйвери портів 0 і 2, а також вхідний буфер порта 0 використовуються при звертанні до зовнішньої пам'яті (ЗП). При цьому через порт 0 в режимі часового мультиплексування спочатку виводиться молодший байт даних. Через порт 2 вводиться старший байт адреси у тих випадках, коли розрядність адреси дорівнює 16 біт.

Всі виводи порта 3 можуть бути використані для реалізації альтернативних функцій (таблиця 2.3), які можуть були задіяні шляхом запису у відповідні біти регістра-фіксатора (Р3.0-Р3.7) порта 3

Порт 0 є двонапрямленим, а порти 1, 2 і 3 – квазідвонапрямленими. Кожна лінія портів може бути використана незалежно для введення або виведення інформації. Для того, щоб деяка лінія порта використовувалася для введення, в в D-тригер регістра-фіксатора порта повинна бути записана 1, яка закриває МОП-транзистор вихідного кола.

За сигналом RESET (RST) у регістри-фіксатори всіх портів автоматично записуються одиниці, які настроюють їх тим самим на режим введення.

Всі порти можуть бути використані для організації введення-виведення інформації за двонапрямленими лініями передачі. Однак порти 0 і 2 не можуть бути використані з цією метою у випадку, якщо МК-система має зовнішню пам'ять, зв'язок з якою організується через загальну розділену шину адреси/даних, яка працює у режимі часового мультиплексування.

Запис у порт

При виконанні команди, яка змінює вміст регістра-фіксатора порта, нове значення фіксується у регістрі у момент S6P2 останнього циклу команди. Однак опитування вмісту регістра-фіксатора вихідною схемою здійснюється під час фази P1 і новий вміст регістра-фіксатора з'явиться на вихідних контактах порта тільки в момент S1P1 наступного машинного цикла.

Таблица 2.3 - Альтернативні функції порта 3

Символ	Позиція	Ім'я і призначення
RD	P3.7	Читання. Активний сигнал низького рівня формується апаратурно при звертанні до зовнішньої пам'яті даних
WR	P3.6	Запис. Активний сигнал низького рівня формується апаратурно при звертанні до зовнішньої пам'яті даних
T1	P3.5	Вхід таймера/лічильника 1 або тест-вхід
T0	P3.4	Вхід таймера/лічильника 0 або тест-вхід
INT1	P3.3	Вхід запиту переривання 1. Сприймається сигнал низького рівня або зріз
INT0	P3.2	Вхід запиту переривання 0. Сприймається сигнал низького рівня або зріз
TXD	P3.1	Вихід передавача послідовного порта в режимі УАПП Вихід синхронізації в режимі регістра зсуву
RXD	P3.0	Вхід приймача послідовного порта в режимі УАПП. Введення/виведення данних в режимі регістра зсуву.

Особливості роботи портів

Звертання до портів введення/виведення можливе з використанням команд, які оперують з байтом, окремим бітом і вільною комбінацією бітів. При цьому у тих випадках, коли порт є одночасно операндом і місцем призначення результата, пристрій керування автоматично реалізує спеціальний режим, який називається «читання-модифікація-запис». Цей режим звертання передбачає введення сигналів не з зовнішніх виводів порта, а з його регістра-

фіксатора, що дозволяє виключити неправильне зчитування раніше виведеної інформації.

Подібний механізм звертання до портів реалізовано у таких командах:

ANL – логічне І, наприклад ANL P1 ,A;

ORL – логічне АБО, наприклад ORL P2,A;

XRL – виключне АБО, наприклад XRL P3,A;

JBC – перехід, якщо в адресованому біті одиниця, і наступне скидання біта, наприклад JBC P1.1, LABEL;

CPL – інверсія біта, наприклад CPL P3.3;

INC – інкремент порта, наприклад INC P2;

DEC – декремент порта, наприклад DEC P2;

DJNZ – декремент порта і перехід, якщо його вміст не дорівнює нулю, наприклад DJNZ P3, LABEL;

MOV PX.Y, C – передача біта переносу в біт Y порту X;

SET PX.Y – встановлення біта Y порта X;

CLR PX.Y – скидання біта Y порта може порту X.

Зовсім не очевидно, що останні три команди у наведеному списку є командами «читання-модифікація-запис». Однак це саме так. За цими командами спочатку считується байт із порта може порту, а потім записується новий байт у регістр-фіксатор.

Опитування двійкового давача

У приладах і системах логічного керування об’єктами події у об’єкті керування фіксується з використанням різноманітних цифрових давачів типу так/ні, наприклад, кінцевих вимикачів, які підімкнені до МК. Схема двійкового давача наведена на рисунку 2.3.

Очікування статичного сигнала

Типова процедура очікування події (WAIT) складається з таких дій: введення сигнала від давача, аналізу значення сигналу і передачі керування у залежності від стану давача. На рисунку 2.4 представлена блок-схема алгоритма процедури очікування події замиканням контакта двійкового давача. Конкретна програмна реалізація процедури залежить не тільки від типу МК, але і від того, яким чином давач підключено до МК. Він може бути підключений до однієї з ліній портів МК або до спеціальних тестованих входів (T0, T1 для МК51).

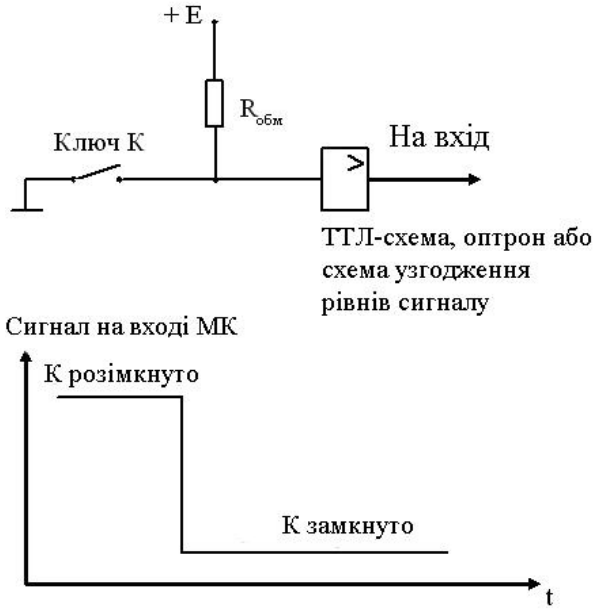


Рисунок 2.3 - Схема двійкового давача

Рисунок 2.4 – Структурно-алгоритмічна схема процедури

```
Далі наведений приклад програмної реалізації
описаного алгоритму
WAITC:JB P1.3,WAITC   ;Очікування замикання
                      ;контакта
Аналогічна реалізація у випадку розмикання
WAITC:JNB P1.3, WAITC ;Очікування розмикання
                      ;контакта
```

Реалізація часових затримок програмним методом

Програмна затримка реалізується шляхом виконання певної кількості команд, за умови, що час виконання кожної команди відомий. Наприклад, для процесорів 8051 при тактовій частоті 12 МГц час виконання команди додавання складає один обчислювальний такт і дорівнює 1 мкс.

Часова затримка малої тривалості.

Для виконання дій, що повторюються, (у нашому випадку це певна кількість команд) зручно використовувати цикли. Наприклад у наступному програмному фрагменті

```
DELAY:          MOV R2, #X
COUNT:          DJNZ R2, COUNT
                RET
```

Число #X, яке записується у регістр R2, визначає скільки разів буде виконана команда DJNZ, що у свою чергу, впливає на загальний час виконня фрагменту. Знайдемо значення X, яке забезпечує затримку тривалістю 100 мкс.

Виконання кожної з команд CALL, MOV, DJNZ, RET становить два машинних цикла (тобто 2 мкс при тактовій частоті 12 МГц). Загальний час виконання фрагменту становить 2+2+2X+2 [мкс], тобто 6+2X [мкс]. Для бажаної затримки, що дорівнює 100 мкс X=(100-6)/2, тобто X=47. Аналогічно занаходяться значення X для затримок іншої тривалості. При цьому, якщо отримане значення є дробовим, його округляють до цілого, що вносить невелику похибку.

Розглянутий вище фрагмент є готовою підпрограмою часової затримки і може бути виконаний у будь-якому місці основної програми за допомогою команди `CALL DELAY`.

Часова затримка великої тривалості

Оскільки розмір регістра = 1 Байт, то час такої затримки не може перевищити 2+2+255*2+2=516 мкс. Якщо у тіло цикла додати два порожніх оператора NOP, час буде дорівнювати t=2+2+4X+2=6+4X. При X=249 час затримки складе 1 мс. Тоді затримку в 1 секунду отримаємо, викликавши 250 разів затримку в 4 мс (рисунок 2.5)

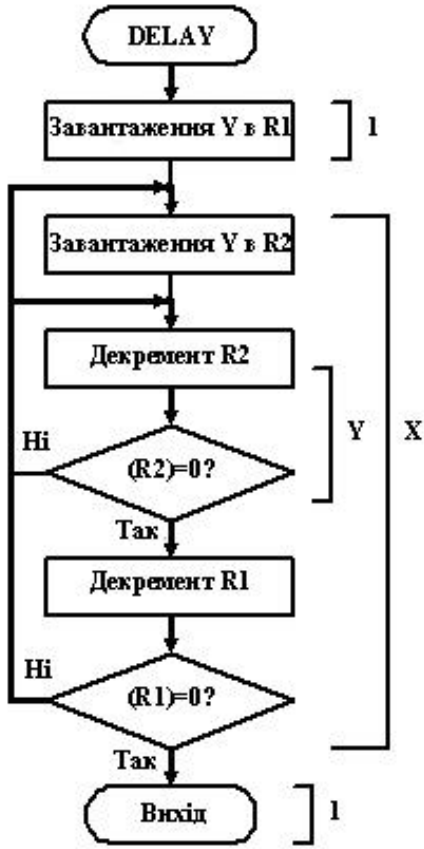


Рисунок 2.5 - Структурно-алгоритмічна схема формування часової затримки

```
DELAY:      MOV R2, #249
COUNT:           NOP
                 NOP
                 DJNZ R2, COUNT
                 RET

ONESEC:     MOV R1, #250
LOOP:       CALL DELAY       ;1 мс
            CALL DELAY       ;+1мс
            CALL DELAY       ;+1мс
            CALL DELAY       ;+1мс=4мс
            DJNZ R1, LOOP    ;250 циклів
                             ;затримки по 4 мс
                             ;складає одну
                             ;секунду
```

2.2.2 Завдання.

Для апаратноъ частини, що зображена на рисунку 2.6 написати програму для реалізації наступних дій,:при натисканні кнопки SB1 здійснити індикацію послідовності знаків на семисегментному індикаторі згідно варіанта індивідуального завдання (таблиця 2.4). Індикацію здійснити у циклі. Між індикацією послідовності знаків організувати затримку за часом.

2.2.3 Зміст звіту з лабораторної роботи.

Звіт з лабораторної роботи повинен містити титульний лист, тему, мету лабораторної роботи, умову індивідуального завдання, текст програми з коментарями до кожної команди (групи команд), які реалізують виконання однієї з умовних частин всієї програми, висновки з лабораторної роботи.

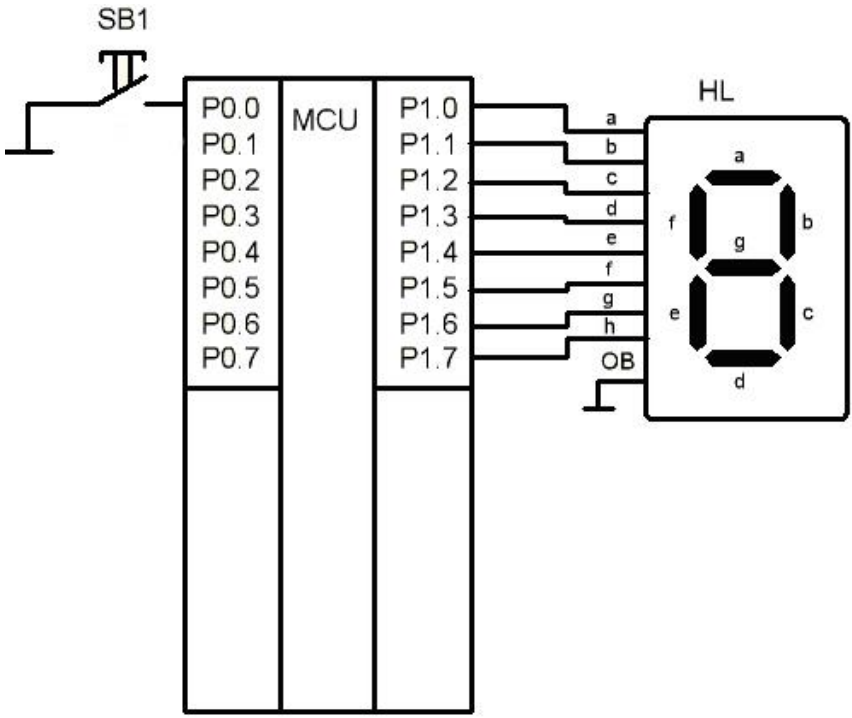


Рисунок 2.6 – Апаратна частина завдання

2.2.4 Контрольні запитання.

1. Принцип роботи портів МК родини x51.
2. Запис у порт. Альтернативні функції порта.
3. Особливості роботи портів.
4. Опитування двійкового давача. Очікування статичного сигналу.
5. Принцип роботи семисегментних індикаторів.
6. Формування часових затримок малої тривалості (до 100 мкс) програмним методом.
7. Формування часової затримки великої тривалості (до декількох секунд) програмним методом.

Таблиця 2.4 – Дані для виконання індивідуального завдання

№ варіанта	Послідовність знаків	Затримка за часом
1	13579	10 мкс
2	02468	20 мкс
3	LABEL	30 мкс
4	GLOBAL	40 мкс
5	FLOPPY	50 мкс
6	12345	10 мкс
7	67890	20 мкс
8	COFFEE	30 мкс
9	APPLE	40 мкс
10	JUICE	50 мкс
11	GLOBAL	10 мкс
12	FLOPPY	20 мкс
13	COFFEE	30 мкс
14	02468	40 мкс
15	13579	50 мкс
16	JUICE	10 мкс
17	APPLE	20 мкс
18	LABEL	30 мкс
19	67890	40 мкс
20	12345	50 мкс
21	HOBBY	10 мкс
22	GLASS	20 мкс
23	CLOUD	30 мкс
24	98765	40 мкс
25	43210	50 мкс
26	CLOUD	10 мкс
27	98765	20 мкс
28	43210	30 мкс
29	HOBBY	40 мкс
30	GLASS	50 мкс

2.3 Лабораторна робота № 3

Система переривань

Мета: навчитися використовувати систему переривань МК сімейства x51 для взаємодії з периферійними пристроями.

2.3.1 Короткі теоретичні відомості.

Система переривань МК-51

Система переривань МК дозволяє реагувати на зовнішні (наприклад, від датчика) і внутрішні (від вбудованого таймера) події, що позбавляє від необхідності регулярно опитувати датчики, на що витрачалося чимало ресурсів, а також виконувати ітерації під час відліку часової затримки.

Спрощена схема переривань МК51 показана на рис. 2.7.

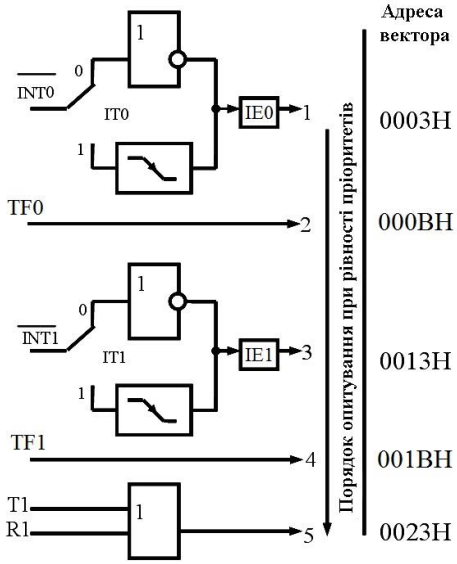


Рисунок 2.7 - Система переривань МК-51

Зовнішні переривання INT0 і INT1 можуть бути викликані або рівнем, або переходом сигналу з 1 в 0 на входах МК51 в залежності від значень керуючих біт IT0 і IT1 в регістрі TCON. Від зовнішніх

переривань встановлюються прапори IE0 і IE1 в регістрі TCON, які ініціюють виклик відповідної підпрограми обслуговування переривання. Скидання цих прапорів виконується апаратурно тільки в тому випадку, якщо переривання було викликано по переходу (зрізу) сигналу.

Якщо ж переривання викликане рівнем вхідного сигналу, то скиданням прапора IE управляє відповідна підпрограма обслуговування переривання шляхом впливу на джерело переривання з метою зняття ним запиту.

Прапори запитів переривання від таймерів TF0 і TF1 скидаються автоматично при передачі управління підпрограмі обслуговування. Прапори запитів переривання RI і TI встановлюються блоком управління УАПП апаратурно, але скидатися повинні програмою. Переривання можуть бути викликані або скасовані програмою, тому що всі перераховані прапори програмно-доступні і можуть бути встановлені / скинуті програмою з тим же результатом, що і якби вони були б встановлені / скинуті апаратними засобами.

У блоці регістрів спеціальних функцій є два регістри (таблиці 2.5 і 2.6), призначених для керування режимом переривань і рівнями пріоритету. Можливість програмної установки / скидання будь-якого керуючого біта в цих двох регістрах робить систему переривань МК51 виключно гнучкою.

Для користувача обробка переривань схожа з викликом підпрограм (CALL), але для повернення з переривання використовується команда RETI, а не RET. Слід звернути увагу на те, що у векторі адрес переривань під кожне переривання виділено мало пам’яті, раціонально в перериванні робити безумовний перехід на вільну область пам’яті, де можна розмістити підпрограму переривань, а потім в кінці цієї підпрограми зробити повернення з переривання.

Наприклад:

```
ORG 13H
JMP LABEL1
      ; Команди основної програми «main»
LABEL1:
      ; Код, який необхідно обробити по перериванню
RETI ; Повернення з переривання
```

Таблиця 2.5 - Регістр масок переривання (РМП)

Символ	Позиція	Ім’я та призначення
EA	IE.7	Розблокування переривань. Скидається програмно для заборони всіх переривань незалежно від станів IE4-IE0
-	IE.6	Не використовується
-	IE.5	Не використовується
ES	IE.4	Біт дозволу переривання від УАПП. Установка / скидання програмою для дозволу / заборони переривань від прапорів T1 або R1
ET1	IE.3	Біт дозволу переривання від таймера 1. Установка / скидання програмою для дозволу / заборони переривань від таймера 1
EX1	IE.2	Біт дозволу зовнішнього переривання 1. Установка / скидання програмою для дозволу / заборони переривань
ET0	IE.1	Біт дозволу переривання від таймера 0. працює аналогічно IE.3
EX1	IE.0	Біт дозволу зовнішнього переривання 0. Працює аналогічно IE.2

Таблиця 2.6 - Регістр пріоритетів переривань

Символ	Позиція	Ім’я та призначення
-	IP.7-IP.5	Не використовуються
PS	IP.4	Біт пріоритету УАПП. Установка / скидання програмою для присвоювання перериванню від УАПП вищого / нижчого пріоритету
PT1	IP.3	Біт пріоритету таймера 1. Установка / скидання програмою для присвоювання перериванню від таймера 1 вищого / нижчого пріоритету
PX1	IP.2	Біт пріоритету зовнішнього переривання 1. Установка / скидання програмою для присвоювання вищого / нижчого пріоритету зовнішньому перериванню INT1
PT0	IP.1	Біт пріоритету таймера 0. Працює аналогічно IP.3
PX0	IP.0	Біт пріоритету зовнішнього переривання 0. Працює аналогічно IP.2

Приклад: При натисканні кнопки, яка підключена до входу переривання INT1, вивести на ССІ, що підключений до Р1 знак «4», а у випадку натискання кнопки, яка підключена до входу переривання INT0, вимкнути всі сегменти ССІ.

```
      ORG 0
      JMP MAIN          ;Перейти на основну
                        ;програму
      ORG 003           ;Вектор переривання по INT0
      JMP INTR0         ;Перехід з області адрес
                        ;векторів переривань в п/п
                        ;переривання від INT0
      ORG 013H          ;Вектор переривання по INT1
      JMP INTR1         ;Перехід з області адрес
                        ;векторів переривань в п/п
                        ;переривання від INT1
MAIN: SETB IE.2         ;Дозволити преривання по INT1
      SETB IE.0         ;Дозволити преривання по INT0
      SETB TCON.0       ;Переривання INT0 по зрізу
                        ;сигналу
      SETB TCON.2       ;Переривання INT1 по зрізу
                        ;сигналу
      MOV P1, # 0H      ;Погасити індикацію
      SETB IE.7         ;Розблокувати всі
                        ;дозволені переривання
L1:   JMP L1            ;Зациклення
INTR0: MOV P1, # 0H     ;Погасити індикацію
      RETI              ;Повернення з підпрограми
                        ;переривання
INTR1: MOV P1, #01100110B;Виведення знака "4" у Р1
      RETI              ;Повернення з п/п
                        ;переривання
      END               ;Кінець тексту програми
```

2.3.2 Завдання.

Для схеми (рисунок 2.8) написати програму для шістнадцяткового реверсивного лічильника (0 ... F). Початковий стан індикатора - «0». При натисканні кнопки «+» значення на індикаторі збільшуються, після натискання кнопки «-» - зменшуються. Лічильник - круговий, тобто при збільшенні після «F» йде «0», при зменшенні після «0» йде «F».

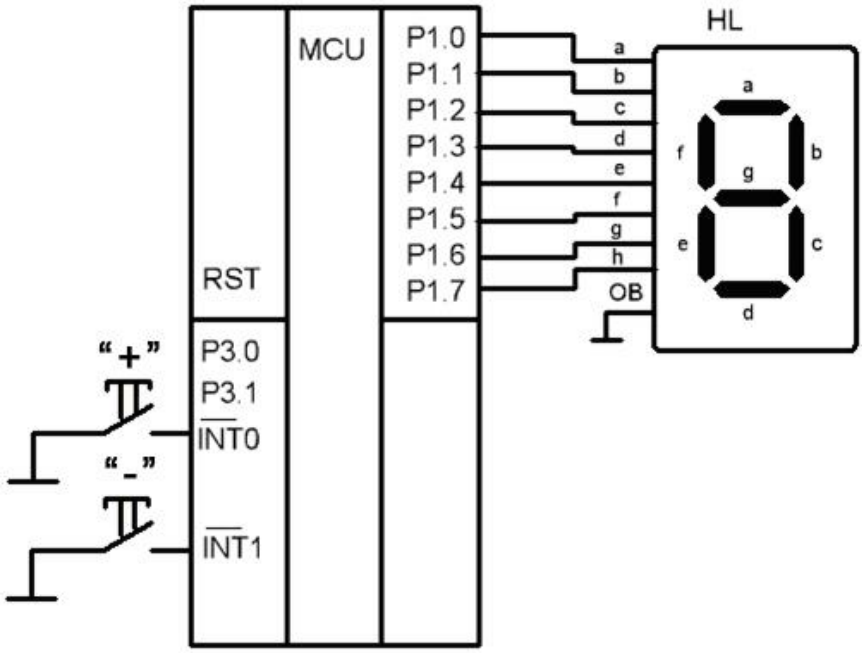


Рисунок 2.8 – Схема з'єднання МК з органами керування та індикатором

2.3.3 Вимоги до звіту з лабораторної роботи.

Звіт з лабораторної роботи повинен містити титульний лист, тему, мету лабораторної роботи, умову індивідуального завдання, текст програми з коментарями до кожної команди (групи команд), які реалізують виконання одної з умовний частин всієї програми, висновки з лабораторної роботи.

2.3.4 Контрольні запитання.

1. Робота з перериваннями. Поняття вектора переривання.
2. Джерела переривання (чим воно може бути викликане).
3. Регістр масок переривання.
4. Регістр пріоритетів переривань.
5. Програмна реалізація обробки переривання (порівняння викликів підпрограм).
6. Механізм обробки переривань (програмно апаратна послідовність дій).
7. Зміна пріоритетів переривань.

2.4 Лабораторна робота №4

Апаратний таймер-лічильник

Мета: навчитись застосовувати апаратный таймер-лічильник.

2.4.1 Короткі теоретині відомості.

Формування часової затримки за допомогою таймеру.

Недоліком програмного способу реалізації часової затримки є нераціональне використання ресурсів мікроконтролера: під час формування затримки він практично простоює, бо не може виконувати жодних задач керування об'єктом. З іншого боку апаратні засоби дозволяють реалізовувати часові затримки на фоні роботи основної програми.

Режими роботи таймера/лічильника

Режим 0

У цьому режимі таймерний регістр має розрядність 13 біт. В момент переходу зі стану «всі одиниці» в стан «всі нулі» встановлюється флаг переривання від таймера TF1 (рисунок 2.9). Вхідний синхросигнал таймера 1 дозволений (поступає на вхід таймера/лічильника (Т/Л), коли керуючий біт TR1 встановлений в 1, а також, або керуючий біт GATE (блокування) є 0, або на зовнішньому виводі запиту переривання INT1 присутній рівень 1.

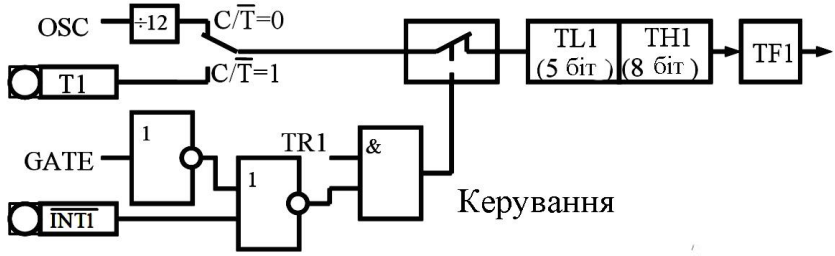


Рисунок 2.9 – Структура таймера/лічильника в режимі 0

Режим 1

Робота будь-якого Т/Л в режимі 1 такаж сама, як і в режимі 0, із-за того, що таймерний регістр (таблиця 2.7) має розрядність 16 біт.

Режим 2

В режимі 2 робота організована таким чином, що переповнення (перехід зі стану «всі одиниці» в стан «всі нулі») 8-бітного лічильника TL1 призводить не тільки до установки флагу TF1, але й автоматично перезавантажує в TL1 вміст старшого байту (TH1) таймерного регістра, яке попередньо було задано програмним шляхом. Перезавантаження не змінює вміст TH1 (рисунок 2.10).

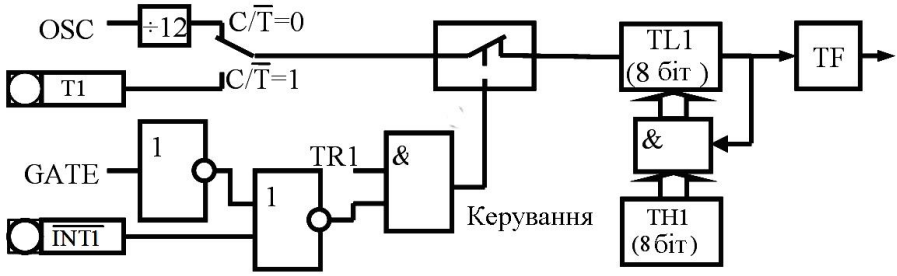


Рисунок 2.10 – Структура таймера / лічільника в режимі 2

Таблиця 2.7 – Регістр керування/статуса таймера

Символ	Позиція	Ім'я та призначення
TF1	TCON.7	Флаг переповнення таймера 1. Встановлюється (стає рівним 1) апаратурно при переповненні таймера/лічильника. Скидається (стає рівним 0) при обробці переривання апаратно.
TR1	TCON.6	Біт керування таймера 1. Встановлюється/ скидається програмою для пуску/зупинки.
TF0	TCON.5	Флаг переповнення таймера 0. Встановлюється апаратурно. Скидається при обробці переривання.
TR0	TCON.4	Біт керування таймера 0. Встановлюється/ скидається програмою для пуску/зупинки таймера/лічильника.
IE1	TCON.3	Флаг фронту переривання 1. Встановлюється апаратурно, коли детектується зріз зовнішнього сигналу (INT1). Скидається при обробці переривання.

Продовження таблиці 2.7

IT1	TCON.2	Біт керування типом переривання 1. Встановлюється/скидається програмно для специфікації запиту INT1 (зріз/ низький рівень).
IE0	TCON.1	Флаг фронту переривання 0. Встановлюється апаратурно, коли детектується зріз зовнішнього сигналу (INT0). Скидається при обробці переривання.
IT0	TCON.0	Біт керування типом переривання 0. Встановлюється/скидається програмно для специфікації запиту INT0 (зріз/ низький рівень).

На вхід таймера/лічильника (Т/Л) можуть подаватися сигнали синхронізації з частою 1 МГц (Т/Л в режимі таймера) або сигнали від зовнішнього джерела (Т/Л в режимі лічильника). Обидва ці режими можуть використовуватися для формування затримок. Якщо використовувати Т/Л в режимі таймера повного формату (16 біт), то можна одержати затримки в діапазоні 1 – 65 536 мкс.

Як приклад розглянемо організацію часової чатримки 50 мс (TIMER). Вважається, що біт IE.7 встановлений, тобто знято блокування всіх переривань. Слід звернути увагу, що в даному випадку використана команда переводу мікроконтролера в режим холостого ходу, який припиняється після перебігу 50 мс. Цей режим реалізований в мікроконтролері Intel 80C51, який виготовлений по технології КМОП, тому при налагодженні програми слід вибирати вказаний тип мікроконтролера (пункт Debug меню Options).

```
 ;организація переходу на мітку NEXT при
                          ;переповнені Т/Л0
      ORG 0BH            ;адреса вектора переривання
                         ;від Т/Л0
      CLR TCON.4         ;зупинка Т/Л0
      RETI               ;вихід з процедури обробки
                         ;переривання
      ORG 30H            ;початкова адреса програми
TIMER:MOV TMOD,#01H      ;настройка Т/Л0
```

```
        MOV TL0, #LOW(NOT(50000-1)) ;завантаження
        MOV TH0, #HIGH(NOT(50000-1));таймера для
                                    ;затримки 50 мс
        SETB TCON.4             ;старт Т/Л0
        SETB IE.1               ;дозвіл переривання
                                ;від Т/Л0
        SETB PCON.0             ;перехід в режим
                                ;холостого ходу
        NEXT:
```

Для вимірювання тривалості сигналу також може використовуватися таймер. Особливо ефективно використання з цією метою таймера в Intel 8051, що має вхід дозволу підрахунку (альтернативна функція входу INT). Сигнал, тривалість якого вимірюється, можна, наприклад, подавати на вхід INT0, а вимірювання тривалості при цьому буде виконуватися в Т/Л0. Програма вимірювання тривалості «додатнього» імпульсу (MSTIMER) матиме такий вигляд:

```
MSTIMER:  MOV TMOD, #00001001B      ;настройка Т/Л0
          MOV TH0, #0               ;скидання
          MOV TL0, #0               ;таймера
          SETB TCON.4               ;старт Т/Л0
WAIT0:    JNB P3.2, WAIT0           ;очікування 1
WAITC:    JB P3.2, WAITC            ;очікування 0
          CLR TCON.4                ;стоп Т/Л0
EXIT:                              ;вихід з процедури
```

Керування повинно бути передано програмі за умови, що на вході INT0 присутній низький рівень. Переривання від Т/Л0 і зовнішнє переривання по входу INT0 повинні бути заборонені.

Після завершення програми в Т/Л0 буде знаходитися число, пропорційне тривалості «додатнього» импульсу на вході INT0. Верхня межа вимірювання дорівнює 65 536 мкс, а максимальна похибка 1 мкс.

Якщо потрібно виміряти інтервал часу більшої тривалості, можна програмним способом підрахувати число переповнень таймера, тобто розширювати його разрядність за рахунок робочого регістра або комірки резидентної пам'яті даних.

Приклад:

При натисканні кнопки SB1 ініціювати переривання INT1 (рисунок 2.11) й засвітити через секунду світлодіод VD1, що підключений до P0.0 (затримку реалізувати за допомогою таймера).

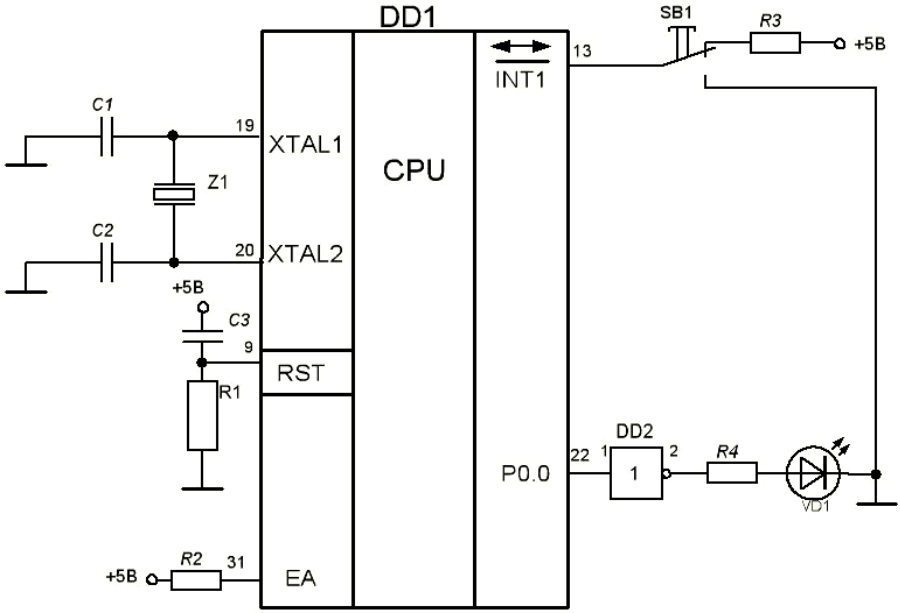


Рисунок 2.11 – Схема з'єднання МК з органами керування та світлодіодом

Приклад для 16-бітного таймера

```
      ORG 0
      JMP MAIN    ;безумовний перехід на мітку
                  ;main
```

```
      ORG 0BH    ;переривання від таймера ТЛ0
      JMP TC0    ;безумовний перехід на мітку
                 ;TC0
      ORG 13H    ;зовнішнє переривання
      SETB TCON.4;запуск таймера
      RETI       ;повернення з переривання
MAIN: MOV IE, #00000110B    ;встановлення флагів
                 ;переривань,які дозволені
      MOV TH0,#HIGH (65535-50000);настройка
      MOV TL0,#LOW (65535-50000); таймера на
                 ;режим 16-ти бітного
                 ;без перезавантаження 2
      MOV TMOD, #00000001B;
      MOV R2,#20
      SETB TCON.2;переривання INT1 по фронту
                 ;сигнала
      SETB IE.7  ;розблокувати всі дозволені
                 ;переривання
      CLR P0.0
LOOP: JMP LOOP   ;бескінечний цикл в очікуванні
                 ;переривання
TC0:  MOV TH0,#HIGH(65535-50000);перезавантаження
      MOV TL0,#LOW (65535-50000) ;таймера, бо
           ;автоперезавантаження - відсутнє

      DJNZ R2, L3;підрахунок кількості
                 ;«спрацьовувань» таймера
      SETB P0.0  ;запалити світлодіод
      CLR TCON.4
L3:   RETI  ;повернення з п/п переривання
      END   ;кінець програми
```

Зверніть увагу, як настроюється таймер.

Двобайтне число можна розділити на старший и молодший байт за допомогою директив відповідно High і Low. Таким чином, число, яке обчислюється відповідно до виразу в дужках, розділяється на 2 окремих байта. Зважаючи, що таймер інкрементний, то необхідно в байти таймера записувати число, з якого починається відлік до

переповнення (65535 мкс). В наведеному прикладі потрібна затримка 50000 мкс, тому в дужках від максимального числа (65535 мкс) віднімається необхідне значення (50000 мкс).

```
      mov TH0,#HIGH (65535-50000) ;Заносимо число в старший
                                  ;байт,
      mov TL0,#LOW (65535-50000)  ;Заносимо число в молодший
                                  ;байт.
```

Приклад для 8-бітного автоперезавантажувального таймера

```
       ORG 0
       JMP MAIN    ;безумовний перехід на мітку
                   ;MAIN
       ORG 0BH     ;переривання від таймера
       JMP TC0     ;безумовний перехід на мітку
                   ;TC0
       ORG 13H     ;зовнішнє переривання
       SETB TCON.4;запуск таймера
       RETI        ;повернення з переривання
MAIN:  CLR P0.0    ;погасити світлодіод
       CLR IE.7
       MOV IE, #00000110B     ;дозволити переривання
                              ;від INT1 і T/Л0
       MOV TMOD, #00000010B;режим таймера
       ;8-ми бітний з автоперезавантажуванням
       MOV TL0, #NOT(250-1)   ;настройка на час
                              ;250мкс
       MOV TH0, #NOT(250-1)   ;настройка на час
                              ;250мкс
       MOV R3, #LOW(4000)     ;4000разів по 250мкс=1с
       MOV R2, #HIGH(4000)    ;4000разів по 250мкс=1с
       SETB TCON.2            ;переривання INT1 по
                              ;фронту сигнала

       SETB IE.7  ; розблокувати всі переривання
L1:    JMP L1     ;безкінечний цикл в очікуванні
                  ;переривання
TC0:   DJNZ R3, L3
```

```
      DJNZ R2, L3
      SETB P0.0  ;запалити світлодіод
      CLR TCON.4 ;зупинка таймера 0
L3:   RETI       ;повернення з п/п переривання
      END        ;кінець програми
```

2.3.2 Завдання.

Написати програму для реалізації секундоміра (00…59). Відповідно до рисунку 2.12. Початковий стан індикаторів – «00». Натискання кнопки «пуск/стоп» запускає відлік часу, повторне натискання – зупиняє. Натискання кнопки «скидання» – обнуляє. Лічильник не круговий, при досягненні максимального значення – зупиняється. Перевищення періоду відліку відображається блиманням індикаторів.

Для зручності можна написати підпрограму-дешифратор для ССІ, яка брала б число з регістра А і повертала б відповідний цьому числу код ССІ, назвемо її, відповідно, Decode

```
DECODE:MOV DPTR,#TABSEMISEG;вказівник в
                           ;початок таблиці
MOVC A,@A+DPTR             ;зчитати код
RET                        ;і повернути через
                           ;акумулятор
                           ;0123456789
TABSEMISEG:
DB 00111111b ; '0'
DB 00000110b ; '1'
DB 01011011b ; '2'
DB 01001111b ; '3'
DB 01100110b ; '4'
DB 01101101b ; '5'
DB 01111101b ; '6'
DB 00000111b ; '7'
DB 01111111b ; '8'
DB 01101111b ; '9'
```

Цей фрагмент кода зручно помістити в кінець програм, в яких використовується цей тип індикаторів.

Тепер, якщо наш індикатор буде підключено, наприклад, до порту 0, нам для введення числа «5» будет достатньо трьох рядків

```
MOV A, #5
CALL DECODE
MOV P0, A
```

Це набагато зручніше, ніж щоразу задавати значення відповідне кожному числу байтів.

2.3.3 Вимоги до звіту з лабораторної роботи.

Звіт з лабораторної робоиі повинен містити титульный лист, тему, мету лабораторної роботи, умову індивідуального завдання, текст програмы з коментарями до кожної команди (групи команд), які реалізують виконання однієї з умовних частин всієї програми, висновки з лабораторної роботи.

2.3.4 Контрольні запитання.

1. Режими роботи таймера-лічильника.
2. Регістр керування таймера.
3. Регістр статуса таймера.
4. Частота змінювання вмісту таймера/лічильника при роботі в режимі таймера.
5. Формування часової затримки за допомогою таймера
6. Організація часової затримки: програмно й за допомогою таймера (порівняльний аналіз).
7. Максимальна тривалість часової затримки, що реалізується за допомогою таймера (для режимів 0, 1, 2).

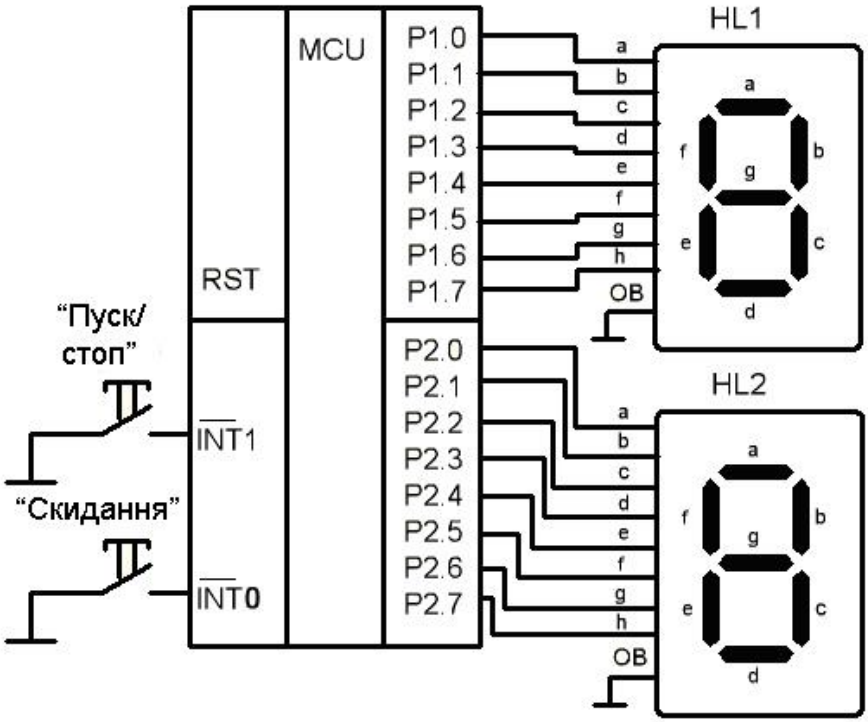


Рисунок 2.12 – Схема з'єднання МК з органами керування та індикаторами

ПЕРЕЛІК ПОСИЛАНЬ

1. Сташин В.В. Проектирование цифровых устройств на одно кристальных микроконтроллерах / В.В. Сташин, А.В. Урусов, О.Ф. Молногонцева – М.: Энергоатомиздат, 1990. – 224 с.

2. Белов А.В. Самоучитель по микропроцессорной технике. – СПб.: Наука и техника, 2003. – 224 с.

3. Мікропроцесорна техніка: навч. посібник / В.В. Ткачов, Г.Грулер, Н. Нойбергер та ін. – Д.: Національний гірничий університет, 2012. – 188 с.

4. Карташов Б. А. Системы автоматического регулирования с микроЭВМ / Б.А. Карташов, Е.А. Шабаев – Зерноград, 2008. – 54 с.

5. Балашов Е.П., Пузанков Д.В. Микропроцессоры и микропроцессорные системы / Под ред. В.Б. Смолова – М.: Радио и свіязь, 1981. – 328 с.

6. Липовецкий Г.П. и др. Однокристальные микроЭВМ семейств МК48, МК51. – М., 1992. – 344 с.

7. Осадчий, В. В. Лабораторный стенд для исследования микропроцессорных систем управления двухмассовым электроприводом / В. В. Осадчий, Е. С. Назарова, В. В. Брылистый, Р. И. Савилов // Електротехнічні та комп'ютерні системи. – 2016. – № 22(98).– С. 33-38. http://dx.doi.org/10.15276/eltecs.22.98.2016.05

8. Назарова, Е. С. Особенности моделирования электромеханических систем с переменным моментом инерции / Е. С. Назарова, Р. А. Ефименко // Вісник НТУ «ХПІ». – 2017. – № 27(1249). – С. 71-74.

9. Осадчий, В. В. Исследование системы управления позиционным электроприводом с дискретным датчиком положения / В. В. Осадчий, Е. С. Назарова, В. В. Брылистый, Р. И. Савилов // Вісник НТУ «ХПІ». – 2017. – № 27(1249). – С. 146-149.

10. Осадчий, В. В. Исследование позиционного электропривода на основе шагового двигателя в микрошаговом режиме / В. В. Осадчий, Е. С. Назарова, С. Ю. Тоболкин // Електротехнічні та комп'ютерні системи. – 2015. – № 19(95).– С. 24-27. http://dx.doi.org/10.15276/eltecs.19.95.2015.06

11. Назарова, Е. С. Математическое моделирование электромеханических систем станов холодной прокатки / Е. С. Назарова // Технічна електродинаміка. – 2015. – Вип. 5 – С. 82-89.

Додаток А

Перелік команд мікроконтролера Intel 8051

Таблиця А.1 - Група команд пересилання даних

Назва команди	Мнемокод	Т	Б	Ц	Операція
Пересилання в акумулятор з регістра (n=0..7)	MOV A,Rn	1	1	1	(A)←(Rn)
Пересилання в акумулятор вмісту комірки ВПД	MOV A,dir	3	2	1	(A)←(dir)
Пересилання в акумулятор байта з РПД (i=0,1)	MOV A,@Ri	1	1	1	(A)←((Ri))
Завантаження в акумулятор константи	MOV A,#d	2	2	1	(A)←#d
Пересилання в регістр з акумулятора	MOV Rn,A	1	1	1	(Rn)←(A)
Пересилання в регістр вмісту комірки ВПД	MOV Rn,dir	3	2	2	(Rn)←(dir)
Завантаження в регістр константи	MOV Rn,#d	2	2	1	(Rn)←#d
Пересилання у комірку ВПД вмісту акумулятора	MOV dir,A	3	2	1	(dir)←(A)
Пересилання у комірку ВПД вмісту регістра	MOV dir,Rn	3	2	2	(dir)←(Rn)
Пересилання у комірку ВПД з комірки ВПД	MOV dir,dir	9	3	2	(dir)←(dir)
Пересилка байта з РПД у комірку ВПД	MOV dir,@Ri	3	2	2	(dir)←((Ri))
Пересилання константи у комірку ВПД	MOV dir,#d	7	3	2	(dir)←#d
Пересилання в РПД вмісту акумулятора	MOV @Ri,A	1	1	1	((Ri))←(A)
Пересилання в РПД вмісту комірки ВПД	MOV @Ri, dir	3	2	2	((Ri))←(dir)

Продовження таблиці А.1

Пересилання в РПД константи	MOV @Ri,#d	2	2	1	((Ri))←#d
Завантаження вказівника даних	MOV DPTR,#d16	13	3	2	(DPTR)←#d16
Пересилання байта коду, пов'язаного з DPTR в акумулятор	MOVC A,@A+DPTR	1	1	2	(A)←((A)+(DPTR))
Пересилання байта коду пов'язаного з PC в акумулятор	MOVC A,@A+PC	1	1	2	(PC)←(PC)+1 (A)← ((A)+(PC))
Пересилання байта із ЗПД в акумулятор	MOVX A,@Ri	1	1	2	(A)←((Ri))
Пересилання байта із ЗПД в акумулятор	MOVX A,@DPTR	1	1	2	(A)←((DPTR))
Пересилання з акумулятора в комірку ВПД	MOVX@Ri,A	1	1	2	((Ri))←(A)
Пересилання з акумулятора комірку ЗПД	MOVX @DPTR,A	1	1	2	((DPTR))←(A)
Завантаження комірки ВПД у стек	PUSH dir	3	2	2	(SP)←(SP)+1 ((SP))←(dir)
Вивантаження зі стека в комірку ВПД	POP dir	3	2	2	(dir)←(SP) (SP)←(SP)-1
Обмін акумулятора з регістром	XCH A,Rn	1	1	1	(A)↔(Rn)
Обмін акумулятора з коміркою ВПД	XCH A, dir	3	2	1	(A)↔(dir)
Обмін акумулятора з непрямоадресованою коміркою ВПД	XCH A,@Ri	1	1	1	(A)↔((Ri))
Обмін молодшими тетрадами між непрямоадресованою коміркою ВПД і акумулятором	XCHD A,@Ri	1	1	1	$(A_{0..3})\leftrightarrow((Ri)_{0..3})$

Таблиця А.2 - Команди арифметичних операцій

Назва команди	Мнемокод	Т	Б	Ц	Операція
Додавання акумулятора і регістра (n=0..7)	ADD A,Rn	1	1	1	(A) ← (A)+(Rn)
Додавання акумулятора і комірки ВПД	ADD A, dir	3	2	1	(A) ← (A)+(dir)
Додавання акумулятора і непрямо адресованої комірки ВПД	ADD A,@Ri	1	1	1	(A) ← (A)+((Ri))
Додавання акумулятора і константи	ADD A,#d	2	2	1	(A) ← (A)+#d
Додавання акумулятора і регістра з урахуванням переносу	ADDC A,Rn	1	1	1	(A) ← (A)+(Rn)+(C)
Додавання акумулятора і коміркою ВПД з урахуванням переносу	ADDC A, dir	3	2	1	(A) ← (A)+(dir)+(C)
Додавання акумулятора і непрямо адресованої комірки ВПД з урахуванням переносу	ADDC A,@Ri	1	1	1	(A) ← (A)+((Ri))+(C)
Додавання акумулятора і константи з урахуванням переносу	ADDC A,#d	2	2	1	(A) ← (A)+#d+(C)
Десяткова корекція акумулятора	DA A	1	1	1	Якщо ($A_{0..3}$)>9 або ((AC)=1), то ($A_{0..3}$) ← ($A_{0..3}$)+6, і якщо ($A_{4..7}$)>9 або ((C)=1), то ($A_{4..7}$) ← ($A_{4..7}$)+6

Продовження таблиці А.2

Віднімання від акумулятора регістра і позики	SUBB A,Rn	1	1	1	(A) ← (A)-(C)-(Rn)
Віднімання від акумулятора комірки ВПД і позики	SUBB A, dir	3	2	1	(A) ← (A)-(C)-(dir)
Віднімання від акумулятора непрямо адресованої комірки ВПД і позики	SUBB A,@Ri	1	1	1	(A) ← (A)-(C)-((Ri))
Віднімання від акумулятора константи і позики	SUBB A,#d	2	2	1	(A) ← (A)-(C)-#d
Інкремент акумулятора	INC A	1	1	1	(A) ← (A)+1
Інкремент регістра	INC Rn	1	1	1	(Rn) ← (Rn)+1
Інкремент комірки ВПД	INC dir	3	2	1	(dir) ← (dir)+1
Інкремент непрямо-адресованої комірки ВПД	INC @Ri	1	1	1	(Ri) ← (Ri)+1
Інкремент покажчика даних	INC DPTR	1	1	2	(DPTR) ← (DPTR)+1
Декремент акумулятора	DEC A	1	1	1	(A) ← (A)-1
Декремент регістра	DEC Rn	1	1	1	(Rn) ← (Rn)-1
Декремент комірки ВПД	DEC dir	3	2	1	(dir) ← (dir)-1
Декремент непрямо-адресованої комірки ВПД	DEC @Ri	1	1	1	(Ri) ← (Ri)-1

Продовження таблиці А.2

Множення акумулятора на регістр В	MUL AB	1	1	4	(B)(A) ← (A)*(B)
Ділення акумулятора на регістр В	DIV AB	1	1	4	(A).(B) ← (A)/(B)

Таблиця А.3 - Команди логічних операцій

Назва команди	Мнемокод	Т	Б	Ц	Операція
Логічне І акумулятора і регістра	ANL A,Rn	1	1	1	(A)←(A) ∧ (Rn)
Логічне І акумулятора і комірки ВПД	ANL A, dir	3	2	1	(A)←(A) ∧ (dir)
Логічне І акумулятора і непрямо адресованої комірки ВПД	ANL A,@Ri	1	1	1	(A)←(A) ∧ ((Ri))
Логічне І акумулятора і константи	ANL A,#d	2	2	1	(A)←(A) ∧ #d
Логічне І комірки ВПД і акумулятора	ANL dir,A	3	2	1	(dir)←(dir) ∧ (A)
Логічне І комірки ВПД і константи	ANL dir,#d	7	3	2	(dir)←(dir) ∧ #d
Логічне АБО акумулятора і регістра	ORL A,Rn	1	1	1	(A)←(A) ∨ (Rn)
Логічне АБО акумулятора и комірки ВПД	ORL A, dir	3	2	1	(A)←(A) ∨ (dir)
Логічне АБО акумулятора і непрямоадресованої комірки ВПД	ORL A,@Ri	1	1	1	(A)←(A) ∨ ((Ri))

Продовження таблиці А.3

Логічне АБО акумулятора і константи	ORL A,#d	2	2	1	(A)←(A) $\vee$ #d
Логічне АБО комірки ВПД і акумулятора	ORL dir,A	3	2	1	(dir)←(dir) $\vee$ (A)
Логічне АБО комірки ВПД і константи	ORL dir,#d	7	3	2	(dir)←(dir) $\vee$ #d
Що виключає АБО акумулятора і регістра	XRL A,Rn	1	1	1	(A)←(A) $\forall$ (Rn)
Що виключає АБО акумулятора і комірки ВПД	XRL A, dir	3	2	1	(A)←(A) $\forall$ (dir)
Що виключає АБО акумулятора і непрямоадресованої комірки ВПД	XRL A,@Ri	1	1	1	(A)←(A) $\forall$ ((Ri))
Що виключає АБО акумулятора і константи	XRL A,#d	2	2	1	(A)←(A) $\forall$ #d
Що виключає АБО комірки ВПД і акумулятора	XRL dir,A	3	2	1	(dir)←(dir) $\forall$ (A)
Що виключає АБО комірки ВПД і константи	XRL dir,#d	7	3	2	(dir)←(dir) $\forall$ #d
Очищення акумулятора	CLR A	1	1	1	(A)←0
Інверсія акумулятора	CPL A	1	1	1	(A) $\leftarrow (\overline{A})$
Зрушення акумулятора вліво	RL A	1	1	1	$(A_{n+1})\leftarrow(A_n)$, n=0..6, $(A_0)\leftarrow (A_7)$

Продовження таблиці А.3

Зрушення акумулятора вліво через прапор переносу	RLC A	1	1	1	$(A_{n+1})\leftarrow(A_n)$, n=0..6, $(A_0)\leftarrow(C)$, $(C)\leftarrow(A_7)$
Зрушення акумулятора вправо	RR A	1	1	1	$(A_{n+1})\leftarrow(A_n)$, n=0..6, $(A_7)\leftarrow(A_0)$
Зрушення акумулятора вправо через прапор переносу	RRC A	1	1	1	$(A_{n+1})\leftarrow(A_n)$, n=0..6, $(A_7)\leftarrow(C)$, $(C)\leftarrow(A_0)$
Обмін місцями тетрад в Акумуляторі	SWAP A	1	1	1	$(A_{0-3})\leftarrow(A_{4-7})$

Таблиця А.4 - Команди роботи з бітами

Назва команди	Мнемокод	Т	Б	Ц	Операція
Очищення переносу	CLR C	1	1	1	$(C)\leftarrow 0$
Очищення біта	CLR bit	4	2	1	$(bit)\leftarrow 0$
Установка переносу	SETB C	1	1	1	$(C)\leftarrow 1$
Установка біта	SETB bit	4	2	1	$(bit)\leftarrow 1$
Інверсія переносу	CPL C	1	1	1	$(C)\leftarrow(\overline{C})$
Інверсія біта	CPL bit	4	2	1	$(bit)\leftarrow(\overline{bit})$
Логічне І біта і прапору переносу	ANL C,bit	4	2	2	$(C)\leftarrow(C)\wedge(bit)$
Логічне І інверсії біта і переносу	ANL C,/bit	4	2	2	$(C)\leftarrow(C)\wedge(\overline{bit})$

Продовження таблиці А.4

Логічне АБО біта і прапора переносу	ORL C,bit	4	2	2	(C)←(C) ∨ (bit)
Логічне АБО інверсії біта і прапора переносу	ORL C,/bit	4	2	2	$(C) \leftarrow (C) \vee (\overline{bit})$
Пересилання біта в прапор переносу	MOV C,bit	4	2	1	(C)←(bit)
Пересилання прапору переносу в біт	MOV bit,C	4	2	2	(bit)←(C)

Таблиця А.5 - Команди передачі керування

Назва команди	Мнемокод	Т	Б	Ц	Операція
Довгий перехід	LJMP	12	3	2	(PC)← dir 16
Абсолютний перехід всередині сторінки у 2 Кбайта	AJMP dir11	6	2	2	(PC)←(PC)+2 $(PC_{0..10})$← dir 11
Короткий відносний перехід всередині сторінки у 256 байт	SJMP rel	5	2	2	(PC)←(PC)+2 (PC)← (PC)+rel
Непрямий відносний перехід	JMP @A+DPTR	1	1	2	(PC)←(A)+(DPTR)
Перехід, якщо акумулятор дорівнює нулю	JZ rel	5	2	2	(PC)←(PC)+2, якщо (A)=0, то (PC)← (PC)+rel

Продовження таблиці А.5

Перехід, якщо акумулятор не дорівнює нулю	JNZ rel	5	2	2	(PC)←(PC)+2, якщо (A)≠0, то (PC)← (PC)+rel
Перехід, якщо прапор переносу дорівнює одиниці	JC rel	5	2	2	(PC)←(PC)+2, якщо (C)=1, то (PC)← (PC)+rel
Перехід, якщо прапор переносу дорівнює нулю	JNC rel	5	2	2	(PC)←(PC)+2, якщо (C)=0, то (PC)← (PC)+rel
Перехід, якщо біт дорівнює одиниці	JB bit,rel	11	3	2	(PC)←(PC)+3, якщо (bit)=1, то (PC)← (PC)+rel
Перехід, якщо біт дорівнює нулю	JNB bit,rel	11	3	2	(PC)←(PC)+3, якщо (bit)=0, то (PC)← (PC)+rel
Перехід, якщо біт установлено, з наступним скиданням біта	JBC bit,rel	11	3	2	(PC)←(PC)+3, якщо (bit)=1, то (bit)←0 і (PC)← (PC)+rel
Декремент регістра і перехід, якщо він не дорівнює нулю	DJNZ Rn,rel	5	2	2	(PC)←(PC)+2, (Rn)←(Rn)-1, якщо (Rn)≠0, то (PC)← (PC)+rel
Декремент комірки ВПД і перехід, якщо її вміст не дорівнює нулю	DJNZ dir,rel	8	3	2	(PC)←(PC)+2, (dir)←(dir)-1, якщо (dir)≠0, то (PC)← (PC)+rel

Продовження таблиці А.5

Порівняння акумулятора з коміркою ВПД і перехід, якщо вони не дорівнюють одне одному	CJNE A, dir,rel	8	3	2	(PC)←(PC)+3, якщо (A)≠(dir),то (PC)← (PC)+rel, якщо (A)<(dir), то (C)←1, інакше (C)←0
Порівняння акумулятора з константою і перехід, якщо вони не дорівнюють одне одному	CJNE A,#d,rel	10	3	2	(PC)←(PC)+3, якщо (A)≠#d, то (PC)← (PC)+rel, якщо (A)<#d, то (C)←1, інакше (C)←0
Порівняння регістра з константою і перехід, якщо вони не дорівнюють одне одному	CJNE Rn,#d,rel	10	3	2	(PC)←(PC)+3, якщо (Rn)≠#d, то (PC)← (PC)+rel, якщо (Rn)<#d, то (C)←1, інакше (C)←0
Порівняння непрямоадресованої комірки ВПД з константою і перехід, якщо вони не дорівнюють одне одному	CJNE @Ri,#d,rel	10	3	2	(PC)←(PC)+3, якщо ((Ri))≠#d, то (PC)← (PC)+rel, якщо ((Ri))<#d, то (C)←1, інакше (C)←0
Довгий виклик підпрограми	LCALL dir16	12	3	2	(PC)← (PC)+3, (SP)←(SP)+1, ((SP))← ($PC_{0..7}$), (SP)← (SP)+1, ((SP))← ($PC_{8..15}$), (PC)← dir 16

Продовження таблиці А.5

Абсолютний виклик підпрограми в межах сторінки в 2 Кбайта	ACALL dir11	6	2	2	(PC)←(PC)+2, (SP)← (SP)+1, ((SP))← ($PC_{0..7}$), (SP)← (SP)+1, ((SP))← ($PC_{8..15}$), (PC0..10)← dir 11
Повернення з підпрограми	RET	1	1	2	(PC8..15)← ((SP)), (SP)←(SP)-1, (PC0..7)← ((SP)), (SP)←(SP)-1
Повернення з підпрограми оброблення переривання	RETI	1	1	2	($PC_{8..15}$)←((SP)), (SP)←(SP)-1, ($PC_{0..7}$)← ((SP)), (SP)←(SP)-1
Порожня команда	NOP	1	1	1	(PC)← (PC)+1

Т – тип команди;
Б – формат у байтах;
Ц – кількість машинних циклів, необхідних для оброблення команди.