МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

з виконання лабораторних робіт дисципліни

«ОСНОВИ МІКРОПРОЦЕСОРНОЇ ТЕХНІКИ»

для студентів спеціальності
141 – Електроенергетика, електротехніка та електромеханіка
освітньої програми «Електротехнічні системи електроспоживання»
денної форми навчання

2020

Методичні вказівки з виконання лабораторних робіт дисципліни «Основи мікропроцесорної техніки» для студентів спеціальності 141 – Електроенергетика, електротехніка та електромеханіка освітньої програми «Електротехнічні системи електроспоживання» денної форми навчання. /Укл: О.С. Назарова - Запоріжжя: НУ «Заорізька політехніка», 2020. – 46 с.

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

Рецензент: А.В. Пирожок, к.т.н., доцент

Відповідальний за випуск: П.Г. Засипко, завідувач лабораторії

Затверджено
на засіданні кафедри
Електропривода і автоматизації
промислових установок
протокол № 12 від 09.07.2019 р.

Рекомендовано
до видання НМК ЕТФ
протокол № 04 від 21.11. 2019 р.

ЗМІСТ

ПЕРЕДМОВА

Методичні вказівки містять опис семи лабораторних робіт з дисципліни «Основи мікропроцесорної техніки» у відповідності до навчальних планів ОКР бакалаврів спеціальності 141 – Електроенергетика, електротехніка та електромеханіка, рекомендації до їх виконання і мають два додатки.

В першій лабораторній роботі наведені короткі відомості про інструментальне програмне забезпечення для розробки та налагодження програм. Решта лабораторних робіт містить короткі теоретичні відомості, індивідуальні завдання та рекомендації щодо їх виконання.

У додатках подано перелік команд мікроконтролера Intel 8051, який включає назву, мнемокод, опис дій, що виконуються, та таблиця представлення даних у двійковій, шістнадцятковій та десятковій системі числення.

Для студентів спеціальності 141 – Електроенергетика, електротехніка та електромеханіка освітньої програми «Електротехнічні системи електроспоживання» денної форми навчання.

1. ЗАГАЛЬНІ ВИМОГИ ДО ОФОРМЛЕННЯ ЗВІТІВ З ЛАБОРАТОРНИХ РОБІТ

Звіт з кожної лабораторної роботи виконується на аркушах формату А4. Текст може бути набраний на комп'ютері у редакторі Word, шрифт 14пт, міжрядковий інтервал 1,5; поля верхнє, нижнє, зліва та зправа по 20 мм.

Роздрукований звіт має бути зброшурований з лівої сторони.

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, необхідні за завданням рисунки і таке інше, висновки з лабораторної роботи.

Зразок оформлення титульної сторінки

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»

Кафедра ЕПА

Звіт з лабораторної роботи №1
дисципліни «Основи мікропроцесорної техніки»

Виконав:
студ. гр. Е-333 Іваненко І.І.

Перевірив:
доцент Петренко П.П.

2019

2. ЛАБОРАТОРНА РОБОТА №1

Ознайомлення із емулятором Franklin Software, вивчення його функціональних можливостей.

Мета: ознайомитись із емулятором Franklin Software, вивчити його функціональні можливості.

Короткі теоретичні відомості.

ProView фірми Franklin Software Inc. – інтегрована середа розробки програмного забезпечення для однокристальних мікроконтролерів сімейства Intel 8051 і його клонів. Вона включає у себе все, що необхідно для створення, редагування, компіляції, трансляції, компонування, завантаження та відлагодження програм. А саме, стандартний інтерфейс Windows; повнофункціональний редактор вихідних текстів з виділенням синтаксичних елементів кольором; організатор проекту; транслятор з мови С; асемблер; відлагоджувач; вбудовану довідкову систему.

Початок роботи.

Запуск інтегрованого середовища ProView32 здійснюється через меню Пуск(Windows):Пуск →Програми → Franklin Software → ProView32.

Далі необхідно створити новий файл. Для цього у розділі меню «File» треба вибрати пункт «New», а також у вікні, що з’явилося, вибрати тип файлу «Assembler Files» (рисунок 2.1).

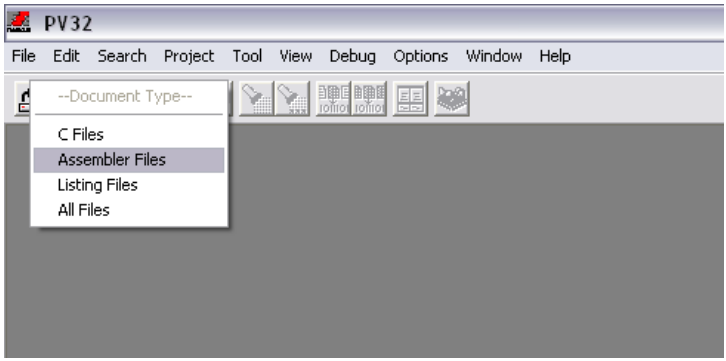


Рисунок 2.1 – Вибір типу файлу.

Новий файл необхідно зберегти на диск (File→Save as…), присвоївши йому ім'я, яке складається з латинських букв та цифр. Довжина імені не повинна перевищувати 8 символів, розширення файла «.asm».

Після цього можна переходити до написання програми для мікроконтролера на мові Асемблер. Згідно завдання та з урахуванням особливостей синтаксису цієї мови складається програма. Після того, як текст програми набрано та збережено, переходимо до компіляції програми – переводу тексту програми у машинний код, який призначено для завантаження у пам'ять програм мікроконтролера.

Для компіляції програми необхідно вибрати пункт меню «Project»→ Build all. Коли процес компіляції завершиться у вікні повідомлень «Message» (рисунок 2.2) буде відображено повідомлення про завершення цього процесу. У випадку виявлення будь-якої помилки, повідомлення про них буде відображено також у цьому вікні.

Рисунок 2.2 – Вікно повідомлень Message.

У разі, якщо програма не містить помилок, можна запускати відлагоджувальник. Для цього у меню «Debug» вибираємо пункт «Start». При першому запуску з'явиться віно (рисунок 2.3), у якому необхідно вказати тип мікроконтролера «Microcontroller: 80c51», а також тактову частоту у мегагерцах.

Запустивши відлагоджувальник з'являються декілька вікон (рисунок 2.4): вікно з текстом програми; вікно коду «Code», у якому кожній інструкції на асемблері відповідає її машинний код; вікно основних регістрів «Main Registers», у якому відображено стан кожного з регістрів. У рядку стану показано час, за який реальний мікроконтролер виконав би команди програми з моменту старту до команди, що виконується саме зараз (виділена синім фоном).

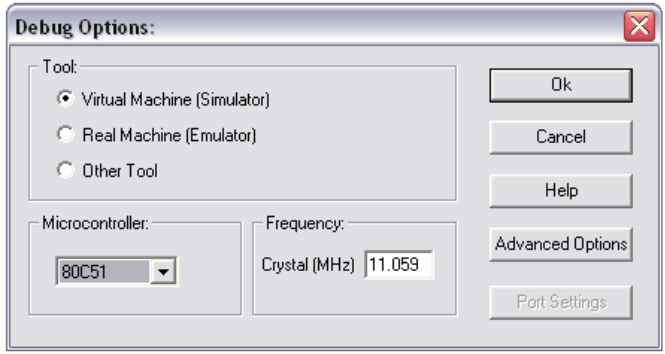


Рисунок 2.3 – Вибір типу контролера та частоти.

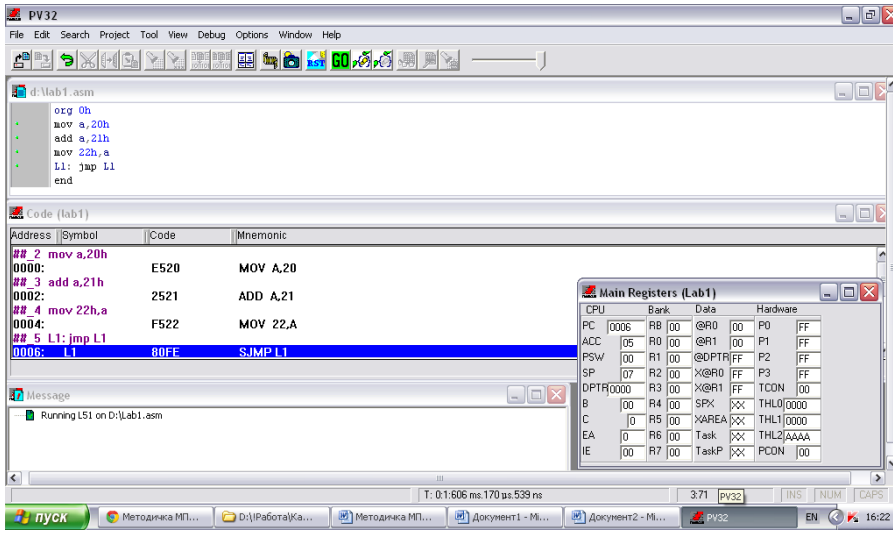


Рисунок 2.4 – Екран режиму «Debug».

Використовуючи меню «View» можна додавати нові вікна. Наприклад, View→Data dump дозволяє відобразити вікно з вмістом різних областей пам’яті даних мікроконтролера: Xdata – зовнішня пам’ять (ОЗП) даних; Data – внутрішня пам’ять даних (внутрішнє ОЗП); Sfr – область регістрів спеціальних функцій (РСФ); Bit – область прямо адресованих бітів. У завданнях необхідно вводити тестові числа у комірки внутрішньої пам’яті даних (ВПД) для перевірки

правильності написаних програм. Для цього виконуємо View→Data dump→ Data view. У вікні «Data view» (рисунок 2.5) відображається вміст комірок ВПД, який згруповано по 8 комірок у рядку із зазначенням першої комірки у групі (адреса вказана зліва з двокрапкою). Після введення даних у комірку треба натиснути кнопку «ENTER» на клавіатурі.

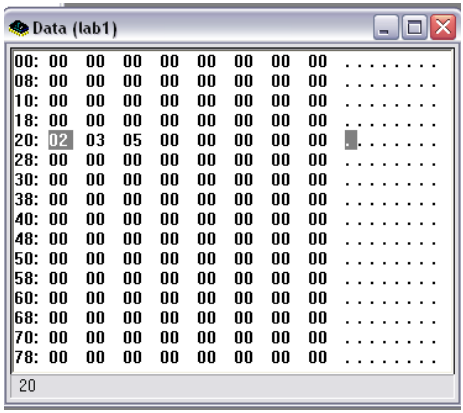


Рисунок 2.5 – Вікно відображення вмісту комірок ВПД «Data view».

Перед запуском програми необхідно натиснути кнопку «Animate»  , потім кнопку скидання «Reset» 

і запустити програму кнопкою «Run» 

.

Для перевірки отриманого після виконання програми результату відкриваємо калькулятор Windows: Пуск → Програми → Стандартні → Калькулятор. Переводимо його у інженерний режим (рисунок 2.6). Вмикаємо режим роботи у шістнадцятковій системі числення (перемикач «Hex») та обмежуємо розмір чисел одним байтом (перемикач «1 байт»).

Програму перевіряємо у відповідності до завдання з урахуванням пріоритетів. Результат має співпасти з тим, що виведений у комірку ВПД згідно завдання. У випадку, коли результат не співпадає, необхідно з'ясувати причину помилки. Відлагодження програми у цьому випадку зручно проводити у покроковому режимі (кнопка «Step into»  або F7).

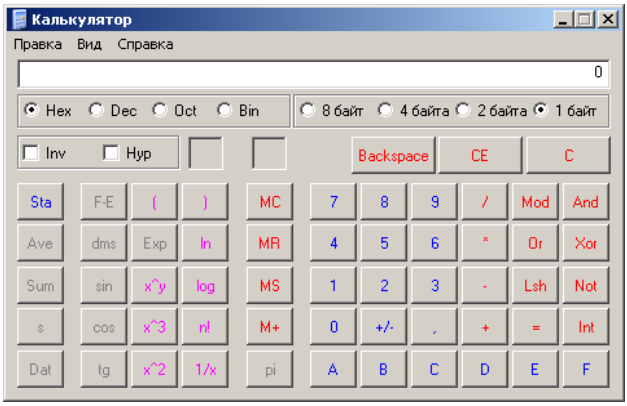


Рисунок 2.6 – Вікно калькулятора Windows.

У цьому разі при кожному натисненні кнопки буде виконуватись тільки одна інструкція на мові асемблер. При цьому у відкритих вікнах з регістрами і вмістом пам'яті буде оновлюватися інформація у відповідності з ходом виконання програми. Порівнюючи результати виконання кроку у вікні «Main Registers» з результатами проміжних обчислень знаходимо помилки у програмі. Якщо під час відлагодження програми з'ясувалося, що алгоритм необхідно скорегувати, то треба перервати процес відлагодження, вибравши пункт меню Debug→Terminate. Після внесення необхідних змін у програму її потрібно повторно компілювати (Project → Build all) і повторити процес відлагодження.

Якщо програма занадто велика і необхідно відлагодити тільки деякі її фрагменти, з метою економії часу, використовують крапки зупинки «Breakpoint», які встановлюються на початку фрагменту, що треба виправити. При цьому перехід до крапки зупинки відбувається натисненням кнопки «Go». Далі відлагодження відбувається у покроковому режимі як було описано раніше.

Основні правила синтаксису мови Асемблер.

1. Кожна команда пишеться у окремому рядку

2. Кожний рядок розбивається на чотири поля, які утворюють чотири колонки. Укожного поля своє призначення:

- перше поле – поле мітки;
- друге поле – поле команди (мнемокода);
- третє поле – поле параметрів команди;

- четверте поле - поле коментарів команди, заповнювати його необов'язково. Воно може займати місце до кінця сторінки. Для відокремлення коментаря від параметрів команди необхідно ставити знак «;». Информація після «;» не обробляється транслятором, що також зручно використовувати при відлагоджуванні програми для тимчасового виключення з неї команди (досить поставити перед командою «;»).

Завдання.

Необхідно внести зміни у готову програму додавання вмісту двох комірок у відповідності з індивідуальним завданням (табл. 2.1) та отримати результат у вказаній комірці.

Перевірити правильність роботи зміненої програми шляхом письмового розрахунку у двійковій системі числення.

Таблиця 2.1 – Дані для виконання індивідуального завдання

№	Завдання	Дані для перевірки	
1	(22H)+(24H)→(23H)	A9	D1
2	(21H)+(20H)→(25H)	BC	DA
3	(23H)+(21H)→(22H)	D0	A6
4	(20H)+(24H)→(25H)	DE	B3
5	(21H)+(25H)→(24H)	DD	DC
6	(23H)+(22H)→(27H)	FA	AB
7	(27H)+(21H)→(20H)	E1	D9
8	(24H)+(22H)→(26H)	DC	B4
9	(25H)+(21H)→(22H)	A5	B4
10	(28H)+(20H)→(27H)	CD	CA
11	(23H)+(21H)→(22H)	E5	C2
12	(28H)+(27H)→(26H)	DD	BD
13	(26H)+(25H)→(20H)	B9	E6
14	(21H)+(23H)→(20H)	EC	D4
15	(24H)+(23H)→(29H)	C3	BD
16	(28H)+(26H)→(25H)	EE	EB
17	(25H)+(21H)→(23H)	D3	FB
18	(22H)+(21H)→(27H)	F1	AF
19	(20H)+(25H)→(28H)	B6	BF
20	(24H)+(23H)→(29H)	F0	C8

Продовження таблиці 2.1.

21	(22H)+(21H)→(24H)	C8	D5
22	(23H)+(27H)→(25H)	E0	C3
23	(28H)+(24H)→(20H)	CB	F1
24	(25H)+(23H)→(22H)	DB	CF
25	(26H)+(23H)→(28H)	F3	DB
26	(25H)+(26H)→(29H)	EF	E5
27	(22H)+(21H)→(27H)	D3	CC
28	(25H)+(21H)→(23H)	FE	FC
29	(21H)+(25H)→(24H)	E5	DA
30	(22H)+(23H)→(27H)	C9	DF

Приклад письмового розрахунку у двійковій системі числення:
0AAH → 1010 1010B
+
055H → 0101 0101B

111111111B → 0FFH

Зміст звіту з лабораторної роботи.

Зміст 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.

Контрольні запитання.

1. Призначення емулятора.
2. Створення нового файлу.
3. Трансляція тексту програми.
4. Редагування тексту програми.
5. Компіляція програми.
6. Основні правила синтаксису команд Асемблер.
7. Введення даних у комірки пам’яті ОЗП.

3. ЛАБОРАТОРНА РОБОТА №2

Вивчення команд логічних операцій.

Мета: здобути навички використання команд логічних операцій.

Короткі теоретичні відомості.

Логічні команди (Таблиця А.3 додатку А) реалізують операції булевої алгебри «І» (AND), «АБО» (OR), «АБО, що виключає» (XOR), побітне зрушення вмісту акумулятора ліворуч або праворуч, інверсію та очищення акумулятора, а також перестановку його тетрад.

Завдання

Написати програму для вирішення індивідуального завдання згідно варіанту (табл. 3.1).

Таблиця 3.1 – Дані для виконання індивідуального завдання

№	Вираз	Дані для перевірки
1.	$(20H)*02H+8AH\wedge\overline{(21H)}\forall(23H)\to(22H)$	12H; 0A2H; 1BH
2.	$(\overline{(20H)-8AH})/02H\vee 3FH\forall(21H)\to(23H)$	0FCH; 88H
3.	$(21H)\vee(22H)+\overline{10H}\forall(23H)/02H\to(24H)$	43H; 56H; 78H
4.	$21H*(21H)\forall(22H)+\overline{10H\wedge 35H}\to(23H)$	12H; 53H
5.	$\overline{\overline{(20H)}\wedge(21H)}-7BH\forall 38H*02H\to(22H)$	11H; 05H
6.	$(21H)\vee\overline{(82H/(22H)+\overline{\overline{58H}}\forall(23H)\to(24H)}$	3BH; 02H; 0CDH
7.	$(22H)\forall(23H)-\overline{21H}*03H\wedge 5DH\to(24H)$	4BH; 58H
8.	$4AH\forall\overline{05H}*\overline{(21H)\vee(22H)}-15H\to(23H)$	87H; 23H
9.	$10H\vee\overline{((21H)+(22H)\forall 0C2H)}/02H\to(23H)$	35H; 2DH
10.	$28H+(21H)*\overline{0AAH}\forall\overline{11H\vee(22H)}\to(23H)$	03H; 73H
11.	$\overline{(23H)}-31H\forall(21H)\wedge 25H/03H\to(22H)$	30H; 1CH
12.	$(21H)\wedge(22H)+\overline{(23H)}\forall 0ACH*02H\to(24H)$	72H; 0ADH; 12H

Продовження таблиці 3.1

13.	$\overline{(((20H)+21H)*02H \forall (21H)) \wedge 54H} \rightarrow (22H)$	43H; 0CDH
14.	$15H*(20H)+\overline{35H} \forall (21H) \vee (23H) \rightarrow (24H)$	03H; FAH; 93H
15.	$25H/02H \wedge \overline{\overline{(21H)} \forall (22H)-11H} \rightarrow (23H)$	32H; 0CCH
16.	$\overline{(10H-08H} \forall (21H)*02H) \vee (22H) \rightarrow (23H)$	05H; 1FH
17.	$(20H) \forall \overline{(22H)}-21H \wedge 0ABH/02H \rightarrow (23H)$	2BH; 0FDH
18.	$\overline{(20H) \forall (21H)-04H} \vee 0CAH/02H \rightarrow (22H)$	3AH; 49H
19.	$\overline{(20H)+23H} \wedge (22H)*02H \forall 0CDH \rightarrow (21H)$	1AH; 08H
20.	$10H \vee (21H)+\overline{(22H)*02H \forall 0CDH} \rightarrow (23H)$	3FH; 23H
21.	$(22H)-\overline{(21H)} \wedge 7BH/03H \forall 0FFH \rightarrow (23H)$	0F4H; 8EH
22.	$(20H)/03H+\overline{\overline{(21H)}} \forall 12H \vee 0CEH \rightarrow (22H)$	09H; 0EFH
23.	$\overline{(21H)-0D2H \forall (22H)} \wedge 0FDH*02H \rightarrow (23H)$	7EH; 19H
24.	$(20H) \vee \overline{(21H)}+ABH \forall CEH-\overline{0FFH} \rightarrow (22H)$	7CH; 63H
25.	$(\overline{32H-(20H)*02H}) \wedge (21H) \forall 0CDH \rightarrow (22H)$	04H; 0B7H
26.	$(0FFH-\overline{(21H)} \forall 89H) \vee \overline{(22H)/03H} \rightarrow (23H)$	73H; 09H
27.	$14H \forall 0DFH+\overline{\overline{(21H) \wedge 18H)}*02H} \rightarrow (22H)$	34H
28.	$(21H) \vee 0C6H-03H \forall (22H)/02H \rightarrow (23H)$	0ADH; 12H
29.	$(\overline{10H+(23H)} \forall \overline{(24H)}*02H) \wedge 58H \rightarrow (25H)$	67H; 0A5H
30.	$((24H) \wedge (\overline{25H})-10H) \forall \overline{0DDH/02H} \rightarrow (26H)$	98H; 35H

Дані для перевірки підставляються послідовно у відповідні комірки внутрішньої пам'яті даних згідно індивідуального завдання.

Приклад:

$(20H)*\overline{(21H)-10H} \rightarrow (23H)$

```
ORG 0H
MOV A, 21H          ;(21H)→(A)
```

```
SUBB A, #10H        ;(A)-10H-(C)→(A)
CPL A               ; (A̅) → (A)
MOV B, 20H          ;(20H)→(B)
MUL AB              ;(A)*(B) →(B).(A)
MOV 23H, A          ;(A) →(23H)
L1: JMP L1
END
```

Дані для перевірки слід підставляти у комірки (20H), (21H) і виконувати дії на калькуляторі з урахування пріоритетів виконання дій.

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.

Контрольні запитання.

1. Назвати логічні операції.
2. Логічне множення. Навести приклади команд з різними методами адресації.
3. Логічне додавання. Навести приклади команд з різними методами адресації.
4. Логічне «АБО, що виключає». Навести приклади команд з різними методами адресації.
5. Логічне заперечення (інверсія) та його команда.
6. Маскування.

4. ЛАБОРАТОРНА РОБОТА №3

Вивчення команд роботи з бітами

Мета. Здобути навички використання команд роботи з бітами.

Короткі теоретичні відомості.

Команди роботи з бітами (Таблиця А.4 додатку А) забезпечують встановлення (запис одиниці), скидання або очищення (запис нуля) та інвертування бітів внутрішньої пам'яті, виконують операції булевої алгебри «І» (AND) та «АБО» (OR) з прапором переносу та бітами внутрішньої пам'яті, здійснюють пересилання між прапором переносу і бітом внутрішньої пам'яті. Адреса біту може бути вказана тільки прямим способом.

Завдання.

Згідно заданої функції алгебри логіки (табл. 4.1)скласти:

- комбінаційну схему;
- електричну схему;
- таблицю істинності для всіх можливих комбінацій бітів порта Р2;
- написати програму.

Таблиця 4.1 - Дані для виконання індивідуального завдання

№	Завдання
1.	$\overline{\overline{P2.1 \vee \overline{P2.2}} \wedge P2.3} \rightarrow P0.0$
2.	$\overline{P2.1} \wedge P2.2 \vee \overline{P2.3} \rightarrow P0.0$
3.	$\overline{\overline{P2.1 \wedge P2.2} \vee P2.3} \rightarrow P0.0$
4.	$\overline{P2.4 \wedge P2.3} \vee \overline{\overline{P2.2}} \rightarrow P0.0$
5.	$\overline{P2.4} \vee \overline{P2.3} \wedge P2.2 \rightarrow P0.0$
6.	$\overline{P2.4} \wedge \overline{P2.3 \vee P2.2} \rightarrow P0.0$
7.	$\overline{P2.2 \vee \overline{P2.3}} \wedge P2.4 \rightarrow P0.0$
8.	$\overline{P2.3 \wedge \overline{P2.2 \vee P2.4}} \rightarrow P0.0$

Продовження таблиці 4.1

9.	$\overline{\overline{\overline{P2.1} \vee P2.2} \wedge P2.3} \rightarrow P0.0$
10.	$\overline{P2.0 \vee \overline{\overline{P2.7}} \wedge P2.6} \rightarrow P0.0$
11.	$\overline{\overline{P2.0} \wedge P2.7 \vee P2.6} \rightarrow P0.0$
12.	$P2.0 \vee \overline{\overline{P2.7} \wedge P2.6} \rightarrow P0.0$
13.	$P2.0 \wedge \overline{P2.7 \vee \overline{P2.6}} \rightarrow P0.0$
14.	$\overline{P2.5 \vee \overline{P2.4 \wedge P2.3}} \rightarrow P0.0$
15.	$\overline{P2.5} \wedge \overline{P2.4 \vee P2.3} \rightarrow P0.0$
16.	$\overline{P2.5 \vee P2.3} \wedge \overline{P2.4} \rightarrow P0.0$
17.	$P2.1 \vee \overline{P2.2} \wedge \overline{P2.3} \rightarrow P0.0$
18.	$\overline{P2.1} \vee \overline{P2.2} \wedge P2.3 \rightarrow P0.0$
19.	$\overline{P2.1} \vee P2.2 \wedge \overline{P2.3} \rightarrow P0.0$
20.	$P2.4 \wedge \overline{P2.2} \vee \overline{P2.3} \rightarrow P0.0$
21.	$\overline{P2.4} \wedge P2.2 \vee \overline{P2.3} \rightarrow P0.0$
22.	$\overline{P2.0 \wedge P2.7} \vee \overline{P2.6} \rightarrow P0.0$
23.	$P2.0 \wedge \overline{P2.7 \vee \overline{P2.6}} \rightarrow P0.0$
24.	$\overline{P2.7 \wedge P2.5 \vee \overline{P2.3}} \rightarrow P0.0$
25.	$P2.7 \wedge \overline{\overline{P2.5} \vee P2.3} \rightarrow P0.0$
26.	$\overline{P2.6} \vee P2.4 \wedge \overline{P2.2} \rightarrow P0.0$
27.	$\overline{P2.6 \vee P2.4 \wedge \overline{P2.2}} \rightarrow P0.0$
28.	$\overline{P2.6 \vee \overline{P2.4}} \wedge P2.2 \rightarrow P0.0$
29.	$\overline{P2.6 \wedge P2.2} \vee \overline{P2.4} \rightarrow P0.0$
30.	$\overline{\overline{P2.1 \wedge P2.7} \vee P2.3} \rightarrow P0.0$

Зміст звіту з лабораторної роботи.

Зміст з лабораторної роботи повинен містити:

- титульний лист, тему, мету лабораторної роботи;
- умову індивідуального завдання;
- комбінаційну схему;
- електричну схему;
- таблицю істинності;
- текст програми з коментарями до кожної команди (групи команд), які реалізують виконання однієї з умовних частин всієї програми;
- висновки з лабораторної роботи.

Контрольні запитання.

1. Назвати команди пересилання при роботі з бітами.
2. Назвати команди логічних операцій при роботі з бітами.
3. Назвати команди установки та очищення біта.
4. Призначення портів вводу/виводу інформації.
5. Скласти таблицю істиності та комбінаційну схему за функцією алгебри логіки згідно завдання викладача.
6. Пояснити принцип складання електричних схем за функцією алгебри логіки.

5. ЛАБОРАТОРНА РОБОТА №4

Розроблення програм розгалуджених структур. Організація циклів.

Мета. Здобути навички розроблення програм розгалужених структур та організації циклів.

Короткі теоретичні відомості.

Команди передачі керування (Таблиця А.5 додатку А) забезпечують безумовні та умовні переходи у програмі. Розгалудження в програмі реалізуються за допомогою команд умовних переходів, причому, якщо умова виконується, програма переходить за адресою, що вказана у команді умовного переходу. При невиконанні умови виконується наступна команда.

Завдання.

Написати програму для формування послідовності коротких та довгих імпульсів на вказаному виході порта 0 при низькому рівні сигналу вказаного біта порта 2 згідно індивідуального завдання (табл. 5.1), де «*» - короткий імпульс; «-» - довгий імпульс.

Таблиця 5.1 - Дані для виконання індивідуального завдання

№	Біт P2	Біт P0	Умовне позначення послідовності імпульсів
1.	P2.0	P0.5	* * - -* * *- - - -
2.	P2.1	P0.4	* * - - -* * * - -
3.	P2.2	P0.3	* * - - - - * * * * - - -
4.	P2.3	P0.2	* * * - -* * * - - -
5.	P2.4	P0.1	* * * - - - * * - - -
6.	P2.5	P0.0	* * * - - - - * * - - -
7.	P2.6	P0.7	* * * * - -* * * - -
8.	P2.7	P0.6	* * * * - - -* * - - - -
9.	P2.0	P0.5	* * * * - - - -* * - -
10.	P2.1	P0.4	- - * * - - - - * * *
11.	P2.2	P0.3	- - - * * - - - - * *
12.	P2.3	P0.2	- - - - * *- - - * * *
13.	P2.4	P0.1	- - * * * - - - - * *

Продовження таблиці 5.1

14.	P2.5	P0.0	- - - * * *- - - - * *
15.	P2.6	P0.7	- - - - * * * - - * *
16.	P2.7	P0.6	- - * * * *- - - - * * *
17.	P2.0	P0.5	- - - * * * * *- - * * *
18.	P2.1	P0.4	- - - - * * * * *- - * *
19.	P2.2	P0.3	* * - -* * * - -
20.	P2.3	P0.2	* * - - -* * * - - - -
21.	P2.4	P0.1	* * - - - - * * * - -
22.	P2.5	P0.0	* * * - -* * * * - - -
23.	P2.6	P0.7	* * * - - - * * - -
24.	P2.7	P0.6	* * * - - - - * * - - -
25.	P2.0	P0.5	- - * *- - - - * * * *
26.	P2.1	P0.4	- - - * * - - - - * * *
27.	P2.2	P0.3	- - - - * *- - - * * *
28.	P2.3	P0.2	- - * * *- - - - * * * *
29.	P2.4	P0.1	- - - * * *- - - - * * * *
30.	P2.5	P0.0	- - - - * * * - - * *

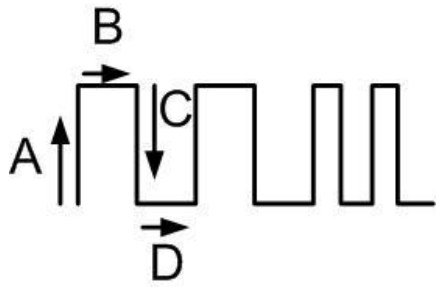


Рисунок 5.1 – Умовне зображення послідовності імпульсів

Приклад фрагменту програми.

```
        ORG 0H
        SETB P2.0
        CLR P0.0                ;
M1:     JB P2.0, M1             ;очікування низького рівня
        MOV R2, #02H            ;задання кількості импульсів
```

```
-----------------------------------------------
M2:     SETB P0.0                   ;A
-----------------------------------------------
        MOV R1, #10H
M3:     DJNZ R1, M3                 ;B
-----------------------------------------------
        CLR P0.0                    ;C
-----------------------------------------------
        MOV R1, #10H
M4:     DJNZ R1, M4                 ;D
-----------------------------------------------
        DJNZ R2, M2
```

І так далі…

```
M100:  JNB P2.0, M100;              очікування високого рівня

       JMP M1
       END
```

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.

Контрольні запитання.

1. Дати визначення поняттю «розгалужена структура».
2. Пояснити процедуру організації циклу.
3. Назвати команди безумовного переходу.
4. Назвати команди умовного переходу.

6. ЛАБОРАТОРНА РОБОТА №5

Вивчення команд непрямої адресації. Робота із масивами даних.

Мета. Здобути навички використання команд непрямої адресації при роботі з масивами даних.

6.1. Короткі теоретичні відомості.

Непряма реєстрова адресація використовується для звертання до комірок внутрішньої пам'яті даних, причому у якості адреси комірки, у якій знаходиться операнд, використовується вміст одного з індексних регістрів R0, R1. Ознакою непрямої регістрової адресації є наявність у команді знаку «@».

6.2. Завдання.

Зі всіма елементами початкового масиву виконати вказані дії і сформувати кінцевий масив згідно індивідуального завдання (табл.6.1).

Таблиця 6.1 - Дані для виконання індивідуального завдання

№	Початкова адреса початкового масиву	Елементи початкового масиву	Дії	Початкова адреса кінцевого масиву
1.	2DH	E6, BE, BE, C3, FF, DB, D4, C2, BA, 0	(...+ 49h)∀68h	5BH
2.	36H	97, AF, AF, B6, F2, C7, AF, C4, AB, F1	(...+ 77h)∀49h	68H
3.	2FH	B4, DC, DC, D7, 9B, D7, DA, D2, 9A	(...+ 1Ch)∀97h	5CH
4.	2BH	C7, DA, E3, EE, 15, E8, DA, E1, E1, 16	(...+ CDh)∀C2h	62H
5.	21H	7B, 60, 60, 9C, 4A, 63, 55, 64, 48, 9B	(...+ BBh)∀77h	64H
6.	39H	14, 2C, 2C, 33, 6F, 26, 33, 32, 2E, 6E	(...+ 7Ah)∀C9h	69H
7.	24H	49, 3D, 34, 43, 3E, 75, 2C, 42, 28, 74	(...+ DCh)∀71h	6AH

Продовження таблиці 6.1

8.	22H	2C, 38, 41, 3D, 4B, 41, 7C, 4A, 41, 4C, 41, 3D, 50	(...+ 6Eh)∀CAh	59H
9.	34H	8C, A9, A5, 9D, 60, A9, AF, 60, A3, AA, 9D, AE	(...+ 46h)∀86h	62H
10.	29H	7D, 5F, 53, 5B, 20, 57, 71, 20, 55, 6E, 5B, 72	(...+ 6Dh)∀ADh	6FH
11.	3DH	0, E5, D6, D9, 9C, E5, CB, 9C, D7, DF, DF, D8	(...+ 7Ah)∀36h	69H
12.	39H	57, 65, 78, 74, 70, 66, B3, 76, 72, 70, 78, B3, 67, 65, 68, 78	(...+ 45h)∀D8h	6AH
13.	2DH	71, 9F, 85, DA, 99, 8C, 95, DA, 9B, 9E, 95, 88, 95, 8C, D9	(...+ 1Bh)∀D5h	65H
14.	34H	81, 93, 9D, CA, 89, 98, 8D, CA, 96, 9D, 87, 8F, A1, C9	(...+ 99h)∀43h	69H
15.	34H	A1, CB, C5, 18, D9, C6, D5, 18, C3, D1, CA, CA, D5, C6, 19	(...+ E6h)∀DEh	5EH
16.	28H	C, 27, 25, 2D, 16, 31, 1A, 67	(...+ 8Ch)∀D2h	66H
17.	23H	92, 84, 84, 3F, 68, 7E, 74, 3F, 7B, 80, 73, 84, 71	(...+ D9h)∀38h	6EH
18.	2FH	3C, 6, 20, D5, 23, 10, 14, 11, D5, 8, 1C, D5, 8, C, 7, 11, 22	(...+ BAh)∀AFh	70H
19.	26H	1D, E1, E8, FD, A9, E0, FA, A9, FB, E0, EE, E1, FD, A8	(...+ 60h)∀29h	68H

Продовження таблиці 6.1

20.	3AH	DE, AA, B3, BE, 72, AB, C1, 72, AC, AB, A4, AF, 73	(…+ 9Ch)∀2Eh	6DH
21.	20H	4A, 3C, 33, 36, 32, 34, 3C, 77, 4F, 32, 34, 3C, 78	(...+ 63h)∀FAh	5FH
22.	23H	FE, F0, E9, EA, E6, E8, F0, 2D, E1, E6, 2D, 10, FD, C, 13, 24	(...+ AEh)∀FBh	58H
23.	2FH	E0, D2, C9, D4, C8, CA, D2, 95, C1, C8, 95, DB, E7, E1, E2	(...+ 89h)∀3Eh	61H
24.	40H	59, C2, 81, 87, 94, 85, 85, C2, 97, 79, 96, 7A, C2, 89, 7F, 95	(...+ A7h)∀49h	60H
25.	35H	25, 84, 47, 39, 34, 84, 53, 4A, 84, 47, 53, 4A, 4A, 49, 49	(...+ 6Ch)∀D0h	55H
26.	25H	AB, 4C, 8D, 97, 9C, 4C, 91, 8A, 4C, 4C, 98, 87, 87	(...+ B9h)∀25h	68H
27.	23H	BD, E1, EE, 29, DC, DE, E7, 29, DC, E1, E2, E7, EE, DC	(...+ CFh)∀D8h	6BH
28.	2BH	FC, D5, C4, C4, CD, 14, FE, D9, C7, 14, ED, D9, D5, C2, 15	(...+ 3Eh)∀72h	60H
29.	28H	22, 36, 27, 27, 2E, 75, 18, 3D, 27, 3E, 28, 29, 42, 36, 28, 76	(...+ 7Bh)∀D0h	5DH
30.	32H	4F, 36, 71, 30, 32, 3F, 36, 33, 46, 2D, 72	(...+ 59h)∀EAh	58H
31.	3DH	2C, 59, 59, 48, 4F, 59, 54, 4E, 4F, 8D, 5D, 51, 48, 4C, 5A, 48, 8C	(...+ DAh)∀47h	6EH

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.

Контрольні запитання.

1. Дати визначення методу непрямої адресації.
2. Навести приклади команд непрямої адресації.
3. Пояснити доцільність застосування непрямої адресації.
4. Дати визначення поняттю «масив даних».
5. Команди пересилання даних з використанням непрямої адресації.
6. Команди арифметичних операцій з використанням непрямої адресації.
7. Команди логічних операцій з використанням непрямої адресації.
8. Команди передачі керування з використанням непрямої адресації.

7. ЛАБОРАТОРНА РОБОТА №6

Паралельне введення-виведення інформації

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

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 7.1).

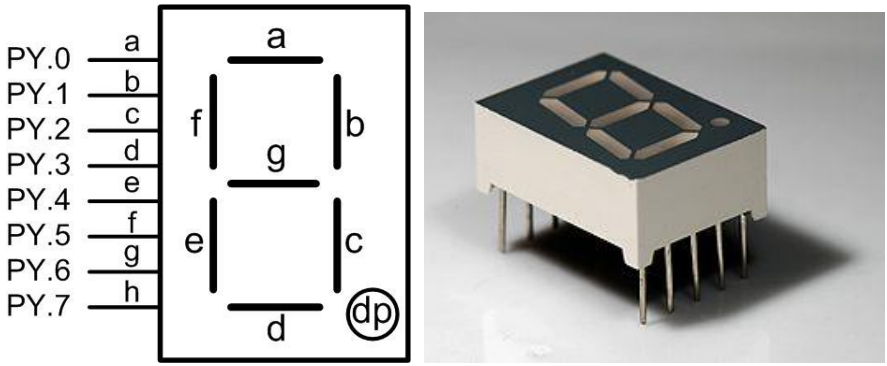


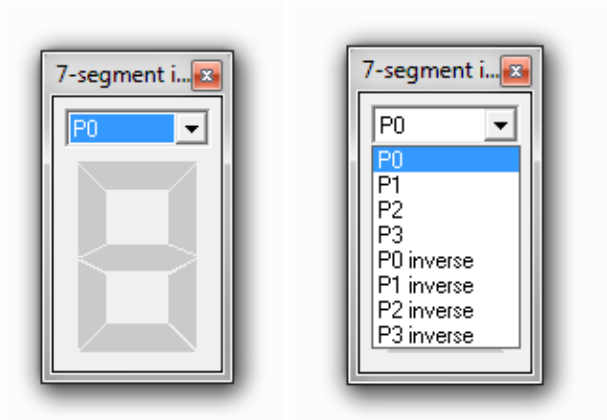
Рисунок 7.1 - Семисегментий індикатор

Наприклад, для того, щоб на семисегментному індикаторі з'явилась цифра «5», треба в один із чотирьох портів МК вивести двійкове число «01101101b» (таблиця 7.1).

Таблиця 7.1 – 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 7.2, а. Зі списку, що відкривається вибирається потрібний порт (рисунок 7.2, б).



а б

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

Рисунок 7.2 – Вікно емулятора CCI

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

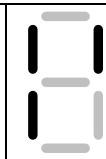
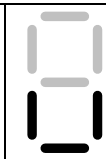
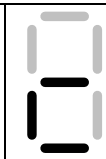
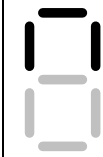
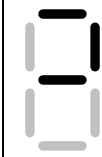
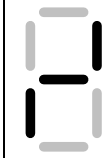
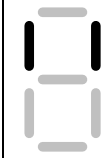
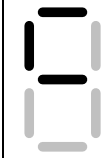
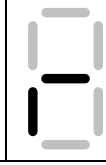
Завдання.

Написати програму для індикації знаків на семисегментному індикаторі згідно варіанта індивідуального завдання (таблиця 7.2), використавши для цього порти P0 і P1.

Таблиця 7.2 – 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 7.2

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

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.

Контрольні запитання.

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

8. ЛАБОРАТОРНА РОБОТА №7

Програмування мікроконтролера ADuC 841.

Мета. Здобуття навичок видачі сигналів керування на зовнішні пристрої.

Короткі теоретичні відомості.

Порядок запису розробленої програми в пульт.

1. Підключити пульт до USB.
2. Запустити програму «WDS Analog Devices» (рисунок 8.1).

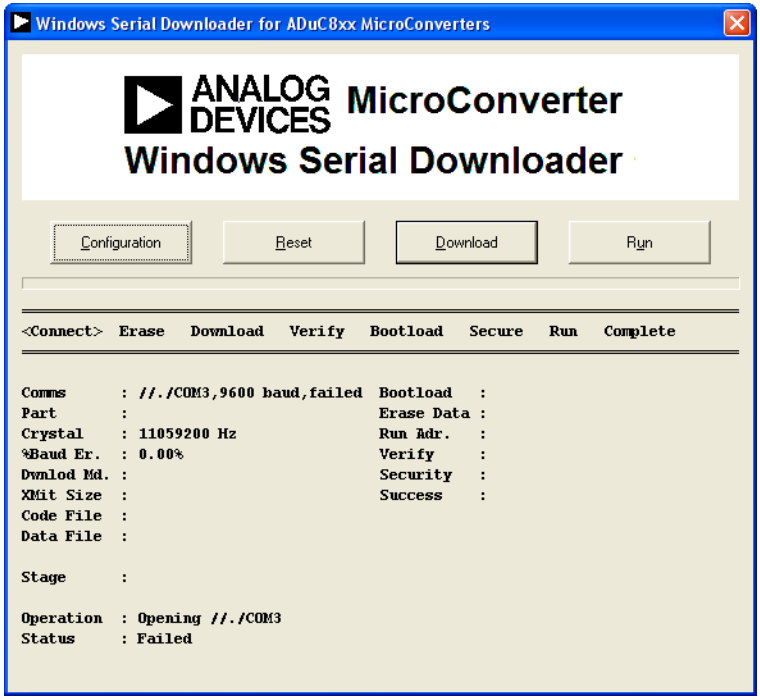


Рисунок 8.1 – Вікно програми «WDS Analog Devices».

3. Натиснути кнопку «Configuration» цього вікна, яка відкриє нове вікно «Configuration» (рисунок 8.2)

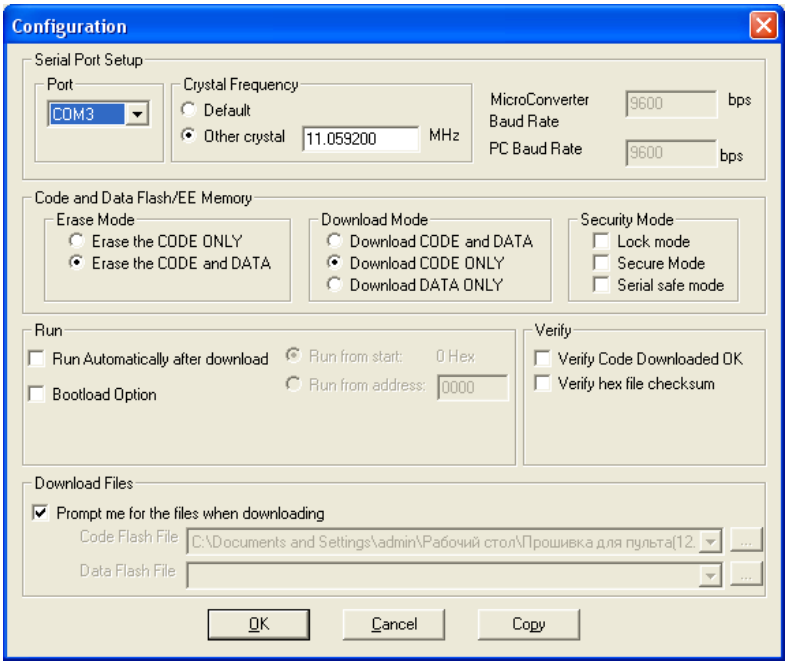


Рисунок 8.2 – Вікно «Configuration».

4. В області «Crystal Frequency» вибрати опцію «Other crustal» з частотою 11,0592 MHz, далі натиснути кнопку «OK».

5. Для встановлення у режим програмування на задній стороні пульта (рисунок 8.3) треба переключити повзунковий перемикач у положення «P» і натиснути кнопку скидання «R», після чого пульт перейде у режим програмування.

6. Далі у вікні програми WDS треба натиснути кнопку скидання «Reset», після чого «Status» повинен змінитися на «Reset OK! ADuC841 device detected» (рисунок 8.4)

7. Якщо з’явиться повідомлення «Status – Failed to connect to the ADuC8xx after 2 attempts», необхідно повторити пункт 6.

8. Далі у вікні програми WDS натиснути кнопку «Download» після чого відкриється вікно для вибору прошивки, вибрати потрібний файл і натиснути кнопку «Открыть», після цього почнеться завантаження програми у пульт.

Рисунок 8.3 – Задня сторона пульта.

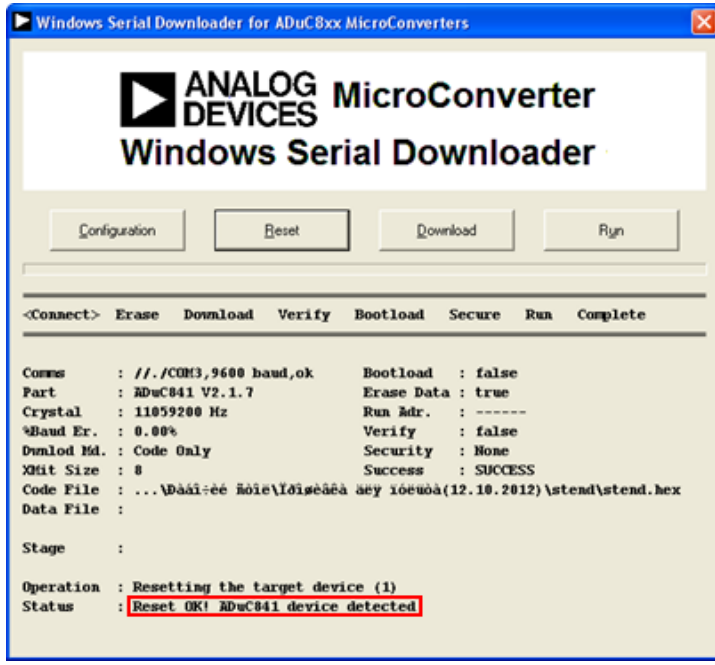


Рисунок 8.4 - Вікно програми WDS після виконання скидання.

9. Після завершення завантаження у «Status» відобразиться параметр «Success».

10. Далі для запуска програми в пульті необхідно натиснути на кнопку «Run» у вікні програми WDS або перемикнути повзунковий перемикач на задній панелі у положення «W» і натиснути кнопку скидання «R».

Завдання.

Розробити программу для сворення ефекту «вогні, що біжать» для бітів порту P0 (напрямок руху повинен залежати від стану біту (задається викладачем) порту P2).

Провести тестування програми у середовищі Franklin Software.

Адаптувати розроблену программу до мікроконтролера.

Записати програму у мікроконтролер.

Зміст звіту з лабораторної роботи.

Зміст 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.

Контрольні запитання.

1. Пояснити принцип створення ефекту «вогні, що біжать».
2. Пояснити яким чином організовано переміщення вліву або праву сторону цих «вогнів».
3. Назвати перелік дій, які необхідно виконати, для того, щоб завантажити розроблену програму у мікроконтролер ADuC 841.
4. Дати визначення поняттю «зовнішній пристрій».
5. Дати визначення поняттю «сигнал керування».

ПЕРЕЛІК ПОСИЛАНЬ

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

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

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

4. Карташов, Б. А. Системы автоматического регулирования с микроЭВМ / Б.А. Карташов, Е.А. Шабаев – Зерноград: АЧГАА, 2008. – 46 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. Ишматов, З. Ш. Микропроцессорное управление електроприводами и технологическими объектами. Полиномиальные методы. – Екатеринбург, УГТУ-УПИ, 2007. – 278 с.

8. Магда, Ю. С. Микроконтроллеры серии 8051: практический поход. – М. ДМК Пресс, 2008. – 228 с.

9. Горюнов А. Г., Ливенцов С.Н. Архитектура микроконтроллера INTEL 8051: Учебное пособие. – Томск, Изд-во ТПУ, 2005. – 86 с.

10. Веприк, В.Н. и др. Микроконтроллеры семейства MCS-51: Учебное пособие / В.Н. Веприк, В.А. Афанасьев, А.И. Дружинин, А.А. Земсков, А.Р. Исаев, О.В. – Новосибирск, 1997. – 62 с.

11. Осадчий, В. В. Лабораторный стенд для исследования алгоритмов микропроцессорных систем управления шаговыми двигателями / В. В. Осадчий, Е. С. Назарова, С. Ю. Тоболкин // Електромеханічні і енергозберігаючі системи. – 2014. – Вип. 2/(26). – С.102-108.

12. Осадчий В. В. Дослідження позиційного електропривода двомасової системи з внутрішнім слідкуючим контуром / В. В. Осадчий, О. С. Назарова, М. О. Олєйніков // Вісник НТУ «ХПІ» - Харків, 2019. – С.47-54.

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	$(A_{0\text{-}3})\leftarrow(A_{4\text{-}7})$

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

Додаток Б

Представлення даних у двійковій, шістнадцятковій та десятковій системі числення

Таблиця Б.1 – Відповідність представлення даних у різних системах числення

BIN (двійкова)				HEX (шістнадцяткова)	DEC (десяткова)
$2^3 = 8$ ст.біт	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$ мол.біт		
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	2	2
0	0	1	1	3	3
0	1	0	0	4	4
0	1	0	1	5	5
0	1	1	0	6	6
0	1	1	1	7	7
1	0	0	0	8	8
1	0	0	1	9	9
1	0	1	0	A	10
1	0	1	1	B	11
1	1	0	0	C	12
1	1	0	1	D	13
1	1	1	0	E	14
1	1	1	1	F	15