Міністерство освіти і науки України
Національний гірничий університет

П.І.Б. _____
Група _____
Варіант _____
Відмітка про залік:
_____

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторної роботи МПП-5
“Асемблер MCS-51. НАВЧАЛЬНО-НАЛАГОДЖУВАЛЬНИЙ СТЕНД EV8031/AVR”,
індивідуальних завдань та самостійної роботи
з професійно-орієнтованої дисципліни
“Мікропроцесорні пристрої”

для студентів спеціальності 7.092203 “Електромеханічні системи автоматизації та електропривод” напряму “Електромеханіка”

Дніпропетровськ
2006

Міністерство освіти і науки України
Національний гірничий університет

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторної роботи МПП-5 “Асемблер MCS-51. Навчально-налагоджувальний стенд EV8031/AVR”, індивідуальних завдань та самостійної роботи з професійно-орієнтованої дисципліни “Мікропроцесорні пристрої” для студентів спеціальності 7.092203 “Електромеханічні системи автоматизації та електропривод” напряму "Електромеханіка”

Дніпропетровськ, НГУ
2006

Методичні вказівки до виконання лабораторної роботи МПП-1 “Асемблер MCS-51. Навчально-налагоджувальний стенд EV8031/AVR”, індивідуальних завдань та самостійної роботи з професійно-орієнтованої дисципліни “Мікропроцесорні пристрої” для студентів спеціальності 7.092203 “Електромеханічні системи автоматизації та електропривод” напряму “Електромеханіка”/ Упорядн.: В.І. Кириченко, О.А. Яланський, В.Г. Алпаєв. – Дніпропетровськ: Національний гірничий університет, 2006. – 42 с.

Упорядники: В. І. Кириченко, д-р техн. наук, проф.
О. А. Яланський, канд. техн. наук
В. Г. Алпаєв, молодш. наук. співробітник

Відповідальний за випуск завідувач кафедри електропривода
О.С. Бешта, д–р техн. наук, проф.

ЗМІСТ

ЗМІСТ 4
ВСТУП 6
1. ЗМІСТ САМОСТІЙНОЇ ТА ЛАБОРАТОРНОЇ РОБІТ 6
1.1 Самостійна робота 6
1.2. Лабораторна робота 6
Програма виконання 6
Вказівки щодо складання звіту 7
2. АСЕМБЛЕР MCS-51 7
2.1. Правила запису програм мовою асемблера 7
2.2. Опрацювання виразів у процесі трансляції 8
2.3. Псевдокоманди асемблера 9
2.4. Директиви символічних визначень 9
2.5. Директиви вибору сегменту 10
2.6. Директиви резервування і ініціалізації пам'яті 11
2.7. Директиви компонування програми 11
2.8. Директиви керування станом асемблера 12
2.9. Директиви макровизначень 13
2.10. Директиви змінювання властивостей файлів, отримуваних при трансляції 14
3. НАВЧАЛЬНО-НАЛАГОДЖУВАЛЬНИЙ СТЕНД “EV8031/AVR” 18
3.1. Загальні відомості, призначення 18
3.2. Склад стенда і технічні характеристики 18
4. ОПИС ОСНОВНОЇ ПЛАТИ СТЕНДА І ЇЇ ФУНКЦІОНУВАННЯ 19
4.1. Організація пристроїв пам’яті 20
4.2. Завантаження даних до стенду і запуск програми користувача на виконання 21
4.3. Розподіл адресного простору 24
4.4. Послідовний прийомопередавач 25
4.5. Пристрої введення і виведення інформації 25
4.6. Поєднання ОЕВМ і EEPROM пам'яті 26
4.7. Розташування елементів, призначення роз’єднувачів і перемикань 26
5. ПЛАТА РОЗШИРЕННЯ ТА ЇЇ ПРИЗНАЧЕННЯ 27
5.1. Цифроаналоговый перетворювач 29
5.2. Аналого-цифровий перетворювач 29
5.3. Генератори 29
5.4. Введення дискретної інформації 29
5.5. Виведення дискретної інформації 29
6. РОБОТА ЗІ СТЕНДОМ 31

6.1. Підготовка і завантаження програми користувача з використанням IIC COMPASS\51 31
6.2. Підготовка і завантаження програми користувача з використанням транслятора ASM51.EXE 32
Запитання для самоперевірки 33
СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ 41

ВСТУП

Мета самостійної та лабораторної робіт – поглиблення знань з дисципліни “Мікропроцесорні пристрої” шляхом теоретичного вивчення та отримання навичок практичного використання мови асемблера для програмування мікроконтролера 8051АН сімейства MCS-51, ознайомлення з організацією, складом та функціонуванням навчально-налагоджувального стенду EV8031/ AVR і отримання інформації та навичок щодо використання макрозасобів і псевдокоманд при підготовці програмного забезпечення.

1. ЗМІСТ САМОСТІЙНОЇ ТА ЛАБОРАТОРНОЇ РОБІТ

1.1 Самостійна робота

Самостійна робота полягає у вивченні теоретичного матеріалу, який викладений у методичних вказівках. Слід ознайомитися з правилами побудови програмного забезпечення на мові асемблера для мікроконтролерів 8051АН сімейства MCS-51, а також складом та функціональними можливостями навчально-налагоджувального стенду EV8031/AVR, побудованому на цьому мікроконтролері. Після засвоєння матеріалу слід відповісти на запитання самоперевірки та виконати індивідуальне завдання до самостійної роботи.

1.2. Лабораторна робота

Виконується у комп’ютерних класах кафедри. Метою роботи є отримання навичок практичної підготовки програм на мові асемблера мікроконтролера 8051АН сімейства MCS-51 із використанням макрозасобів та псевдокоманд, ознайомлення зі складом, призначенням та функціонуванням навчально-налагоджувального стенду EV8031/AVR. До виконання допускаються студенти, які ознайомилися з теоретичним матеріалом п.п. 2…6 методичних вказівок, відповіли на запитання самоперевірки, виконали індивідуальне завдання до самостійної роботи, підготували попередній звіт відповідно до індивідуального завдання (варіант – за номером у журналі групи) та вибраного рівня складності. Захист роботи – шляхом демонстрації викладачеві роботи розробленої програми та заключної тестової перевірки на ПЕОМ.

Програма виконання

- Ознайомитись з особливостями мови асемблера для MCS-51.
- Вивчити псевдокоманди мови асемблера.
- Вивчити структурну схему і склад стенду, а також призначення його окремих компонентів.

- Ознайомитись з роботою стенду та послідовністю дій при завантаженні програм до нього.
- Виконати пробне завантаження програми, яка наведена у п.6.
- Після засвоєння матеріалу п.п. 2…6 методичних вказівок відповісти на запитання самоперевірки.
- Виконати індивідуальний варіант самостійної роботи (номер варіанту повинен збігатися з порядковим номером студента у журналі групи).
- Виконати індивідуальне завдання лабораторної роботи.
- Скласти підсумковий звіт та захистити роботу шляхом тестової перевірки.

Вказівки щодо складання звіту

Попередній звіт повинен містити:

- Назву, мету та програму роботи.
- Відповіді на запитання самоперевірки.
- Відповіді на запитання індивідуального варіанту самостійної роботи.
- *Підсумковий звіт*, окрім перших трьох пунктів попереднього звіту, повинен містити алгоритм та налагоджену програму індивідуального завдання вибраного рівня складності, перевірені та підписані викладачем.

Увага! Допуском до виконання лабораторної роботи є підготовлений попередній звіт згідно поставлених до нього вимог.

2. АСЕМБЛЕР MCS-51

Асемблер являє собою спеціальну програму, призначену для трансляції мнемонік команд вихідної текстової програми (вихідного модуля), написаної мовою асемблера, в об'єктну програму (об'єктний модуль), що містить машинні коди команд.

2.1. Правила запису програм мовою асемблера

Вихідний текст програми мовою асемблера має певний формат. Кожна команда (і псевдокоманда) являє собою рядок складений із чотирьох ланок:

МІТКА, ОПЕРАЦІЯ, ОПЕРАНД(И), КОМЕНТАР.

Ланки (поля) можуть відокремлюватись одна від одної довільним числом пропусків або символів табуляції.

Мітка. У поле мітки розміщається символічне ім'я елементу пам'яті, у якому зберігається відзначена команда. Мітка – буквено-цифрова комбінація, перший елемент якої буква. Використовуються лише букви латинського алфавіту. Асемблер МК51 допускає використання в мітках символу підкреслювання (_). Для МК51 довжина мітки не повинна перевищувати 31 символ. Мітка завжди завершується двокрапкою (:).

Псевдокоманди асемблера не перетворюються у двійкові коди, а тому не мають міток. Виключення становлять псевдокоманди резервування пам'яті та визначення даних (DS, DB, DW). У псевдокоманд, що здійснюють визначення символічних імен, у поле мітки записується обумовлене символічне ім'я, після якого двокрапка не ставиться.

Як символічні імена і мітки не можуть бути використані мнемонічні імена команд, псевдокоманд і операторів асемблера, а також мнемонічні позначення регістрів та інших внутрішніх блоків МК.

Операція. У поле операції записується мнемонічне позначення команди МК або псевдокоманди асемблера, що є скороченням (абревіатурою) повного англійського найменування виконуваної дії. Наприклад: MOV – move – перемістити, JMP – jump – перейти, DB – define byte – визначити байт.

Для МК51 використовується чітко визначений і обмежений набір мнемонічних кодів. Будь-який інший набір символів, розміщений у полі операції, сприймається асемблером як помилковий.

Операнди. У цьому полі визначаються операнди (або операнд), тобто дані, які беруть участь у виконанні операції (команди). Команди асемблера можуть бути без-, одно- або двохоперандними. Операнди розділяються комою (,).

Операнд може бути заданий безпосередньо або у вигляді його адреси (прямої або непрямої). Безпосередній операнд представляється числом (MOV A, #15) або символічним ім'ям (ADDC A, #OPER2) з обов'язковою вказівкою префікса безпосереднього операнда (#). Пряма адреса операнда може бути задана мнемонічним позначенням (IN A, P1), числом (INC 40), символічним ім'ям (MOV A, MEMORY). Вказівкою на непряму адресацію служить префікс (@). У командах передачі керування операндом може бути число (LCALL 0135H), мітка (JMP LABEL), непряма адреса (JMP @A) або вираз JMP $-2, де $ – поточний вміст лічильника команд.

Використані як операнди символічні імена і мітки повинні бути визначені, а всі числа представлені із вказівкою системи числення, для чого використовують суфікс (буква, що ставиться після числа): B – для двійкової, Q – для вісімкової, D – для десяткової й H - для шістнадцяткової. Число без суфікса за замовчуванням вважається десятковим.

Коментар. Поле коментарю може бути використане програмістом для текстового або символьного пояснення логічної організації прикладної програми. Поле коментарю повністю ігнорується асемблером, а тому в ньому припустимо використовувати будь-які символи. За правилами мови асемблера поле коментарю починається після крапки з комою (;).

2.2. Опрацювання виразів у процесі трансляції

Асемблер МК51 допускає використання виразів у полі операндів, значення яких обчислюються у процесі трансляції.

Вираз – це сукупність символічних імен і чисел, пов'язаних операторами асемблера. Оператори асемблера забезпечують виконання арифметичних («+»– додавання, «–« – віднімання, «*» – множення, «/» – ціле ділення, MOD – ділення за модулем) і логічних (OR – АБО, AND – І, XOR що виключає АБО, NOT – заперечення) операцій у форматі 2-байтних слів.

Наприклад, запис ADD A, #((NOT 13) + 1) еквівалентний запису ADD A, #0F3H і забезпечує додавання вмісту акумулятора із числом 13, поданим у додатковому коді.

Широко використовуються також оператори LOW і HIGH, що дозволяють виділити молодший і старший байти 2-байтного операнду.

2.3. Псевдокоманди асемблера

Асемблер транслює початкову програму в об'єктні (машинні, чисельні) коди. Хоча він бере на себе багато з рутинних завдань програміста, таких як встановлення абсолютних адрес, перетворення чисел, встановлення чисельних значень символьним змінним тощо, програміст все ж таки повинен вказати йому деякі параметри: початкову адресу та кінець прикладної програми, формати даних тощо. Всю цю інформацію програміст розміщує у початковому тексті прикладної програми у вигляді директив, які керують процесом трансляції і не перетворюються в коди об'єктного модуля.

Асемблер сприймає низку директив, які дозволяють надавати символічні визначення змінним, резервувати і ініціалізувати простір пам'яті, визначати розташування отримуваного об'єктного коду в пам'яті. За винятком DB і DW директиви не створюють об'єктного коду. Директиви враховуються на етапі трансляції програми з мови асемблера в об'єктний модуль.

Директиви асемблера можуть бути розділені на низку категорій:

- символічні визначення;
- резервування простору пам'яті та ініціалізація даних;
- керування станом асемблера;
- визначення макрокоманд;
- зміна властивостей файлів, отриманих на етапі трансляції.

Нижче наведені найчастіше вживані директиви по категоріях і стисло описуються результати їх дії. Інформацію щодо інших можна знайти в довідковій системі COMPASS/51.

2.4. Директиви символічних визначень

Директиви символічних визначень можуть бути використані для того, щоб зарезервувати простір пам'яті, поставити у відповідність символічним іменам певні числові значення, регістри процесора і сегменти (області) пам’яті. Ці директиви

вимагають, щоб символьне ім'я було визначене разом з адресою, числовим значенням, регістром або типом сегменту.

BIT Визначає символічне ім'я, що посилається на адресу біта:
<символічне ім’я> BIT <адреса біту>

EQU Призначає символічному імені числове значення або ім'я регістра:
<символічне ім’я> EQU <ім’я регістру або число>

SET Призначає символічне ім'я числовому значенню або регістру. Ім'я може бути згодом змінене за допомогою директиви SET:
<символічне ім’я> SET <адреса элемента пам’яті>

```
NAME1 SET 50H    ;встановлення відповідності регістру з
                 ;адресою 50 символьному імені NAME1
PEREP BIT PSW.7  ;встановлення відповідності біта C
                 ;символьному імені PEREP
```

DATA Пов'язує адресу внутрішньої пам'яті, що безпосередньо адресується, з символьною змінною. Якщо значення адреси лежить в межах 0..127 (0..07FH) – це адреси з внутрішньої пам'яті даних, якщо в межах 128..255(80..0FFP) – це адреси одного із регістрів спеціальних функцій (SFR).

```
PSW   DATA 0D0H ;адреса слова стану програми
BUFER DATA 32   ;адреса із внутрішньої пам'яті даних
```

IDATA Пов'язує адресу внутрішньої пам'яті даних, що безпосередньо адресується, з символьною змінною. Числове значення адреси може бути в діапазоні 0…255 (0..FFH).

```
ADR IDATA 60
```

XDATA Пов'язує адресу, розташовану в зовнішній пам'яті даних, із символьною змінною. Числове значення адреси може бути в діапазоні 0..65535 (0..FFFFH) (64K – максимально можливий розмір зовнішньої пам'яті для контролерів сімейства x51).

```
ADR XDATA 2060
```

2.5. Директиви вибору сегменту

Наступні директиви визначають сегменти даних і коду:

BSEG Вибирає абсолютний бітовий сегмент;
CSEG Вибирає сегмент програми в машинному коді;
DSEG Вибирає абсолютний сегмент резидентної пам'яті даних;
RSEG Вибирає заздалегідь визначений переміщуваний сегмент;
XSEG Вибирає абсолютний сегмент зовнішньої пам'яті даних.

Сегмент **CSEG** за умовчанням встановлюється при запуску асемблера.

Кожен сегмент має свій лічильник адрес, який скидається при запуску асемблера. Вміст цього лічильника може бути змінено за допомогою використовування команди **AT**, яка ставиться після директиви вибору сегменту. Як параметр команди **AT** може бути число, вираз, раніш визначена символьна змінна. Слід пам'ятати про те, що значення лічильника не повинно виходити за межі сегменту.

```
DSEG
BSEG  AT 12
```

2.6. Директиви резервування і ініціалізації пам'яті

Ці директиви використовуються для резервування і ініціалізації слів, байтів або бітів. У абсолютному сегменті пам’яті зарезервований простір починається з поточної адреси, у переміщуваному – з поточного зсуву.

DB Заносить в пам'ять програм байтові константи. Може використовуватися, коли активний сегмент CSEG. Константа, що зберігається, може бути числом, арифметичним виразом, символьним значенням або символом ASCII. Якщо після оголошення директиви **DB** кількість змінних більше 1, то вони розділяються комами.

```
CONSTANTS:
     DB   12, 34, 34, 56, 67
STRING:
     DB   ‘sadfgghd’, ‘qwerrt’
MURA:
     DB   12, ‘sdf’, 2*23, ‘sdf’
```

DL Заносить в пам'ять програм чотирьохбайтову константу

DW Заносить в пам'ять програм двохбайтову константу

Синтаксис запису усіх трьох директив однаковий:

DB{DW,DL} [[<кількість ділянок пам'яті>]]<дані> [,[[< кількість ділянок пам'яті>]]<дані> …]

```
DB [5] 1,[4] 3  ;резервування і ініціалізація 10 ;елементів пам'яті:
                ;5 перших встановлюється в одиницю,
                ;4 других встановлюється в трійку.
```

2.7. Директиви компонування програми

Директивами компонування програми вживають для присвоєння об'єктному модулю імені, а також визначення взаємозв’язків між символьними іменами і мітками поточного програмного модуля з іншими частинами прикладної програми. Ці директиви використовують в COMPASS/51 для об'єднання окремих об'єктних модулів в єдиний абсолютний об'єктний модуль.

EXTERN Визначає символічні імена, які оголошені в інших об'єктних модулях:

EXTERN <мітка> [:<тип пам'яті>][<мітка>[:<тип пам'яті>...]]

TITLE Визначає ім'я об'єктного модуля. Бажано, щоб ім'я модуля співпадало з ім'ям файлу, в якому він міститься:

TITLE <строка символів>

PUBLIC Визначає символічні імена, які можуть бути використані в інших об'єктних модулях:

PUBLIC <символічне ім'я1 > [,<символічне ім'я 2>...]

INCLUDE Виконує вставку тексту з вказаного файлу до процесу асемблювання:

INCLUDE <шлях до файлу>

GLOBALS Оголошує всі мітки і імена, які використані іншими модулями**:**

GLOBALS<ON/OFF>

```
EXTERN m1Mod1, m2Mod1, m3Mod1    ;пересилання міток з ;модуля 1
TITLE PROGRAMM1                  ;заголовок PROGRAMM1
PUBLIC NAME11,NAME12             ;пересилання символічних
                                 ;імен з модуля 1
INCLUDE “C:\project1\prog0.asm”  ;текст з файлу prog0.asm
                                 ;ставиться у початкову
                                 ;програму і асемблюється
GLOBALS ON                       ;всі мітки і символьні
                                 ;імена поточного модуля
                                 ;зробити доступними ;іншим модулям
```

2.8. Директиви керування станом асемблера

Ці директиви використовують для того, щоб повідомити про кінець трансляції програми, вибрати початкову адресу або зсув для переміщуваного сегменту, визначити використовуваний банк регістрів.

END Повідомляє про кінець трансльованого модуля:

END [<початкова адреса>]

ORG Змінює вміст асемблерного лічильника адреси поточного сегменту програми. Адресація кодів команд починається із значення, вказаного в директиві:

ORG < початкова адреса >

IF/ENDIF Трансляція за умовою:

IF <умова> <дія> [ELIF <умова> <дія> … ELSE<умова>] ENDIF

SCOPE Визначає нову область локальних міток і не має параметрів. Після застосування даної директиви знов можна використовувати мітки, визначені в програмі раніше.

```
ORG 100H              ;розташувати програму в пам'яті
                      ;розпочинаючи з елементу 100H
A1 SET #2H            ;присвоїти A1 число 2H
IF A1=1 MOV R1,A1     ;записати A1 в R1 якщо A1=1
ELIF A1=2 MOV R2,A1   ;записати A1 в R2 якщо A1=2
ELSE MOV A,A1         ;інакше записати A1 в акумулятор
ENDIF
END 101H              ;почати виконання програми з
                      ;коду, що міститься в елементі 101H
```

2.9. Директиви макровизначень

Наступні директиви використовуються для визначення макрокоманд:

MACRO Початок макровизначення, визначає ім'я макрокоманди і параметри, які можуть бути передані макроко-манді:
< ім'я макрокоманди>: MACRO <параметр 1>, <параметр 2>, …

ENDMAC Закінчує макровизначення:
ENDMAC < ім'я макрокоманди>

На мові асемблера макрокоманди є своєрідним аналогом функцій користувача на мовах високого рівня. За їх допомогою можна реалізувати математичні і логічні операції, не включені в стандартний перелік команд асемблера. На відміну від функцій мови високого рівня, макрокоманда може мати необмежену кількість початкових параметрів, кожен з яких них може бути як аргументом, так і функцією залежно від структури власне макрокоманди.

Будь-яка макрокоманда складається з трьох основних частин:

- заголовка, де вказується ім'я макрокоманди і перелік формальних параметрів;
- тіла макрокоманди, де описуються дії, які вона виконує;
- рядка завершення.

Макровиклик здійснюється вставкою в тіло основної програми імені макрокоманди зі списком фактичних параметрів, які асемблер ставить замість формальних параметрів макрокоманди. Кількість фактичних параметрів завжди повинна відповідати кількості обумовлених в заголовку макрокоманди формальних параметрів (навіть якщо останні з якихось обставин не використовуються макровикликом).

```
BRA:  MACRO ARG1,ARG2  ;заголовок макрокоманди, в якому
                       ;оговорюється ім'я макровизначення
                       ;“BRA” і список формальних ;параметрів ARG1,ARG2
      mov ARG1,ARG2    ;тіло макрокоманди
      ENDMAC BRA       ;рядок завершення макрокоманди

      BRA A,R0         ;макровиклик, де формальним параметрам
                       ;ARG1 і ARG2 ставляться у від
                       ;повідність акумулятор і регістр R0.
                       ;при виконанні макрокоманди над акумулятором ;
                       ;і регістром R0 будутьвиконані всі
                       ;операції, які виконувались з відповідними їм
                       ;формальними параметрами.
```

2.10. Директиви змінювання властивостей файлів, отримуваних при трансляції

При виконанні трансляції програми IIC COMPASS/51 окрім створення об'єктного модуля генерує ще декілька файлів, у тому числі і файл роздруку (лістинг-файл), в якому виводиться структура програми і повідомлення про помилки, якщо такі є. Часто, щоб одержати дані про результати трансляції в максимально зручній формі, параметри створення довідкових файлів потребують коректування. Для цього призначені наступні директиви:

CONDLIST Керування роздрукуванням блоків з умовами (директива IF/ENDIF):

CONDLIST<ON/OFF>

LIST Керування виведенням тексту програми в лістинг-файлі (не відпрацьовується після виконання директиви NOLIST):

LIST<ON/OFF>

NOLIST Вимикає виведення тексту програми в лістинговому файлі (параметрів не має). Не відпрацьовується після виконання директиви LIST.

NEWPAGEРозпочати виведення лістингу з нової сторінки. Директива не має параметрів і недійсна після відпрацювання директив NOLIST або LIST OFF.

MACLIST Керування виведенням в лістинг-файл макрокоманд:

MACLIST<ON/OFF>

PW Вибір ширини сторінки лістинг-файлу в межах 80…128 символів (за умовчанням – 80 символів):

PW <ціле число>

PL Встановлення довжини сторінки файлу роздрукування в межах 15…255 рядків (за умовчанням 56 рядків):

PL <ціле число>

```
CONDLIST OFF    ;блоки, які не задовільняють умові в лістинг
                ;не виводяться;
NEWLIST    ;виводити з нової сторінки;
LIST ON    ;друкувати програму в лістинг-файл;
MACLIST OFF     ;не друкувати макрокоманди;
PW 100     ;ширина сторінки 100 символів;
PL 200     ;довжина сторінки 200 рядків.
```

Приклад.

Як приклад наведена програма таймеру зворотного відліку (установочна межа 9,9 с) з використанням директив різного типу і призначення.

Програмний модуль 1.

```
;---------- Директивна ділянка основного модуля----------------
;-------------------------------------------------------------
;---- Директиви зміни властивостей *.lst – файлу---------------
;-------------------------------------------------------------
MACLIST    on              ;виводити у лістингу
                           ;макровизначення
pw 100                     ;ширина сторінки - 100 символів
pl 200                     ;довжина сторінки - 200 рядків
;-------------------------------------------------------------
;--------- Опис символічних імен користувача-------------------
;-------------------------------------------------------------
D equ 15h                  ;поставити у відповідність
;змінній D адресу 15h елемента пам’яті
S equ 14h                  ;поставити у відповідність
                           ;змінній S адресу 14h елемента пам’яті
;-------------------------------------------------------------
;--------- Опис символічних імен,-----------------------------
;--------- що використовуються у декількох модулях-------------
;-------------------------------------------------------------
globals on            ;увімкнути режим обміну символічними
                      ; іменами між окремими модулями
public vozvr1,vozvr2  ;дозволити звертатись до міток vozvr1,
                      ;vozvr2
extern desyatie,sek   ;оголошення міток desyatie,sek, які
;знаходяться в інших модулях
;-------------------------------------------------------------
;-------------- Опис макророзширення--------------------------
;-------------------------------------------------------------
READROM: MACRO ADRROMnach,RAM_S,RAM_D ;Створення макровизначення
mov dptr,ADRROMnach                   ;за допомогою якого, можна
mov a,#1h                             ;зчитувати з пам’яті програм
movc a,@a+dptr                        ;масив даних з 2 елементів
```

```
mov RAM_S,a                             ;розпочинаючи з певної адреси
mov a,#2h                               ;При звертанні до цього
movc a,@a+dptr                          ;макророзширення перший
mov RAM_D,a                             ;параметр – початкова адреса
ENDMAC READROM                          ;масиву, другий і третій –
                                        ;елементи пам’яті програм
                                        ;для зберігання зчитаних даних
;-------------------------------------------------------------
;------------ Блок керування станом транслятора ---------------
;-------------------------------------------------------------
org 50h                                 ;почати заносити коди у
                                        ;пам'ять програм, розпочинаючи
                                        ;з адреси 50h
db 0h, 9h, 1h                           ;резервування і ініціалізація
                                        ;3 байтів у пам’яті програм
                                        ;перший встановлюється у 0h
                                        ;другий у 9h (секунди)
                                        ;третій у 1h (десяті секунди)
org 0h                                  ;почати заносити коди у
                                        ;пам'ять програм, розпочинаючи
                                        ;з адреси 0h (завантаження ко
                                        ;дів програми)
scope                                   ;визначити нову область
                                        ;локальних міток у програмному
                                        ;модулі
;-------------------------------------------------------------
;---------------------- Початок програми ----------------------
;-------------------------------------------------------------
READROM #50h,S,D                        ;виклик макровизначення
                                        ;читання даних з памяті
                                        ;програм за адресами 51h,52h)
;-------------------------------------------------------------
mov r2,D                                ;запис кількості секунд(r1)і
mov r1,S                                ;десятих секунди(r2)у проміжні
                                        ;регістри для зворотного
;рахунку
;-------------------------------------------------------------
m1:                                     ;виклик
jmp desyatie                            ;затримки часу на одну десяту
vozvr1:                                 ;секунди
dec r2                                  ;зменшення кількості десятих
mov D,r2                                ;секунди на одиницю
cjne r2,#0h,m1                          ;перевірка на нуль кількості
;десятих секунди (якщо
;кількість десятих не дорівнює
;нулю, продовжити зменшення)
cjne r1,#0h,m2                          ;перевірка на нуль кількості
                                        ;секунд. Якщо кількість секунд
jmp m3                                  ;не дорівнює нулю, виконати
                                        ;підпрограму зменшення
m2:                                     ;кількості секунд, інакше
jmp sek                                 ;зациклити програму
vozvr2:                                 ;
jmp m1                                  ;
```

```
m3:                                     ;
nop                                     ;
nop                                     ;
jmp m3                                  ;
;-------------------------------------------------------------
Програмний модуль 2.
;-------------------------------------------------------------
;-------- Директивна ділянка допоміжного модуля ----------------
;-------------------------------------------------------------
pw 100                                  ;ширина сторінки -100 символів
pl 200                                  ;довжина сторінки -200 рядків
;-------------------------------------------------------------
globals on                              ;увімкнути режим обміну
                                        ;символічними іменами між
;окремими модулями
extern vozvr1,vozvr2                    ;оголошення міток vozvr1,
;vozvr2 , які знаходяться
;в інших модулях
public desyatie,sek                     ;дозволити звертатись до міток
                                        ;desyatie, sek
scope                                   ;визначити нову область
                                        ;локальних міток у програмному
                                        ;модулі
;-------------------------------------------------------------
org 30h                                 ;почати заносити коди у
                                        ;пам'ять програм, розпочинаючи
                                        ;з адреси 30h(завантаження ко-
                                        ;дів програми)
;-------------------------------------------------------------
D set 15h                               ;поставити у відповідність
                                        ;змінній D елемент пам’яті 15h
S set 14h                               ;поставити у відповідність
                                        ;змінній S елемент пам’яті 14h
;-------------------------------------------------------------
;-------------------------------------------------------------
;--------------------- Початок допоміжного модуля -------------
;-------------------------------------------------------------
;                        Підпрограма затримки часу
;-------------------------------------------------------------
desyatie:                               ;початок
mov r6,#08h                             ;
c3:                                     ;затримка часу на одну
mov r5,#64h                             ;десяту секунди
c2:                                     ;на вкладених
mov r4,#64h                             ;циклах
c1:                                     ;
dec r4                                  ;
cjne r4,#0,c1                           ;
dec r5                                  ;
cjne r5,#0,c2                           ;
dec r6                                  ;
cjne r6,#0,c3                           ;
jmp vozvr1                              ;повернення
;-------------------------------------------------------------
```

```
;           Підпрограма зменшення кількості секунд
;-------------------------------------------------------------
sek:                                    ; початок
                                        ;
mov a,r1                                ;декрементування кількості
subb a,#1h                              ;секунд
                                        ;
mov r1,a                                ;десяткова корекція кількості
subb a,#0h                              ;секунд
da a                                    ;
                                        ;
mov S,a                                 ;встановлення лічильника
mov r2,#9h                              ;десятих секунди у початковий
mov D,#9h                               ;стан
jmp vozvr2                              ;повернення
;-------------------------------------------------------------
```

3. НАВЧАЛЬНО-НАЛАГОДЖУВАЛЬНИЙ СТЕНД “EV8031/AVR”

3.1. Загальні відомості, призначення

Навчально-налагоджувальний стенд “EV8031/AVR” – програмно-апаратний комплекс, зорієнтований для застосування в навчальних цілях по курсам програмування (мова Асемблер, СІ), а також як засіб розробки програмного забезпечення для контролерів на базі однокристальної ЕОМ серії MSC-51 або контролерів архітектури AVR.

3.2. Склад стенда і технічні характеристики

До складу стенду входять наступні компоненти:

- однокристальний процесор (ОЕОМ) типу КР1830ВЕ31, Intel 80С31, 80С51, Philips P80C31, Atmel AT89C51, TS80C31X2-MIA, AT90S8515 (4414) (DIP корпус);
- зовнішня пам'ять програм – 8 кб (6264);
- зовнішня пам'ять даних – 2×32 кб (2×62256);
- послідовна EEPROM пам'ять, 256 байт (AT93C46);
- два послідовних канали передачі даних RS232 (перемикання програмно-апаратне);
- системний інтерфейс із гніздом IDC-40;
- інтерфейс розширення (КР580ВВ55, порт Р1 ОЕВМ);
- клавіатура 4х3;
- статична 4-розрядна двоїчно-десяткова індикація;
- цифро-аналоговий та аналого-цифровий перетворювач (плата розширення);
- генератор з фіксованою частотою 10 кГц, генератор із частотою, що змінюється, від 1 кГц до 50 кГц (плата розширення);

- динамічна 4-х розрядна індикація (плата розширення);
- пристрій дискретного введення інформації: 2 кнопки (плата розширення);
- пристрій дискретного виводу інформації, світло-діоди 8 шт. (плата розширення);
- знакосинтезуючий індикатор 5х7, 1 шт. (плата розширення).

Стенд складається з основної плати і плати розширення, яка підключена до роз'єднувача Х4.

Перелік інтегральних мікросхем, а також їхні аналоги, які використані в стенді, наведені у табл. 3.1.

Таблиця 3.1.

Перелік комплектуючих мікросхем

№ п/п	Мікросхема	Позначення	Аналог	Короткий опис ІМС
1	DD1	74245	1533АП16	Прийомопередавач
2	DD2	80С31	1816ВЕ31	Однокристальна ЕОМ
3	DD3	74HC573N	1533 ИР33	8-розрядний регістр
4	DD18	8464(62256)	К537РУ17	ОЗП 8 кб (ОЗП 32кб)
5	DD19	8464(62256)	К537РУ17	ОЗП 8 кб (ОЗП 32кб)
6	DD4	27С64	К573РФ6	ПЗП 8 кб
7	DD17	8255	К580ВВ55	Паралельний прийомопередавач
8	DD12	7400	К155ЛА3	4 елементи 2 І-НІ
9	DD8,15	7404	К155ЛН1	6 елементів НІ
10	DD10	74257	1533 КП11	Мультиплексор
11	DD7	74138	1533ИД7	Дешифратор
12	DD9,13	75189	К559ИП20	Перетворювач рівня
13	DD5,22	7432	К555ЛЛ1	4 елементи АБО
14	DD20,21 DD23,24	4511	Немає аналога	Двоїчно-десятковий дешифратор
15	DD11	93С46	Немає аналога	EEPROM

4. ОПИС ОСНОВНОЇ ПЛАТИ СТЕНДА І ЇЇ ФУНКЦІОНУВАННЯ

Структурна схема стенда зображена на рис.4.1, де 8888 – система статичної індикації, ОЕОМ – однокристальна ЕОМ, BIOS – базова система вводу-виводу, ЗПП – зовнішня пам'ять програм, ЗПД – зовнішня пам'ять даних, ППП – паралельний прийомопередавач, ІПП – інтерфейс периферійних пристроїв (ПП), СВПІ – схема вибору послідовного інтерфейсу.

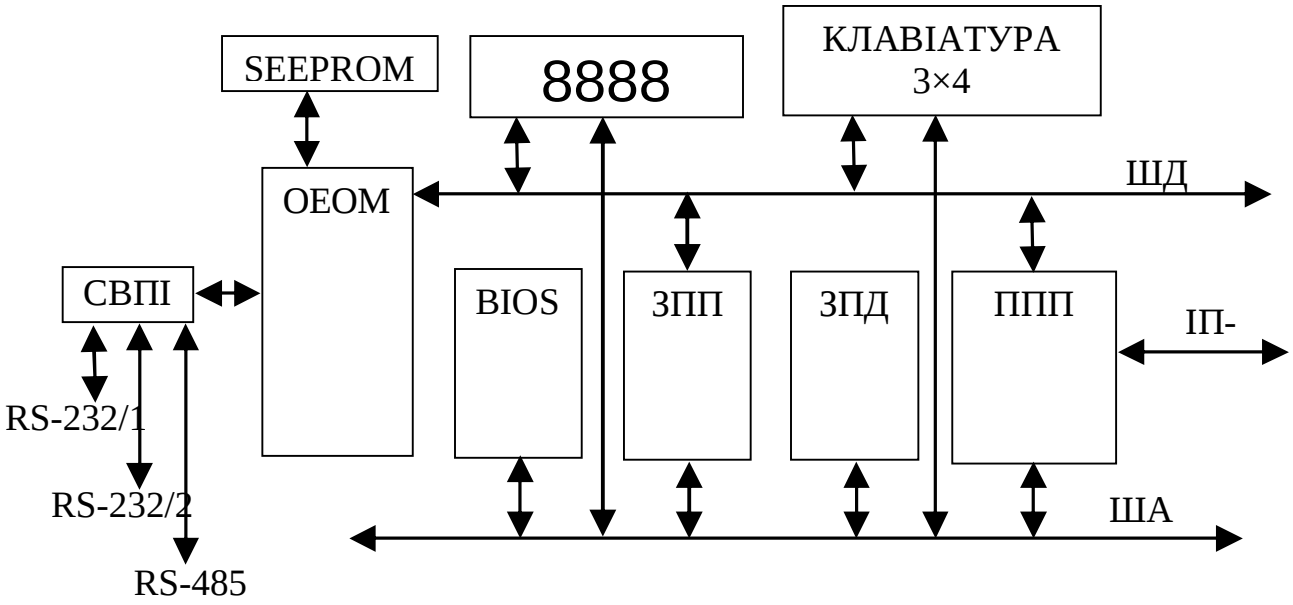


Рис.4.1. Структурна схема стенда

Принципова схема навчально-налагоджувального стенда наведена у додатку А.

Основою стенду є центральний процесор, виконаний на базі мікросхеми 80С31 (DD2), робота якого забезпечується генератором синхроімпульсів (ZQ1 – кварцовий резонатор, два конденсатори).

Вісім мультиплексованих виводів контролера (порт Р0) виконують функції шини даних ШД та молодших розрядів шини адреси ША. Вони служать або для вводу-виводу даних, або для видачі молодшого байту адреси в буферний регістр DD3, запис у який виконується за зрізом сигналу дозволу фіксації молодшого байта адреси ALE (Address Low Enable) від мікроконтролера. Разом з ним старший байт адреси елементу пам’яті видається безпосередньо з порту Р2 мікроконтролера (додаток А, рис.А-2).

4.1. Організація пристроїв пам’яті

У досліджуваній мікропроцесорній системі використані: інтегральна мікросхема пам'яті програм DD4 (BIOS), дві інтегральні мікросхеми пам'яті даних DD18, DD19 (ЗПД) (додаток А: рис.А-2,А-4). При завантаженні програми з ПК пам'ять DD19 використовується як пам'ять програм ОЕОМ (пам'ять DD4 при цьому заблокована). Подібний спосіб організації керування зовнішньою пам'яттю використаний для можливості оперативного перезавантаження прикладних програм після їх модифікації за допомогою ПК.

Доступ до ЗПД забезпечується керуючими сигналами читання RD і запису WR (активні сигнали низького рівня, що формуються апаратно при звертанні до ЗПД) з можливістю використанням 16-бітної (`MOVX A, @DPTR`) або 8-бітної адреси (`MOVX A, @Ri`). Так, при звертанні до елементу пам’яті даних з метою запису

інформації спочатку по каналу ALE передається сигнал високого рівня. За спадом цього сигналу адреса вибраного для запису елемента пам’яті, що надіслана мікроконтролером до порту P0, фіксується у проміжному регістрі DD3. Тим часом у порту P2 вже знаходиться старший байт адреси, який також надісланий в нього мікроконтролером. Таким чином, на виходах Q0...Q7 проміжного регістра і порту P2 з’являється повна адреса елементу пам’яті, до якого відбувається звернення (додаток А, рис.А-2). Через певний проміжок часу у порт P0 мікроконтролером заноситься байт корисної інформації, який необхідно записати у пам’ять. Якщо найстарший біт адреси (A15) має нульовий рівень і враховуючи, що після завантаження програми з комп’ютера мультиплексор DD10 працює у режимі з’єднування інформаційних каналів B0...B3 та Y0...Y3, то цей біт (A15), переданий через канал B1-Y1 мультиплексора, виконає роль сигналу вибору мікросхеми пам’яті даних CS RAM (Chip Select RAM) і активізує її своїм низьким рівнем. Запис корисної інформації відбудеться за надходженням до відповідного входу мікросхеми оперативної пам’яті низького рівня сигналу $\overline{WR}$ (Write Data) мікроконтролера. Зчитування даних відбувається майже аналогічно запису з тією лише різницею, що низький рівень сигналу дозволу читання надходить з мікроконтролера по каналу $\overline{RD}$ (Read Data) замість $\overline{WR}$ і напрямок руху корисної інформації змінюється на протилежний, тобто від мікросхеми пам’яті до порту P0.

При зчитуванні команд програми користувача генерація адреси елементу пам’яті, де знаходиться команда відбувається таким же чином як і при зверненні до пам’яті даних. Сигнал дозволу читання низького рівня надходить за каналом PME через мультиплексор DD10 (виводи B2-Y2 після завантаження програми користувача) до входу $\overline{OE}$ мікросхеми DD19. З надходженням цього сигналу до порту P0 передається код команди. Можливість запису до мікросхеми DD19 умисно блокована, щоб виключити спотворення коду програми користувача випадковим записом даних до цієї мікросхеми (що забезпечується -Y3 мультиплексора DD10). подачею логічної одиниці на вивід $\overline{W}/R$ мікросхеми DD19 через виводи B3

4.2. Завантаження даних до стенду і запуск програми користувача на виконання

При подачі живлення (про наявність напруги живлення свідчить запалений світлодіод HL2) або загальному скиданні програма-завантажувач із ПЗП (DD4) проводить ініціалізацію послідовного прийомопередавача OEOM (DD2), перевіряє наявність і ємність пам'яті даних не порушуючи цілісності даних у пам'яті програм, після чого формує на індикації HG1 число ємність пам'яті даних у кілобітах. Цей процес протікає у наступному порядку. Оскільки при подачі живлення до мікроконтролера усі біти його портів мають одиничний стан, струм буде протікати через резистор R10 (рис. 4.1), минаючи канал reset (сигнал reset надходить з виво-

ду P1.3 і має потенціал логічної одиниці), через вузли a, b, d та ємність C3, по якій тече зарядний струм, а також через резистор R12 , вузли c, e, f, g, ємність C4 (зарядний струм) і вузел h. При цьому вивід 1 логічного елементу I-HI DD12-1 має потенціал логічного нуля, а вивід 2 – логічної одиниці і таким чином виконується скидання мікроконтролера одиничним результуючим сигналом з виходу RST.

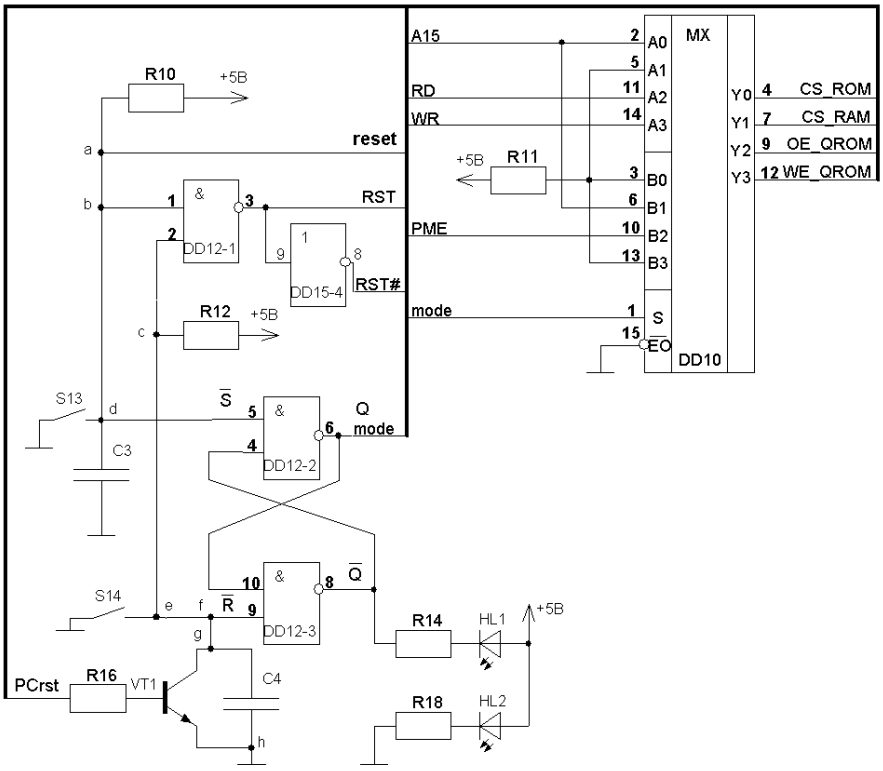


Рис.4.1. Вузол керування завантаженням програми користувача

Одночасно на асинхронний RS – тригер з вузлів d та e – також надходять сигнали логічного нуля, які переводять його у невизначений стан (RS – тригер виконаний на елементах I-HI і має інверсні входи). Після заряду ємностей C3 та C4 між корпусним виводом і вузлами d, g виникне розрив ланцюга. Як наслідок, вузли b, d, та f опиняться під потенціалом логічної одиниці.

Враховуючи, що опори R10 і R12 однакові, а ємність C3 менша ємності C4, то одиничний потенціал у вузлах b і d з'явиться раніше ніж у вузлах e, f і g. Таким чином, на входи $\overline{S}$ і $\overline{R}$ прийдуть логічні одиниця і нуль відповідно. Це викличе скид тригера, тобто вихід Q набуде потенціалу нуль, а вихід $\overline{Q}$ – одиниці. Через деякий час, коли заряд C4 перевищить порогів рівень, вхід $\overline{R}$ набуде одиничного потенціалу і тригер перейде у режим зберігання стану. Окрім того, на виводи DD12-1 надійде логічна одиниця і мікроконтролер вийде з режиму «скидання» (вхід RST опиниться під нульовим потенціалом).

Поява на виході Q тригеру DD12 логічного нуля переводить мультиплексор DD10 у стан комутування каналів A0…A3 – Y0…Y3 (сигнал mode відповідно до сигналу Q стає нульовим). У цьому режимі мікросхема постійної пам'яті DD4, що

містить завантажувач програм користувача з ПЕОМ, використовується як пам'ять програм. Сигнал зчитування команд РМЕ передається безпосередньо до цієї мікросхеми (див. принципіальну схему).

Сигнали вибору мікросхем пам’яті і дозволу на зчитування та запис даних передаються через мультиплексор DD10 (зчитування даних за каналом A2 – Y2 – вивід $\overline{OE}$ мікросхеми DD19, запис даних за каналом A3 – Y3 – $\overline{W}/R$ мікросхеми DD19, вибір мікросхеми DD4 за каналом A0 – Y0 – вивід $\overline{CE}$ мікросхеми DD4) та безпосередньо на відповідні виводи мікросхем пам’яті (біт P2.7 мікроконтролера – вивід $\overline{CE}$ мікросхеми DD19). Таким чином старший біт адреси будь-якого елементу пам’яті обов’язково повинен дорівнювати нулю, щоб забезпечити звертання до запам’ятовуючих пристроїв (активний сигнал для всіх мікросхем пам’яті стенда – логічний нуль).

Звернення до мікросхеми DD18 блокується одиничним сигналом, що надходить по каналу A1 – Y1 – вивід $\overline{CE}$ DD18. Це дає змогу залишити її чистою для використання як пам’яті даних після завантаження програми користувача. Таким чином, після подачі живлення або сигналу загального скидання стенду мікроконтролер має можливість виконувати лише програму, жорстко записану у DD4 виробником (що він і здійснює).

При передачі даних з персонального комп’ютера, MS-DOS–програма Eval32.exe надішле через вивід 9 роз’єднання X2 сигнал PCrst (імпульс одиничного рівня), який надійде на базу транзистора VT1 і на короткий час відкриє його. Короткочасно відкритий VT1 зашунтує ємність C4, що викличе її розрядження. Через розряджену ємність C4 почне протікати зарядний струм, що, у свою чергу, викличе появу у вузлах c, e, f, g нульового потенціалу до повного заряду C4. Такий самий ефект викликає натискання кнопки S14 (повний скид стану пристроїв стенду). Логічний нуль на виводах 2 мікросхеми DD12-1 і 9 мікросхеми DD12-3 (вхід $\overline{R}$ тригеру DD12) викличе скидання мікроконтролера (на вхід RST надійде логічна одиниця) і тригера DD12 (виходи Q і $\overline{Q}$ видадуть сигнали 0 і 1 відповідно). У випадку, якщо стенд виконував програму користувача, це призведе до перемикання каналів мультиплексора DD10 з B0…B3 – Y1…Y3 у A0…A3 – Y0…Y3, з подальшим блокуванням мікросхеми пам’яті програм DD18 і розблокуванням мікросхеми DD4 постійної пам’яті. Тобто, буде виконана підготовка до прийому нових даних програмою-завантажувачем мікросхеми DD4.

Із зарядженням ємності C4 мікроконтролер вийде з режиму скидання (сигнал RST змінює рівень з високого на низький), а тригер DD12 перейде у стан зберігання. При цьому мультиплексор комутує канали A0…A3 – Y0…Y3 і мікроконтролер починає виконувати програму-завантажувач з мікросхеми DD4. Коди програми користувача, прийняті з ПЕОМ, записуються до мікросхеми DD19, яка у режимі завантаження виконує функцію пам’яті даних. Після прийняття останнього

байту з комп’ютера, програма-завантажувач передає до входу P1.3 сигнал логічного нулю, який зашунтує джерело живлення на резистор R10.

Після розрядження конденсатора C3 нижче порогового рівня на входи 1 елементу DD12-1 та $\overline{S}$ тригеру DD12 подається логічний нуль, що викликає скидання мікроконтролера та зміну стану тригера (виходи Q і $\overline{Q}$ видадуть сигнали 1 і 0 відповідно). Це, у свою чергу, переводить мультиплексор DD10 у режим комутування каналів B0…B3 – Y0…Y3. Як наслідок, вибір мікросхеми DD4 блокується подачею по каналу B0 – Y0 логічної одиниці. Також блокується запис до мікросхеми DD19 (передавання логічної одиниці по каналу B3 – Y3 – $\overline{W}/R$ мікросхеми DD19), тому що вона починає працювати як пам'ять програм.

Необхідність переводу мікросхемі DD19 до режиму «тільки для читання» обумовлена захистом збережених у неї кодів програми користувача від спотворення. Сигнали керування обміном даних між пам’яттю програм, пам’яттю даних та мікроконтролером надходять або через мультиплексор DD10 (вибір мікросхеми пам’яті даних за каналом B1 – Y1 – вивід $\overline{CE}$ мікросхеми DD18, сигнал зчитування кодів команд за каналом B2 – Y2 – вивід $\overline{OE}$ мікросхеми DD19) або безпосередньо до відповідних виводів мікросхеми (біт P2.7 мікроконтролера – вивід $\overline{CE}$ мікросхеми DD19, вхід $\overline{RD}$ мікроконтролера – вивід $\overline{OE}$ мікросхеми DD18, вхід $\overline{WR}$ мікроконтролера – вивід $\overline{W}/R$ мікросхеми DD18).

Після скидання мікроконтролера сигнал з виводу P1.3 знову стає одиничним і ланцюг: джерело живлення, резистор R10, вивід P1.3 мікроконтролера розривається. Одразу ж із зарядженням ємності C3 на виводах 1 та 2 елементу DD12-1 та $\overline{S}$ тригеру DD12 з’являється логічна одиниця, яка виводить мікроконтролер з режиму скидання та переводить тригер DD12 у стан зберігання інформації. Після скидання мікроконтролер починає виконувати програму користувача, коди якої зчитуються з мікросхеми DD19. Логічний нуль, який з’являється на виході тригера $\overline{Q}$ після завантаження програми, запалює світлодіод HL2.

У випадку, коли перемичка J1 не встановлена, логічний нуль, що надсилає програма-завантажувач до P1.3, не може надійти на вихід reset і автоматично виконати скидання мікроконтролера по завершенню завантаження програми користувача. У цьому випадку користуються кнопкою S13, дія якої подібна надходженню на вивід reset логічного нуля (тобто виконується скидання мікроконтролера без втрати даних у DD19 і переведення селектора DD10 у режим комутування каналів A0…A3 – Y0…Y3 з подальшим перезапуском програми користувача з DD19).

4.3. Розподіл адресного простору

Пам'ять даних (мікросхеми DD18, DD19) мають адресацію з 0000H до 7FFFH, залежно від ємності використовуваної в стенді мікросхеми пам'яті. Всі інші периферійні пристрої адресують елементи пам'яті в просторі від 8000H до

FFFFH. Старший біт адреси A15 надходить на входи мікросхем пам’яті та вхід дозволу роботи дешифратора DD7 (додаток А, рис.А-2) і служить для активації або вибору сигналів CS (Chip Select) периферійних пристроїв стенда (A15=1), або пам’яті даних та програм (A15=0).

Пристрій відображення інформації виконано на чотирьох статичних семи сегментних двоїчно-десяткових індикаторах. Звертання до них відбувається по адресах A000H, B000H. Вибір каналу прийомопередачі здійснюється за адресою C000H, який доступний тільки для запису.

Опитування клавіатури здійснюється по адресах 9003H, 9005H, 9006H (доступні тільки для читання).

Таблиця 4.1

Розподіл адресного простору стенда

Адреса	Опис
0000 H-7FFFH	Пам'ять даних
8000H	Порт А паралельного прийомопередавача
8001H	Порт В паралельного прийомопередавача
8002H	Порт С паралельного прийомопередавача
8003H	РУС паралельного прийомопередавача
9006H	Перший стовпчик клавіатури
9005H	Другий стовпчик клавіатури
9003H	Третій стовпчик клавіатури
A000H	Ліва половина статичної індикації
B000H	Права половина статичної індикації
C000H	RS-232 & десяткові крапки статичної індикації

4.4. Послідовний прийомопередавач

Модуль послідовного зв'язку сформований на мікросхемі приймача 75189 (DD9,13), передавача К155ЛН1 (DD8), схеми вибору каналу передачі (мультиплексор DD14) (додаток А, рис.А-1). Вибір каналу послідовної передачі здійснюється сигналами CFG0, CFG1 за адресою C000H. Встановлення цих бітів у «0» вмикає порт 1 з неповним набором сигналів (роз’єднувач X2), який призначений для запису програми в стенд. Програмне встановлення сигналів CFG0 в «0», а CFG1 – в «1» формує вибір додаткового каналу послідовної передачі даних (роз’єднувач X3), який має повний набір сигналів інтерфейсу RS232.

Швидкість обміну по послідовному порту в режимі завантаження 9600 бод (бод – один біт за секунду). Швидкість обміну по послідовному порту у налагоджувальній програмі може бути змінена.

4.5. Пристрої введення і виведення інформації

Введення дискретної інформації здійснюється за допомогою дванадцятикнопочної клавіатури S1, S12 (додаток А, рис.А-1). Дані з клавіатури передаються

через буфер DD1 основної плати (входи A1-A4) до першої тетради бітів порту P0. Для активації процесу передачі даних з клавіатури до мікроконтролера необхідне надходження сигналу CS1 з дешифратора периферійних пристроїв DD7 (сигнал надходить, коли старша тетрада бітів адреси 1001B=9h) та низький рівень сигналу $\overline{RD}$ з мікроконтролера).

Вибір кнопкового стовпчика, з якого знімається корисна інформація, здійснюється за допомогою трьох наймолодших бітів коду адреси A0-A2, встановлюючи один із них у нульовий стан. Таким чином, опитування стовбців клавіатури здійснюється при звертанні до них як до звичайних елементів зовнішньої пам’яті даних за адресами 9006h, 9005h або 9003h у режимі читання.

Виведення інформації організується через чотирьох позиційний статичний індикатор (додаток А, рис.А-3), об’єднаний у два двопозиційних знакомісця. Цей індикатор складається з цифрового статичного табло HG1 і чотирьох приєднаних до нього дешифраторів семисегментного коду DD20-DD24. Вибір першого або другого знакомісця виконується за допомогою сигналів CS2 та CS3 відповідно. Ці сигнали генерує дешифратор вибору периферійних пристроїв при надходженні до нього відповідної комбінації станів бітів A15-A12 (1010B для лівого знакомісця, 1011B для правого). Активація процесу виведення даних відбувається за надходженням низького рівня сигналу дозволу на запис $\overline{WR}$ мікроконтролера. Тобто, для виведення байта корисної інформації необхідно і достатньо подати дані на запис за адресою A000h або B000h залежно від вибраного знакомісця. Слід також враховувати, що статичний індикатор двійково-десятковий, що виключає можливість виведення літерних позначень шістнадцяткових чисел. У разі передачі чисел, що містять літерні позначення (наприклад, 7Ah), позиції, де вони мають з’явитися, згаснуть.

4.6. Поєднання ОЕВМ і EEPROM пам'яті

Лінії даних і синхроімпульсів мікросхеми EEPROM (FLASH пам'ять) 93C46 (DD11) (додаток А, рис.А-1) підключені відповідно на виводи P1.3–P1.7 однокристальної ЕОМ. Порт P1 ОЕОМ може бути частково відключений від зовнішніх ліній (перемикання J 1–J5), окрім ліній інтерфейсу розширення. На роз’єднувач інтерфейсу розширення ці сигнали надходять минаючи перемикання.

4.7. Розташування елементів, призначення роз’єднувачів і перемикань

Розташування елементів та призначення роз’єднувачів і перемикань видно із рис. 4.2.

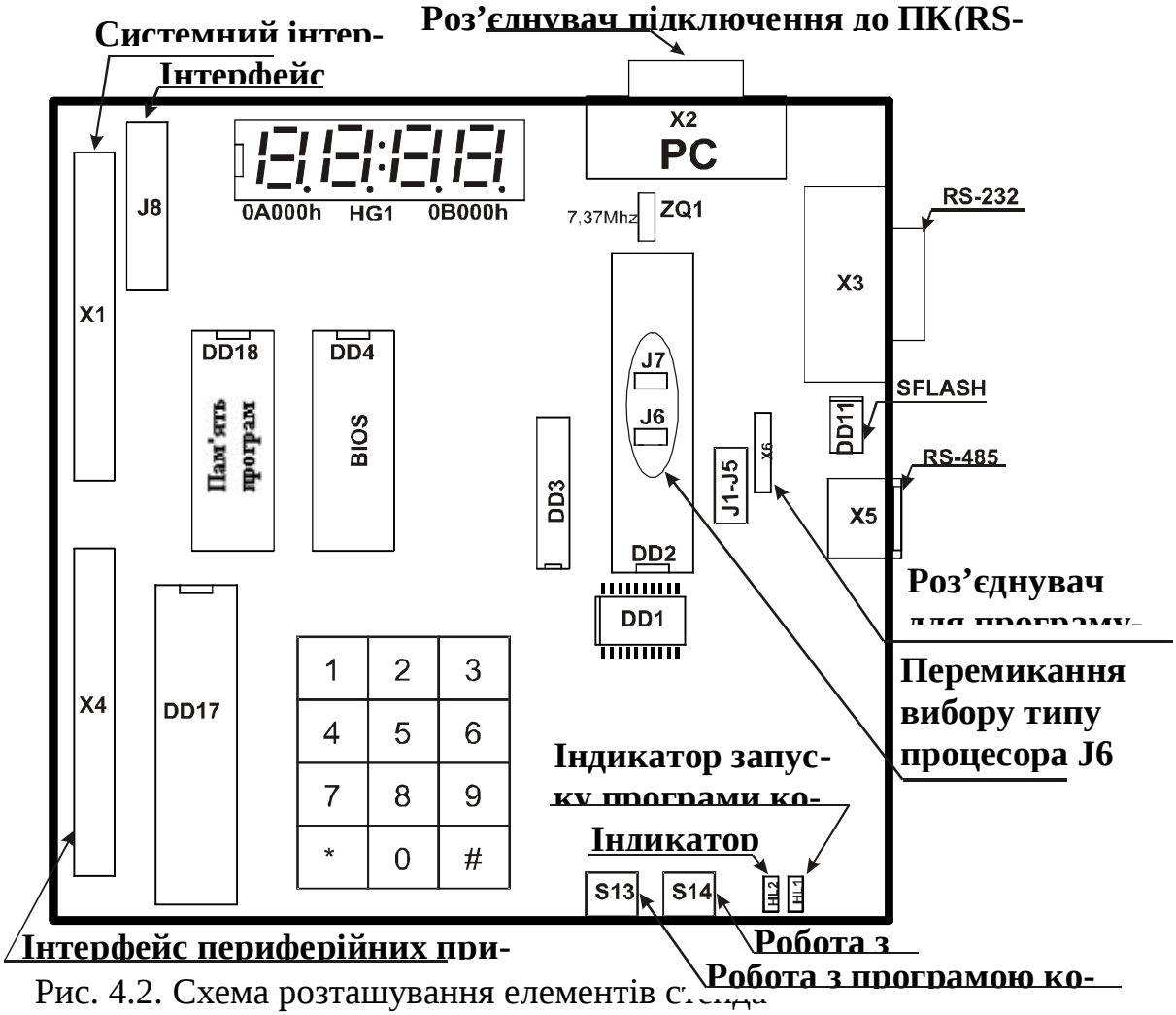


Рис. 4.2. Схема розташування елементів стенда

X1 – системний інтерфейс із повним адресним простором; X4 – інтерфейс розширення для підключення зовнішніх пристроїв з використанням паралельного приймопередавача; X2 – інтерфейс послідовного порту COM1 для зв'язку стенда із PC; X3 – інтерфейс послідовного порту COM2 для зв'язку стенда з іншими пристроями, які мають стандартний порт RS232C; X6 – інтерфейс програмування AVR; J4 – перемикання для підключення сигналу синхронізації послідовної пам'яті; J2,J3 – перемикання для підключення лінії передачі даних послідовної пам'яті; J1 – перемикання для дозволу автоматичного запуску завантаженої програми; J5 – перемикання для підключення сигналу вибору послідовної пам'яті

5. ПЛАТА РОЗШИРЕННЯ ТА ЇЇ ПРИЗНАЧЕННЯ

Плата розширення (у комплексі з навчально-налагоджувальним стендом на базі однокристальної ЕОМ серії 8031) призначена для проведення лабораторних робіт пов'язаних з цифроаналоговим, аналого-цифровим та частотним перетворен-

ням, а також з опрацюванням дискретних сигналів. Структурна схема плати розширення наведена на рис. 5.1.

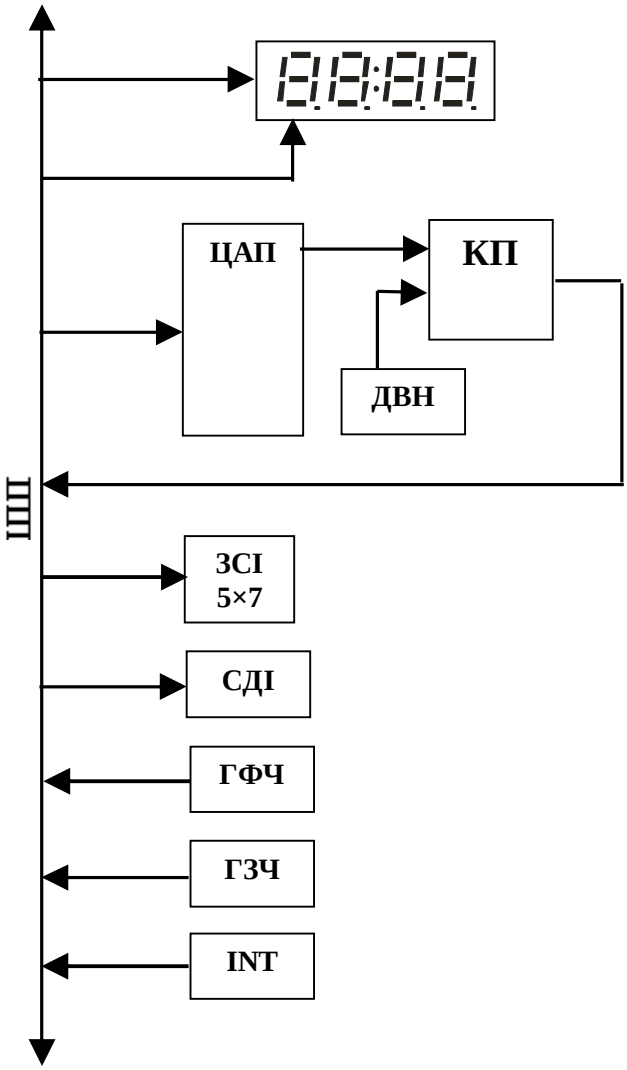


Рис. 5.1. Структурна схема плати розширення

8888 – 4 розрядна динамічна індикація; ІПП – інтерфейс периферійних пристроїв; ЦАП – цифроаналоговый перетворювач; СДІ – світлодіодні індикатори; ЗСІ – знакосинтезуючий індикатор 5х7; ГФЧ – генератор з фіксованою частотою; ГЗЧ – генератор зі змінною частотою; INT – кнопки запиту переривання; ДВН – джерело вимірюваної напруги; КП – компаратор

Принципова схема плати розширення наведена в додатку Б.

Перелік інтегральних мікросхем, використаних у стенді, а також їхні аналоги наведені у табл. 3.1.

5.1. Цифроаналоговый перетворювач

ЦАП з розрядністю 10 виконаний на мікросхемі КР572ПА1 (DA1). Вхідними сигналами для ЦАП є лінії PA0-PA7, PC0-PC1 мікросхеми К580ВВ55 (DD17) паралельного прийомопередавача. Вихідний сигнал знімається з роз'єднувача BNC, до якого, при необхідності, можна приєднати осцилограф.

5.2. Аналого-цифровий перетворювач

АЦП з розрядністю 10 виконаний на мікросхемі КР572ПА1 (DA2), інтегральному компараторі КР554СА3 (DA1), інверторі К155ЛН1 (DD3). Вхідним аналоговим сигналом для АЦП є сигнал зі змінного резистора R27, увімкненого як подільник напруги. Лінії порту PA0-PA7, PC0, PC1 мікросхеми К580ВВ55 використані для формування цифрового вхідного коду (D0-D9 відповідно). На виході ЦАП (перетворювача код/напруга) формується напруга, яка пропорційна вхідному коду. Якщо вимірювана напруга і напруга ЦАП співпадають, компаратор DA1 змінює свій стан. Сигнал спрацьовування компаратора знімається з інвертора DD3.1 (вхід ОЕОМ P1.7) і відображається горінням світлодіода HL9.

5.3. Генератори

У схемі присутні два генератори: один з фіксованою частотою 50 кГц (елементи C1, R4, R3, DD1.4-DD1.6), а інший – зі змінною частотою від 2 кГц до 10 кГц (елементи R5, R1, R7, C2, VT1, DD1.1-DD1.3). Змінювання частоти здійснюється за допомогою резистора R7.

Вихідні сигнали з генератора фіксованої частоти надходять на вхід ОЕОМ P3.4 (вхід таймера/лічильника T0), а з генератора змінної частоти – на вхід P3.5 (вхід таймера/лічильника T1).

5.4. Введення дискретної інформації

Введення дискретної інформації здійснюється за допомогою кнопок S10 та S11. Сигнали з плати розширення надходять на входи ОЕОМ P3.2 і P3.3 (P3.2 – вхід апаратного переривання INT0 ОЕОМ, P3.3 – INT1).

5.5. Виведення дискретної інформації

Виведення дискретної інформації здійснюється за допомогою чотирьох семисегментних індикаторів, поєднаних у один індикатор HL10 (увімкнений за схемою динамічної індикації), восьми світлодіодів HL1-HL8 та знакосинтезуючої світлодіодної матриці HG7.

Керування динамічною індикацією здійснюється за допомогою елементів DD3 і DD4, на які надходять сигнали з порту PB мікросхеми паралельного прийо-

мопередавача DD17 (див. додаток А, рис. А-4). Сигнали вибору відповідного індикатора надходять від ліній PC0, PC1 порту PC.

Підсвітлювання світлодіодів лінійки і знакосинтезуючої матриці здійснюється пересиланням відповідної комбінації бітів у порти PC і PA паралельного прийомопередавача. Слід врахувати, що приєднання до порту PA катодів світлодіодної лінійки здійснене через інвертори на логічних елементах АБО-НІ мікросхем DD2, DD3 і тому для запалювання відповідного світлодіода слід на приєднаний до нього інвертор подавати логічну «1».

Схема розташування елементів плати розширення показана на рис.5.2. На схемі прийняті такі позначення:

HG7 – знакосинтезуючий індикатор 5x7;
HL10 – 4-х розрядна динамічна індикація;
HL1-HL8 – світлодіодні індикатори, які підключені до порту А паралельного прийомопередавача;
HL9 – світлодіодний індикатор спрацьовування компаратора;
J1 – перемикання для підключення до роз'єднувача J2 виходу генератора постійної частоти;
J2 – роз'єднувач підключення зовнішніх контрольно-вимірювальних приладів;
J3 – перемикання для підключення до роз'єднувача J2 виходу генератора змінної частоти;
J4 – перемикання для підключення до роз'єднувача виходу ЦАП;
J5 – підключення кнопки S11 як джерела зовнішнього переривання INT1;
J6 – підключення входу переривання INT1 від зовнішнього джерела через роз'єднувач JP1;
J7 – інтерфейс підключення плати розширення до стенда;
J8 – підключення до входу АЦП зовнішнього джерела сигналу до роз'єднувача JP2;
J9 – підключення змінного резистора R27 як джерела сигналу для АЦП;
JP1 – роз'єднувач підключення джерела зовнішнього переривання INT1;
JP2 – роз'єднувач підключення зовнішнього джерела вхідного сигналу для АЦП;
S10 – кнопка як джерело зовнішнього переривання INT0;
S11 – кнопка як джерело зовнішнього переривання INT1;
R27 – змінний резистор (джерело вхідного сигналу для ЦАП);
R7 – змінний резистор, який змінює частоту змінного генератора.

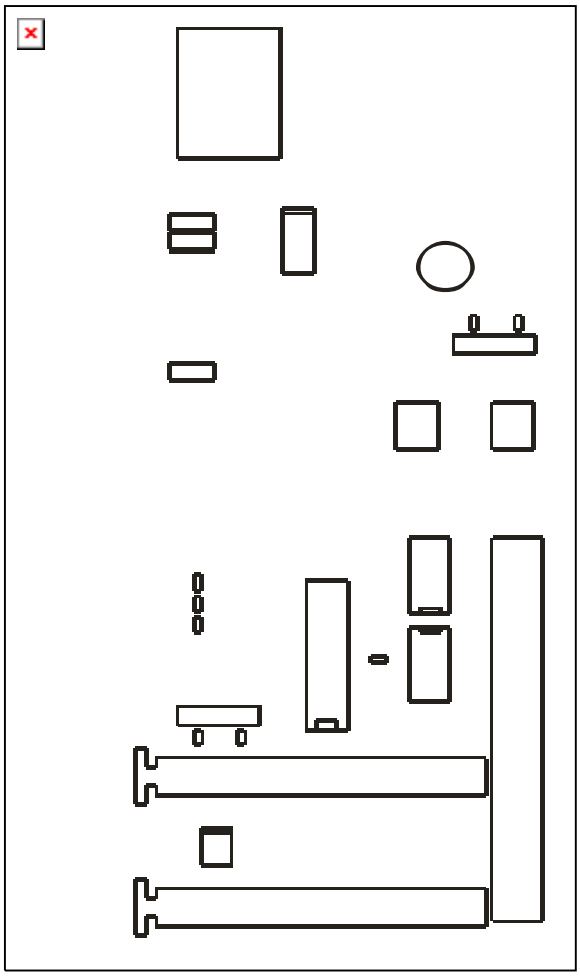


Рис. 5.2. Схема розташування елементів плати розширення

6. РОБОТА ЗІ СТЕНДОМ

Для перевірки ефективності попередньо розробленої програми її слід завантажити до навчально-налагоджувального стенду і запустити на виконання.

6.1. Підготовка і завантаження програми користувача з використанням IIC COMPASS\51

1. Створити новий проект у попередньо підготовленій папці.
2. На персональному комп'ютері відкрити текстовий редактор, наприклад, NOTEPAD.
3. У текстовому редакторі набрати текст розробленої програми у мнемокодах мови Асемблер мікроконтролерів серії MCS-51.
4. Зберегти набраний файл із розширенням *.ASM у папці проекту.
5. Відтранслювати набрану програму за допомогою інструменту Асемблер IIC COMPASS\51.
6. Можливі помилки в програмі слід переглянути в одноіменному файлі з розширенням *.LST.

7. Після того як будуть усунені всі помилки програми, за допомогою укладача слід отримати модуль з розширенням *.IHX, який завантажується до стенду програмою EVAL32.EXE (попередньо скопійованою у папку проекту).

Послідовність дій при завантажуванні модуля з розширенням *.IHX до стенду наступна. Спочатку слід викликати пункт «Выполнить» в меню кнопки «Пуск» панелі задач операційної системи. У командному рядку вікна, яке з'явиться, слід набрати послідовність (рядок) символів

Z:\...\EVAL32.EXE -hs -com 1 9600 IND.IHX

і натиснути «Enter».

У набраному рядку символів Z:\...\EVAL32.EXE – шлях до папки, в якій знаходяться програма-завантажувач EVAL32.EXE, а IND.IHX – ім'я створеної програми. У випадку, коли місцезнаходження завантажувача і завантажувального модуля програми не співпадають, замість одного імені файлу з розширенням *.IHX вказується його повний шлях. Виведення на екран підказки про параметри програми EVAL32.EXE здійснюється її запуском (подвійне натискання лівої кнопки мишки на піктограмі цієї програми).

6.2. Підготовка і завантаження програми користувача з використанням транслятора ASM51.EXE

1. На персональному комп'ютері відкрити текстовий редактор, наприклад, NOTEPAD.
2. У текстовому редакторі NOTEPAD набрати текст розробленої програми у мнемокодах мови Асемблер мікроконтролерів серії MCS-51.
3. Зберегти набраний файл із розширенням *.ASM до наперед створеної папки і скопіювати до неї файли ASM51.EXE і EVAL32.EXE.
4. Для трансляції асемблерної програми викликати пункт «выполнить» із меню кнопки «Пуск» панелі задач операційної системи. В командному рядку вікна, яке з'явиться, набрати послідовність символів

Z:\...\ASM51.EXE PROG.ASM

і натиснути «Enter».

У набраному рядку символів Z:\...\ASM51.EXE – шлях до папки, в якій знаходяться програма-транслятор ASM51.EXE, а PROG.ASM – ім'я файлу з програмою користувача. У випадку, коли місцезнаходження транслятора і файлу асемблерної програми не співпадають, замість імені файлу з розширенням *. ASM вказується його повний шлях. Можливі помилки в програмі слід переглянути в одноіменному файлі з розширенням *.LST. Після усунення всіх помилок отриманий модуль з розширенням *.HEX слід завантажити до стенду за допомогою програми EVAL32.EXE згідно правил, наведених у підпункті 7 пункту 6.1.

При передачі даних з персонального комп'ютера у стенд на екрані монітора відображається хід процесу передавання даних. Після передачі останнього байта

при наявності перемикання J1 завантажена програма запускається автоматично. За відсутності перемикання J1 або за необхідності переза-пуску завантаженої в стенд програми спочатку слід натиснути кнопку S14, а після цього – S13. Запуск програми на виконання супроводжується загоранням світлодіода HL1. Для зупинки завантаженої програми і переходу в режим очікування прийому даних з персонального комп'ютера можна натиснути кнопку S14 (світлодіод HL1 при цьому згасне). Запис нової програми можливо здійснити в будь-який момент часу роботи завантаженої програми.

Приклад програми для завантаження в стенд:

```
ORG 0
Start:
mov DPTR,#0A000h ; занести в регістр DPTR
                      ; адресу індикації
mov A,#28h       ; занести в регістр A дані для відображення
movx @DPTR,A          ; відправити на індикацію, адреса якої
                           ; знаходиться в регістрі DPTR, число 28
jmp Start                  ; перехід у початок
```

Запитання для самоперевірки

- Вкажіть достоїнства і недоліки мови асемблера.
- Що є результатом виконання трансляції?
- Яка структура рядка програми, написаної на мові асемблера, які поля рядка є обов'язковими?
- Як розділяються поля у рядку тексту програми?
- Які поля в рядку програми на мові асемблера можна залишати вільними?
- Вкажіть правила вибору імені мітки.
- Вкажіть правила вибору символічних імен.
- Що таке операнд і як вони задаються?
- Яких правил необхідно дотримуватись при використанні символічних імен як операндів?
- Яким чином виділяється коментар?
- Які символи не можна використовувати у коментарях?
- Для чого використовують директиви асемблера?
- На які групи можна розділити директиви асемблера?
- Які директиви використовуютьдля вказівки початку і кінця програмного модуля?
- Вкажіть призначення директив ORG і END.
- Вкажіть призначення директив EQU, SET, BIT. У яких випадках і коли їх слід вживати?
- Який формат директиви BIT?

- Вкажіть призначення директив DB, DS, DW. Наведіть приклади і відмінності їх використання.
- Вкажіть призначення директив TITLE, PUBLIC, GLOBALS, INCLUDE. Для чого їх слід вживати і коли?
- Вкажіть призначення директив CONDLIST, LIST, NOLIST, NEWPAGE, MACLIST.
- Наведіть приклади використання.
- У чому полягає відмінність директив EQU і SET?
- Яким чином визначається макрокоманда?
- Вкажіть достоїнства і недоліки підпрограм у порівнянні з макрокомандами.
- У яких випадках доцільно використовувати підпрограми, а в яких – макрокоманди?
- За якими правилами виконують виклик макрокоманди з основної програми?
- Із скількох плат складається навчально-налагоджувальний стенд?
- Назвіть вузли з яких складається стенд.
- Які схеми індикації реалізовані у стенді?
- Яким чином організована зовнішня пам’ять програм та зовнішня пам’ять даних? Поясніть на прикладі організації схемних рішень стенду.
- Як протікає процес завантаження даних у стенд з персонального комп’ютера?
- За яким принципом організовані шина даних та адресна шина?
- Як розподілений зовнішній адресний простір пам’яті?
- Яку функцію виконують мультиплексори та дешифратори у стенді?
- Яке призначення кнопок S13 та S14?
- З якими пристроями підтримується зв'язок через паралельний прийомопередавач?
- За яким каналом даних і як виконується завантаження програми користувача до навчально-налагоджувального стенду?
- Для чого призначена плата розширення?
- Яким чином виконується підготовка програми до завантаження у стенд?
- Які дії необхідно виконати для завантаження програми до навчально-налагоджувального стенду?
- Як перервати виконання програми без вивантаження її із пам’яті стенду?
- Як перевести навчально-налагоджувальний стенд до режиму очікування завантаження даних?
- У чому відмінність підготовлення завантажувального модуля за допомогою транслятора ASM51.EXE і IIC COMPASS\51 ?

Індивідуальні завдання низького рівня складності

Для отримання оцінки 3 (задовільно) необхідно виконати наступне:

Скориставшись розробленими (згідно порядкових номерів у журналі викладача) індивідуальними програмами низького рівня складності до лабораторної роботи МПП-3, подати їх у вигляді програм з використанням директив ORG, EQU, SET, BIT і перевірити їх ефективність за допомогою ІІС COMPASS\51 (текст нової програми повинен містити згадані директиви).

Набрати у текстовому редакторі приклад програми приведений у п.6.2, відтранслювати його і завантажити у стенд. Перевірити правильність виконання завантаженої програми.

Індивідуальні завдання середнього рівня складності

Для отримання оцінки 4 (добре), необхідно виконати наступне:

Додатково до завдання низького рівня складності МПП-5 та у відповідності до індивідуального завдання середнього рівня складності до роботи МПП-3 розробити програму з використанням у ній директив DB, DW, ORG, EQU, SET, END. Перевірити роботу програми і відпрацювання зазначених директив за допомогою ІІС COMPASS\51.

Індивідуальні завдання підвищеного рівня складності

Для отримання оцінки 5 (відмінно), необхідно виконати наступне:

Додатково до завдання низького рівня складності МПП-5 виконати завдання підвищеного рівня складності роботи МПП-3 з використанням мінімум трьох об’єктних модулів. Використати у розроблених програмах директиви DB, DW, ORG, EQU, SET, END, TITLE, EXTERN, PUBLIC, INCLUDE, GLOBALS і макрокоманду та перевірити роботу програми і відпрацювання відповідних директив і макрокоманди за допомогою ІІС COMPASS\51.

Принципова схема основної плати навчально-налагоджувального стенда

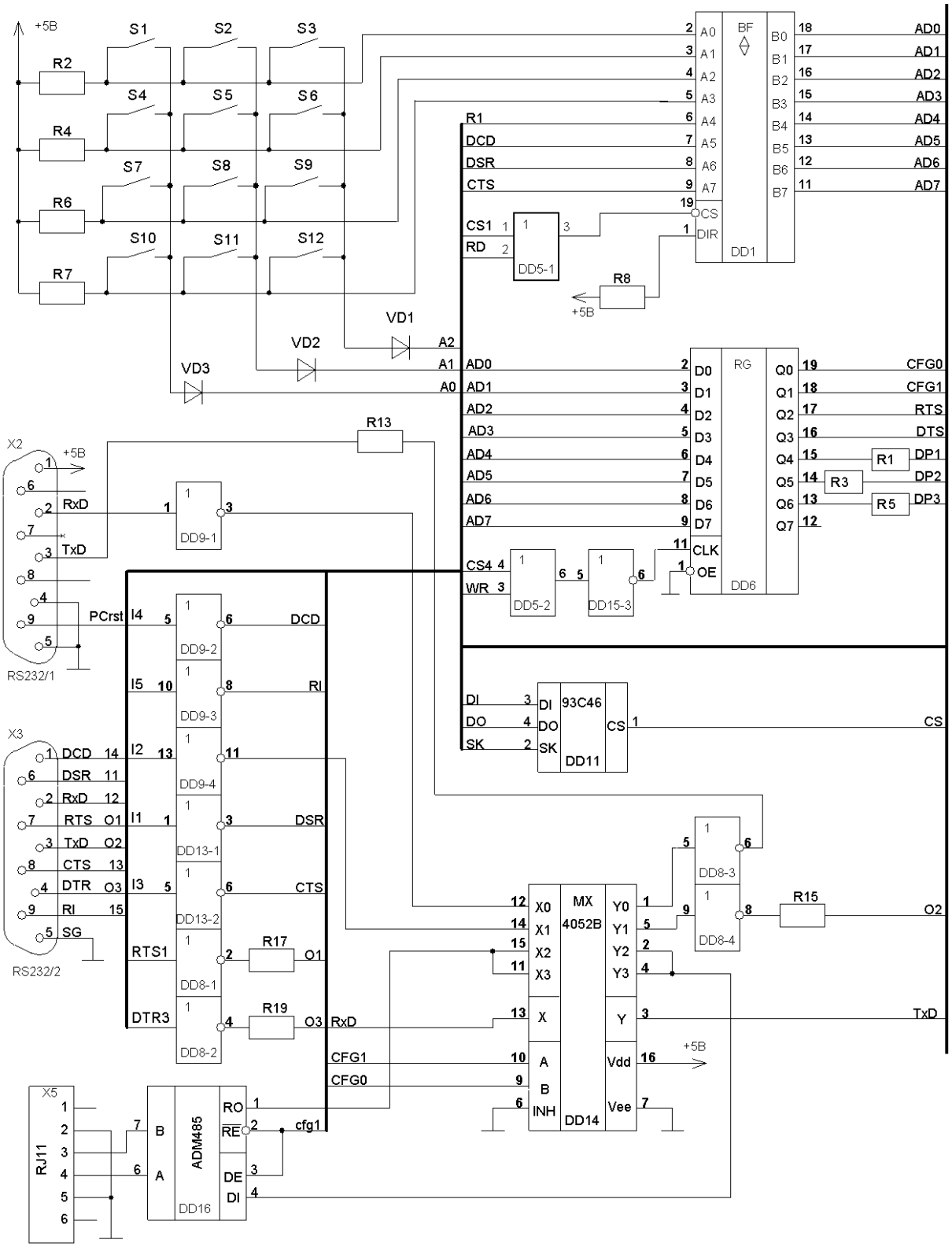


Рис. А -1

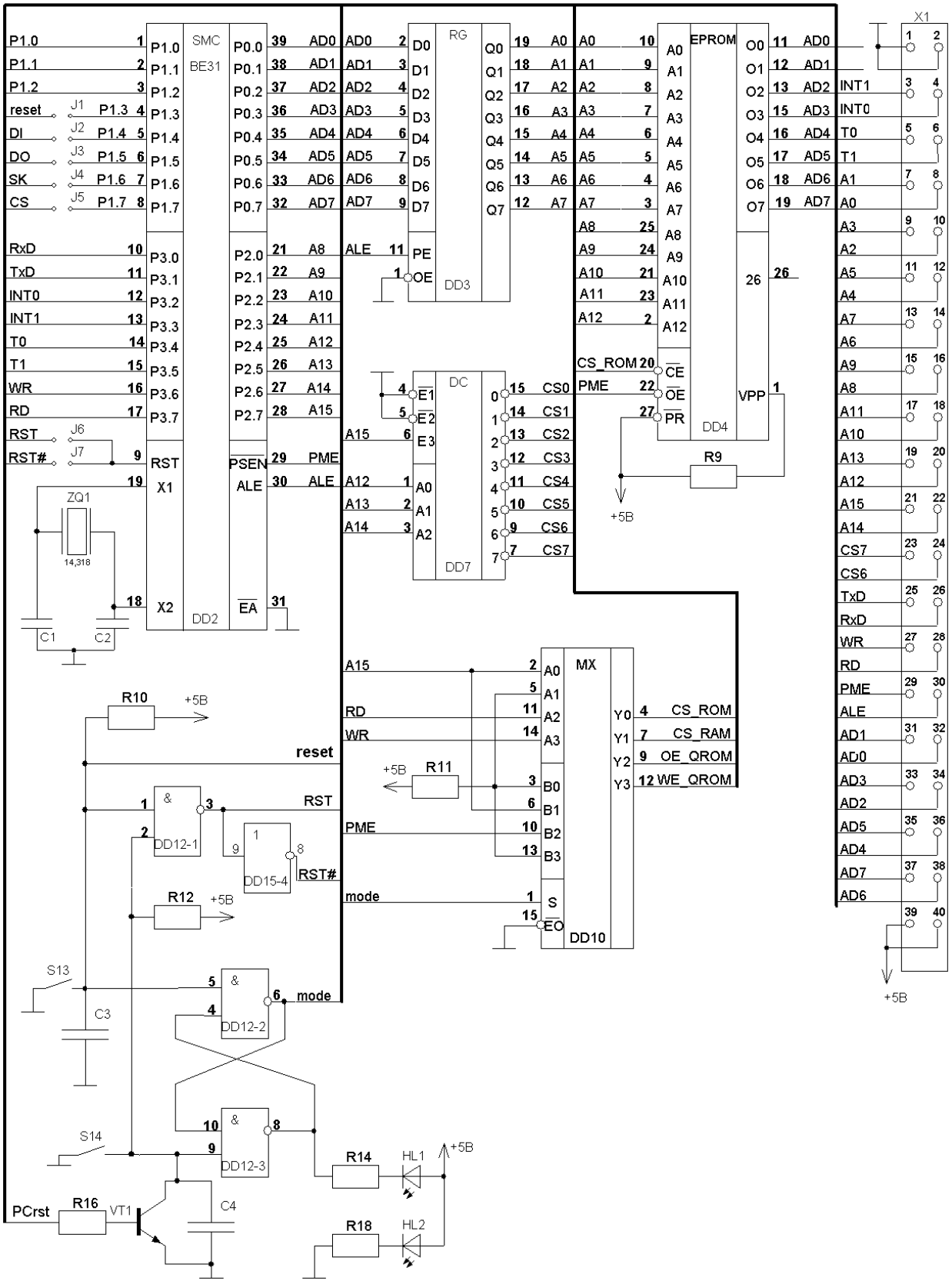

SMC
BE31
DD2
RG
DD3
EPROM
DD4
DC
DD7
MX
DD10
X1
P1.0
P1.1
P1.2
reset
DI
DO
SK
CS
RxD
TxD
INT0
INT1
T0
T1
WR
RD
RST
RST#
ZQ1
14,318
C1
C2
PSEN
ALE
PME
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
A8
A9
A10
A11
A12
A13
A14
A15
CS0
CS1
CS2
CS3
CS4
CS5
CS6
CS7
CS_ROM
CS_RAM
OE_QROM
WE_QROM
VPP
R9
R10
R11
R12
+5B
DD12-1
DD15-4
DD12-2
DD12-3
S13
S14
C3
C4
mode
PCrst
R16
VT1
R14
R18
HL1
HL2

Рис. А-2

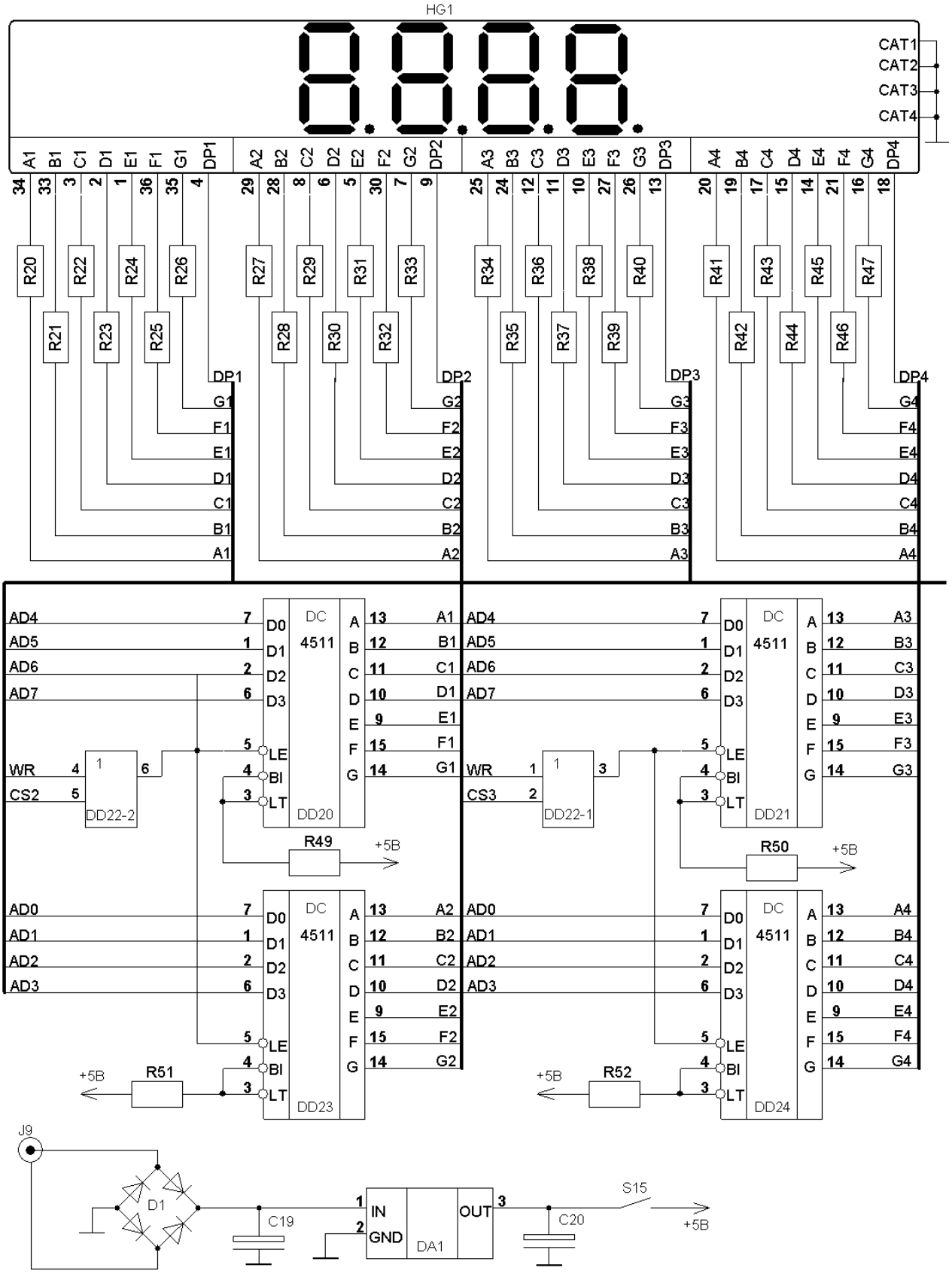

HG1
CAT1
CAT2
CAT3
CAT4
A1 B1 C1 D1 E1 F1 G1 DP1
A2 B2 C2 D2 E2 F2 G2 DP2
A3 B3 C3 D3 E3 F3 G3 DP3
A4 B4 C4 D4 E4 F4 G4 DP4
R20 R21 R22 R23 R24 R25 R26 R27 R28 R29 R30 R31 R32 R33
R34 R35 R36 R37 R38 R39 R40 R41 R42 R43 R44 R45 R46 R47
DC 4511
DD20
DD21
DD22-2
DD22-1
DD23
DD24
AD4 AD5 AD6 AD7
AD0 AD1 AD2 AD3
WR CS2 CS3
LE BI LT
R49 R50 R51 R52
+5B
J9
D1
C19 C20
DA1
IN GND OUT
S15

Рис. А-3

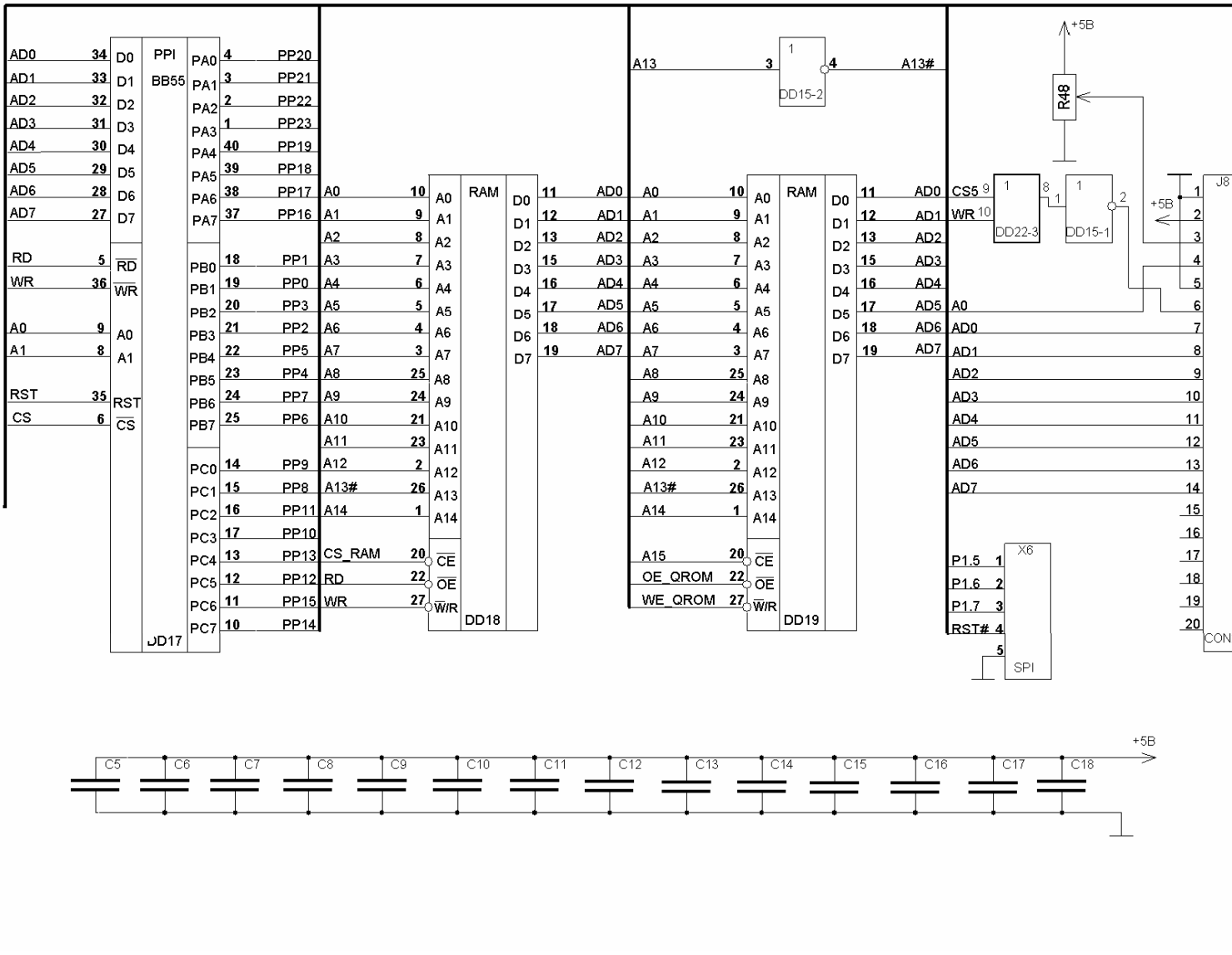
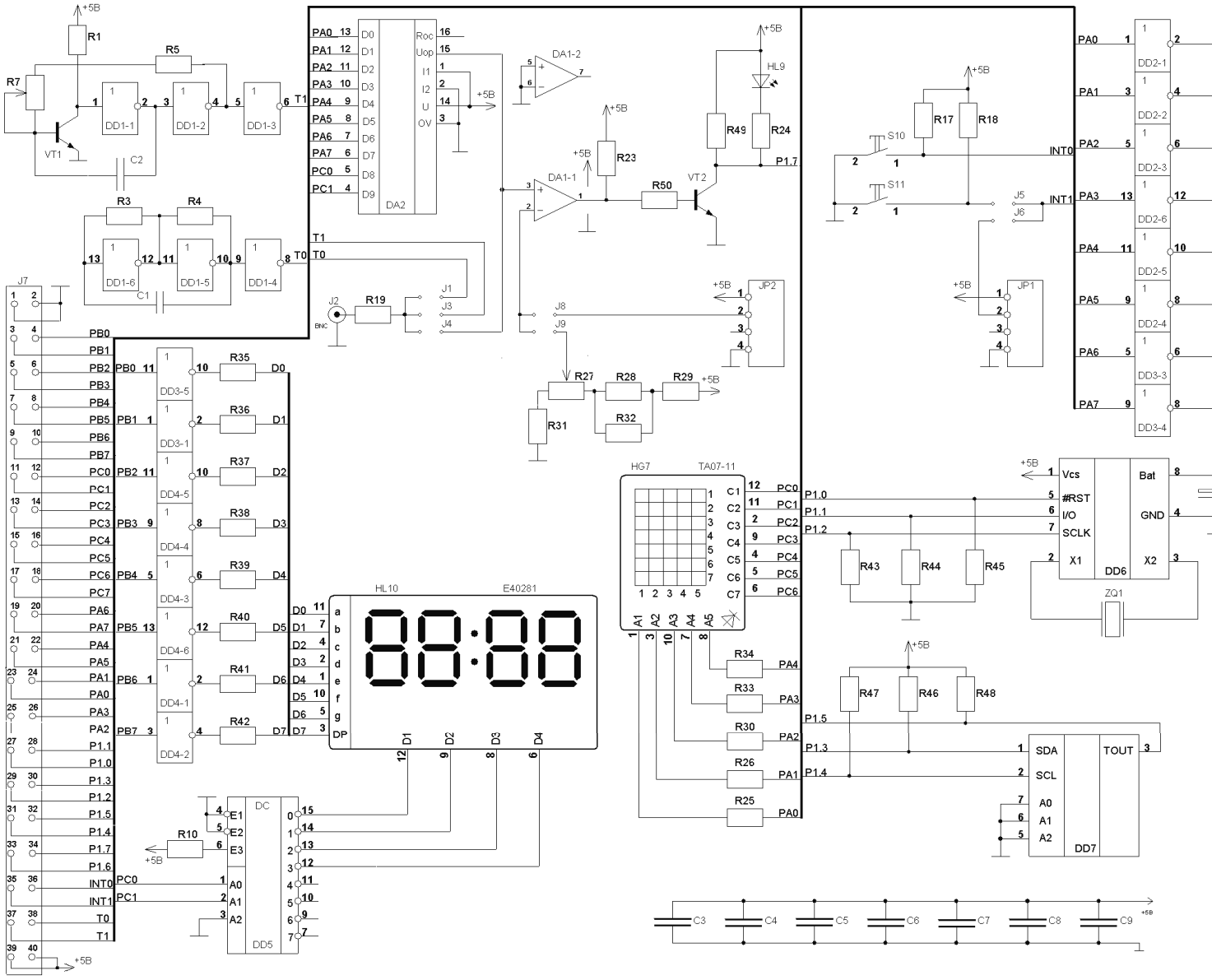

PPI
BB55
DD17
RAM
DD18
DD19
DD15-2
DD22-3
DD15-1
R48
+5B
X6
SPI
J8
CON
CS_RAM
OE_QROM
WE_QROM
A13#
C5
C6
C7
C8
C9
C10
C11
C12
C13
C14
C15
C16
C17
C18

Рис. А-4

Додаток Б

Принципова схема плати розширення навчально-налагоджувального стенда



СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Однокристальные микроконтроллеры Microchip: PIC16C5X / Под ред. А.Н. Владимирова: Пер. с англ. – Рига: ORMIX, 1996. – 96 с.
2. Встраиваемый микроконтроллер 8XC251SB: Руководство пользователя: Пер. с англ. – К.: Квазар-микро, 1995. – 418 с.
3. Гуртовцев А.Л., Гудыменко С.В. Программы для микропроцессоров: Справ. пособие. – Минск: Вышейш. шк., 1989. – 352 с.
4. Злобин В.К., Григорьев В.Л. Программирование арифметических операций в микропроцессорах. – М.: Высш. шк., 1991. – 303 с.
5. Козаченко В.Ф. Микроконтроллеры: Руководство по применению 16-разрядных микроконтроллеров Intel MCS-196/296 во встроенных системах управления. – М.: ЭКОМ, 1997. – 688 с.
6. Бродин В.Б., Шагурин И.И. Микроконтроллеры. Архитектура, программирование, интерфейс. – М.: ЭКОМ, 1999. – 400 с.
7. Однокристальные микроЭВМ: Справочник / А.В. Бобрыкин, Г.П. Липовецкий, Г.В. Литвинский и др. – М.: МИКАП, 1994. – 400 с.
8. Современные микроконтроллеры: Архитектура, средства проектирования, примеры применения, ресурсы сети Интернет / Под ред. И.В. Коршуна; Составление, пер. с англ. Б.Б. Горбунова. – М.: Аким, 1998. – 272 с.
9. Enhanced RISC Microcontroller. Data Book. Atmel Corporation, 1997, p. I – V, 1.1 – 1.2, 2.1 – 2.50, 3.1 – 3.74, 4.1 – 4.88, 5.1 – 5.88, 6.1 – 6.110, 7.1 – 7.12, 8.1 – 8.6, 9.1 – 9.9.
10. 8XC51GB CHMOS Single-Chip 8-Bit Microcontroller. Preliminary. November 1994. Order Number: 272337-002. Intel Corporation, 1995, 22 p.
11. MCS-51 and MCS-96 Packaging Information. November 1994. Order Number: 272118-001. Intel Corporation, 1995, 16 p.
12. Schue R. 32-Bit Math Routines for the 8051. AB-40 Application Brief. October 1992. Order Number: 270530-002. Intel Corporation, 1996, 8 p.
13. 8XC251SA/SB/SP/SQ High-Performance CHMOS Microcontroller. Product Preview. December 1995. Order Number: 272783-002. Intel Corporation, 1995, 35 p.

Упорядники:
КИРИЧЕНКО Віталій Іванович
ЯЛАНСЬКИЙ Олексій Анатолійович
АЛПАЄВ Володимир Георгійович

МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ МПП-5 “АСЕМБЛЕР MCS-51. НАВЧАЛЬНО-НАЛАГОДЖУВАЛЬНИЙ СТЕНД EV8031/ AVR”, ІНДИВІДУАЛЬНИХ ЗАВДАНЬ ТА САМОСТІЙНОЇ РОБОТИ З ПРОФЕСІЙНО-ОРІЄНТОВАНОЇ ДИСЦИПЛІНИ “МІКРОПРОЦЕСОРНІ ПРИСТРОЇ”

для студентів спеціальності 7.092203 “Електромеханічні системи автоматизації та електропривод” напряму “Електромеханіка”

Редакційно-видавничий комплекс
Друкується у редакційній обробці упорядників

Підписано до друку . Формат 30×42/4.
Папір офсет. Ризографія. Умовн. друк. арк. .
Обліково-видавн. арк. . Тираж 100 прим. Зам. №

НГУ
49027, м. Дніпропетровськ-27, просп. К. Маркса, 19.