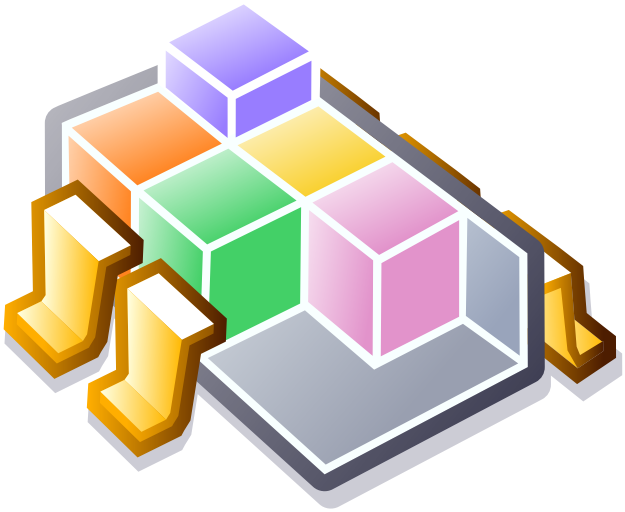
Підручник з KTechlab

David Saxton
Daniel Clarke



Підручник з KTechlab

Зміст

Анотація

KTechlab — це комплексне середовище розробки для мікроконтролерів та електроніки.

Розділ 1

Короткий огляд

1.1 Вступ

KTechlab — комплексне середовище розробки для електронних схем та мікроконтролерів. Програма здатна виконувати імітацію роботи широкого спектра компонентів (логічних, інтегрованих, лінійних, нелінійних та реактивних), імітації і діагностики роботи мікроконтролерів PIC за допомогою gpsim і постачається із власними щільно пов'язаними високорівневими мовами програмування: FlowCode та Microbe.

Програму створено простою у користуванні і ненав'язливою: для усіх компонентів і частин діаграми передбачено контекстну довідку, для імітації електроніки достатньо перетягнути і скинути компоненти на робочу область і створити з'єднання, які програма автоматично маршрутизує між контактами. FlowCode надає змогу користувачам, у яких немає значного досвіду роботи з PIC, негайно розпочати створення власних програм, а імітація електроніки надає змогу покроково виконувати програму асемблера PIC у схемі.

1.2 Документи

Для того, щоб розпочати роботу у KTechlab, вам слід створити документ, чий тип залежатиме від вашого завдання:

- Документ FlowCode — побудувати програму PIC за допомогою блок-схеми.
- Документ схеми — імітувати електронні схеми і мікроконтролери.
- Документ Microbe — високорівнева мова програмування для PIC, також використовується FlowCode для створення асемблерного коду.
- Документ асемблера — розпочати створення програми асемблера для PIC.

У KTechlab використано архітектуру «документ-перегляд», у якій логіку документа повністю відокремлено від відкритих панелей перегляду документа. Таким чином, можна використовувати декілька панелей перегляду одного файла.

Новостворений документ буде відкрито у окремій вкладці. У кожній з вкладок може бути довільна кількість панелей перегляду, поєднаних у форматі мозаїки. Це, наприклад, надає змогу імітувати роботу програми PIC у схемі, одночасно виконуючи покроково програму у документі асемблера у тій самій вкладці.

Вміст вкладок можна дублювати перетягуванням вкладки на порожню області смужки вкладок. Вставити вкладку у іншу вкладку можна перетягуванням її заголовка на відповідну вкладку.

Докладний опис згаданих вище документів можна знайти у відповідних розділах цього підручника.

1.3 Малювання

Для документів схем і FlowCode передбачено декілька засобів малювання, зокрема засіб додавання тексту. Доступ до них можна отримати натисканням кнопки з піктограмою пензля на панелі інструментів. Для малювання натисніть ліву кнопку і перетягніть вказівник миші для формування форми або лінії відповідно до використаного інструмента малювання.

Якщо позначено елемент креслення, можна змінити його розміри перетягуванням відповідних елементів керування. Утримування клавіші **Shift** під час перетягування призведе до того, що елемент керування прилипатиме до базової ґратки. Для кожного інструмента передбачено базові параметри, зокрема параметр кольору, доступ до яких можна отримати за допомогою панелі інструментів. Крім того, додаткові параметри можна змінити за допомогою бічної панелі **Редактор елементів**, зокрема стиль ліній і кінчиків ліній.

Розділ 2

Програми PIC

2.1 Керування

Після створення документа FlowCode чи текстового документа ви побачите на панелі інструментів спадне меню із піктограмою ракети. За допомогою цього меню ви можете керувати вашою програмою PIC, змінюючи її форму.

- **Перетворити на Microbe** — використовується лише у документах FlowCode. Подальший опис можна знайти у розділі розділ 4.
- **Перетворити на асемблер** — цим варіантом можна скористатися у чотирьох контекстах. Якщо відкрито документ FlowCode, буде створено команди асемблера на основі FlowCode. Якщо відкрито документ Microbe, буде викликано програму **microbe**, яка є частиною пакунка KTechlab, для компіляції програми. Так само, якщо відкрито програму мовою C, буде виконано спробу зібрати її за допомогою SDCC. Якщо відкрито текстовий документ, що містить шістнадцяткові коди PIC, буде викликано програму **gpdasm** для дизасемблювання шістнадцяткових кодів.
- **Перетворити на шістнадцятковий код** — цим варіантом також можна скористатися у чотирьох контекстах. Як і у випадку з **Перетворити на асемблер**, перетворенням можна скористатися для документів FlowCode, Microbe та C. Також цим варіантом обробки можна буде скористатися, якщо відкрито документ з кодами асемблера, для збирання його за допомогою **gpasm**.
- **Вивантажити на PIC** — за допомогою цього варіанта можна зібрати програму PIC, яку відкрито для редагування, і вивантажити її за допомогою програматора, який вибрано користувачем.

Для жодної з цих дій не потрібно зберігати документ, що дуже корисно для пришвидшення програмування. Для цілей, відмінних від PIC, після вибору одного з цих варіантів буде відкрито діалогове вікно **Виведення**, за допомогою якого можна буде вивести результат до нового документа або до якогось файла. Якщо результат зберігається до файла, програма також запропонує завантажити файл після створення і додати новостворений файл до відкритого проекту (якщо відкрито проект).

Зауважте, що ви можете наказати KTechlab завжди використовувати одну панель для виведення даних, позначивши відповідний пункт на сторінці **Загальне** параметрів роботи програми.

2.2 Вивантаження

Для вивантаження програм на PIC у KTechlab використовуються сторонні програматори. Параметри деяких типових програматорів вже визначено. Параметри інших можна додати за допомогою діалогового вікна **Параметри**.

Список портів буде отримано шляхом сканування послідовних та паралельних портів, які придатні до читання або запису даних. Пошук послідовних портів виконуватиметься у таких діапазонах:

- /dev/ttyS*[0..7]*
- /dev/tts/*[0..7]*
- /dev/ttyUSB*[0..7]*
- /dev/usb/tts/*[0..7]*

Пошук паралельних портів виконується у діапазонах:

- /dev/usb/parport*[0..7]*
- /dev/usb/parports/*[0..7]*

Розділ 3

Схеми

3.1 Розташовування компонентів

Ліворуч розташовано вкладку **Компоненти**.

Перетягування компонента з бічної панелі на панель схеми розташує його у тій точці, де перебував вказівник, коли ви відпустили ліву кнопку миші. Крім того, можна двічі клацнути лівою кнопкою миші на пункті списку бічної панелі **Компоненти**, щоб вставити декілька однакових компонентів до схеми. У цьому режимі копію позначеного компонента буде вставлено у відповідь на одинарне клацання лівою кнопкою миші, аж доки не буде натиснуто клавішу **Esc** або клацнуто правою кнопкою миші.

Щоб змінити розташування компонента, наведіть на нього вказівник миші, натисніть ліву кнопку миші і перетягніть компонент туди, куди слід. Передбачено прилипання компонентів до ґратки полотна. Якщо ви перетягнете компонент за правий або нижній край робочої області, робочу область буде відповідним чином розширено так, щоб у ній міг вміститися компонент.

З кожним компонентом пов'язано орієнтацію — компонент може бути обернуто на 0, 90, 180 або 270 градусів. Обертання не є симетричним перетворенням, тому елементи можна іще і віддзеркалювати. Щоб обернути компонент, або наведіть на нього вказівник миші, клацніть правою кнопкою і виберіть у контекстному меню пункт **Орієнтація** або натисніть одну з кнопок обертання на панелі інструментів. Віддзеркалення можна виконати натисканням клавіш **[** і **]** (подібно до того, як це робиться у Inkscape). На бічній панелі **Редактор елементів** (праворуч) передбачено кнопки для встановлення орієнтації із попереднім переглядом компонентів. Віддзеркалити компоненти можна також за допомогою бічної панелі **Редактор елементів**.

3.2 З'єднування компонентів

Передбачено два режими створення з'єднань: автоматичний і вручну. Режим можна вибрати за допомогою спадного списку **Режим маршрутизації з'єднань** панелі інструментів. Можете поекспериментувати з режимами: автоматична маршрутизація часто дає кращі результати для невеликих схем, а складні схеми можуть потребувати маршрутизації вручну.

У автоматичному режимі створити з'єднання можна так: наведіть вказівник миші на контакт компонента або наявне з'єднання, натисніть і утримуйте ліву кнопку миші, перетягніть вказівник до кінцевого контакту або з'єднання і відпустіть ліву кнопку миші. Ви бачитимете помаранчеву лінію між початковою і кінцевою точкою, якщо з'єднання є коректним. Якщо ж лінія залишається чорною, це означає, що або під вказівником миші нічого немає, або ви намагається з'єднати елементи, які вже з'єднано. У блок-схемах критерії коректності з'єднання є складнішими, їх ми обговоримо пізніше.

Найкращим способом ознайомитися із режимом з'єднування вручну є експериментування з цим режимом. Наведіть вказівник миші на початковий контакт або з'єднання, клацніть лівою

кнопкою миші і перетягніть вказівник миші від початкової точки. Щоб загнути з'єднання, клацніть лівою кнопкою миші. Щоб скасувати малювання з'єднання або натисніть клавішу **Esc**, або клацніть правою кнопкою миші.

KTechlab намагається не змінювати встановлених вами маршрутів з'єднань. Втім, якщо перетягування компонента схеми призводить до зміни розташування кінцевих точок з'єднання одна відносно одної, KTechlab у примусовому режимі перемалює з'єднання із використанням автоматичної маршрутизації. Перед пересуванням компонента ви зможете бачити з'єднання, маршрутизацію яких буде змінено. Такі з'єднання буде позначено сірим кольором.

Щоб вилучити наявне з'єднання, позначте його, намалювавши невеличку рамку над частиною з'єднання, і натисніть клавішу **Del**.

3.3 Атрибути компонентів

Для більшості компонентів передбачено можливість зміни атрибутів, наприклад опору для резисторів. Типово, ви можете редагувати прості атрибути на панелі інструментів, якщо позначено групу компонентів одного типу. Якщо позначено компоненти різних типів (наприклад, резистори і конденсатори), атрибутів для редагування показано не буде.

Деякі компоненти мають додаткові атрибути, доступ до яких не можна отримати за допомогою панелі інструментів. Ці атрибути можна змінити за допомогою бічної панелі **Редактор елементів**, розташованої праворуч. Діод, наприклад, має декілька характеристик поведінки, які можна змінити за допомогою цієї панелі.

Існує один тип атрибутів, які не можна редагувати за допомогою панелі інструментів або бічної панелі редактора елементів — багаторядковий текст. Подвійне клацання на такому елементів відкриє діалогову панель, за допомогою якої ви зможете ввести багаторядковий текст.

3.4 Імітація

Типово, одразу після створення нової схеми буде розпочато імітацію. Стан імітації буде показано у нижньому правому куті панелі схеми. Змінити його можна за допомогою меню **Інструменти**. Спершу, обговоримо те, як працює імітатор. Це надасть змогу вам скористатися ним на повну.

Після створення схеми або внесення змін до неї змінені області поділяються на групи контактів і з'єднання, які може бути розглянуто окремо. Далі, імітація для кожної групи виконується окремо (хоча групи і взаємодіють через компоненти) із визначенням способу імітації залежно від складності групи. Імітація у складних групах, що містять нелінійні компоненти, зокрема LED, є повільною. Групи, у яких містяться лише логічні контакти, з яких лише один керує значеннями у інших, є найшвидшими для імітації.

Результати імітації буде показано за допомогою декількох графічних засобів.

Для контактів компонентів програма показує індикатори напруги. Індикатори додатної напруги позначаються помаранчевим кольором, від'ємної — синім кольором. Довжина індикатора залежить від рівня напруги, а ширина відповідає силі струму, яка пропускається крізь контакт. Вимкнути індикатори можна за допомогою сторінки **Загальне** діалогового вікна **Налаштування**.

Якщо навести вказівник миші на контакт або з'єднання, програма покаже невеличку панель підказки із значеннями напруги і сили струму у відповідній точці схеми. Для деяких компонентів передбачено також графічний показ даних, наприклад для LED, вольтметрів і амперметрів.

Нарешті, є осцилоскоп, опис якого наведено у наступному розділі.

3.5 Осцилоскоп

Осцилоскоп здатен записувати дані щодо логіки, напруги та сили струму. Зонд логіки оптимізовано для зберігання булевих даних, тому ним слід користуватися замість зонду напруги, якщо вами потрібно вимірювати логічні дані у схемі.

Щоб зібрати дані, створіть новий компонент-зонд і вставте його у відповідну точку схеми. Виведені зондом дані можна буде негайно побачити на панелі осцилоскопа. Якщо додати декілька зондів, їхні дані буде розміщено на панелі осцилоскопа паралельно. Змінити розташування кривих можна за допомогою стрілочок, розташованих ліворуч від панелі осцилоскопа, а кольори ліній можна змінити у атрибутах зонда.

Для зондів напруги і сили струму діапазон вхідних значень можна визначити за допомогою бічної панелі **Редактор елементів**, розташованої праворуч.

Масштабом можна керувати за допомогою повзунка. Масштаб є логарифмічним: пересування повзунка на декілька пікселів призведе до множення величини даних на відповідний коефіцієнт. Програма KTechlab може імітувати логіку схеми з максимальною точністю у 1 мікросекунду. При максимальному масштабі одній мікросекунді відповідає 8 пікселів.

Після перетягування повзунка смужки гортання даних осцилоскопа повзунок залишатиметься у крайній правій позиції, показуючи нові записані дані. Якщо ж повзунок перебуває не у крайній правій позиції, його буде прив'язано до фіксованого моменту часу. Область перегляду на осцилоскопі можна пересувати ліворуч і праворуч перетягуванням повзунка на смужці гортання за допомогою лівої кнопки миші. Через обмеження системи віджетів гортання при максимальному масштабі може бути уривчастим.

Якщо навести вказівник миші на панель осцилоскопа і клацнути правою кнопкою, програма відкриє контекстне меню, за допомогою якого ви зможете визначити кількість кадрів (оновлень) зображення за секунду. Регулюючи цей параметр, ви зможете встановити баланс між плавністю показу і навантаженням на процесор комп'ютера.

3.6 Підсхеми

Підсхеми — зручний спосіб повторного використання схем, якщо вам цікаві лише взаємодії із зовнішніми з'єднаннями схеми. Підсхема створюється як контролер переривань, у якого контакти працюють для взаємодії із внутрішньою схемою.

Спочатку, слід побудувати схему, яку буде використано як шаблон для створення підсхеми. Точки взаємодії визначаються за допомогою компонентів **Зовнішнє з'єднання**. Їх має бути з'єднано зі схемою і розташовано там, де мають бути контакти контролера переривань підсхеми.

Далі, позначте групу компонентів і зовнішніх з'єднань, які слід перетворити на підсхему, і скористайтеся пунктом **Створити підсхему** у контекстному меню позначеного фрагмента. Програма запропонує вам ввести назву підсхеми. Після створення підсхеми цю назву буде додано на панель **Компоненти** у розділ **Підсхеми**. Працювати з новим компонентом можна буде так само, як із будь-яким звичайним компонентом. Єдиною відмінністю є те, що ви зможете клацнути правою кнопкою на пункті у списку компонентів і вибрати у контекстному меню пункт **Вилучити**.

Розділ 4

FlowCode

4.1 Вступ

За допомогою FlowCode можна дуже швидко і просто побудувати програму PIC. Після створення користувачем блок-схеми на основі доступних програмних частин KTechlab може перетворити цю блок-схему на код у одному з декількох форматів. Для виведення результатів у форматі шістнадцяткового коду, наприклад, буде виконано такий ланцюжок перетворень:

1. FlowCode буде перетворено на Microbe, код високорівневої мови програмування, компілятор якої є частиною пакунка KTechlab.
2. Після цього програма **microbe** створить на основі файла Microbe код асемблера PIC.
3. Нарешті, програма **gpasm** на основі коду асемблера PIC створить набір шістнадцяткових кодів програми PIC.

Звичайно ж, якщо у вашій системі не встановлено пакунок gputils, частиною якого є **gpasm**, останній крок виконати не вдасться.

4.2 Створення програми

У кожної програми FlowCode має бути одна точка входу — місце, з якщо програма запускатиметься після вмикання живлення PIC. Щоб визначити цю точку, відкрийте бічну панель частин діаграми, розташовану ліворуч, і перетягніть на полотно частину **Початок**. KTechlab дозволить вам створити лише одну частину такого типу.

Після цього ви зможете побудувати вашу програму, використовуючи типові частини з лівої бічної панелі або вводячи власний код (у форматі коду асемблера або Microbe) за допомогою частини **Вставка**. Порядок виконання програми керується з'єднаннями між частинами діаграми. Докладний опис створення з'єднань можна знайти у розділі Розділ 3.2.

FlowCode накладає додаткові обмеження, окрім визначених схемою, на з'єднувані частини. Наприклад, будь-яка частина діаграми може мати лише одне вихідне з'єднання. Додаткові обмеження описано у розділі Розділ 4.4.

4.3 Параметри PIC

Після створення документа FlowCode у верхньому лівому куті робочої області буде показано зображення PIC, яким ви користуєтеся. Це зображення відповідає початковим параметрам PIC.

Кожен контакт на зображенні PIC відповідає початковому типу контакту (вхід або вихід) та початковому стану (відкрито чи закрито). Ви можете змінити тип контакту перетягуванням, а його стан — клацанням лівою кнопкою миші.

За допомогою діалогового вікна **Параметри**, відкрити яке можна натисканням кнопки **Параметри**, ви можете редагувати початкові типи і стани контактів, у цьому випадку, редагуючи двійкові значення, записані до регістрів PORT і TRIS. Втім, як і визначення параметрів окремих контактів, це діалогове вікно надає змогу змінити лише початкові значення змінних у програмі PIC.

У нижній частині вікна буде показано список поточних визначених карт контактів, а також кнопки для керування цими картами. Карти контактів використовуються для визначення способу, у який сегментний індикатор або клавіатуру з'єднано з PIC. Щоб скористатися частинами FlowCode **Сегментний індикатор** або **Клавіатура**, вам слід спочатку визначити тут карту контактів.

4.4 Вкладення FlowCode

Багато частин діаграми, зокрема підпрограми і цикли, можуть містити програмний код. Після створення таких контейнерів ви зможете додати до них частини діаграми простим перетягуванням зі скиданням частин до контейнера. При цьому програма підсвічуватиме контейнер, щоб позначити, що він стане батьківським елементом для нової частини діаграми.

Контейнер є відповідальним за усі вкладені до нього частини діаграми. Якщо натиснути кнопку згортання контейнера, усі частини діаграми, що містяться у ньому, буде приховано. І навпаки, якщо знову натиснути кнопку згортання, усі ці частини знову буде показано. Ви не зможете створювати з'єднання між окремими частинами у різних контейнерах, а вміст контейнера пересуватиметься лише разом із самим контейнером.

Розділ 5

Microbe

5.1 Вступ і загальні зауваження щодо синтаксису

Microbe компілює програми, написані на нетиповій мові програмування для PIC і є супутньою програмоюKTechlab. Синтаксис мови програмування розроблено так, щоб він відповідав програмам FlowCode. Запустити **microbe** з командного рядка можна так:

```
microbe [параметри] [вхідний.microbe] [вихідний.asm]
```

Параметри роботи програми:

- `--show-source` — додати кожен з рядків початкового коду Microbe у код асемблера як коментар перед командами асемблера, що відповідають цим рядкам.
- `--no-optimize` — запобігати оптимізації команд, створених на основі коду. Зазвичай, оптимізації безпечні, тому цей параметр, в основному, корисний лише для діагностики проблем.

Файл вхідного коду .microbe має містити дані щодо PIC призначення. Такі дані вказують на початку файла .microbe. Наприклад, назвою PIC16F84 є «P16F84».

Example 5.1 Проста цілісна програма Microbe

```
P16F84

a = 0
repeat
{
        PORTA = a
        a = a + 1
}
until a == 5

end
```

5.1.1 Правила щодо назв

До назв і міток змінних застосовуються такі правила:

- Назви і мітки можуть складатися лише з цифр і латинських літер [a..z][A..Z][0..9], а також символів підкреслювання, «_».

- Назви і мітки розрізняються за регістром символів.
- Назви і мітки не можуть починатися з цифри.
- Назви і мітки не можуть починатися з «__» (подвійного підкреслювання), оскільки воно використовується компілятором.

5.1.2 Правила щодо уживання дужок

Фігурні дужки, {}, позначають початок і кінець блоку коду. Їх можна розташовувати будь-де до початку і після кінця блоку коду. Приклади прийнятних блоків коду:

```
інструкція1 {
        якийсь код
}
```

```
інструкція2 {
        ще якийсь код }
```

```
інструкція3
{
        інший код
}
```

```
інструкція5 {
        блок коду
} інструкція6
```

5.1.3 Коментування

Додавання коментарів має той самий синтаксис, що і у C. «//» робить коментарем решту символів рядка. «/*» і «*/» використовуються для багаторядкових коментарів.

```
// Це коментар
x = 2
/* А це багато-
рядковий коментар */
```

5.1.4 Структура програми

На початку програми має бути вказано ідентифікатор PIC. Основна частина програми має завершуватися командою «end». Підпрограми має бути розміщено після команди «end».

5.1.5 Підпрограми

Підпрограму можна викликати з будь-якого місця у коді. Синтаксис:

```
sub НазваПідропрограми
{
        // Код...
}
```

Підпрограму викликають командою «call *НазваПідпрограми*».

5.2 Довідник з мови Microbe

5.2.1 if

Розгалуження за умовою. Синтаксис:

```
if [вираз] then [інструкція]
```

або

```
if [вираз] then
{
        [блок інструкцій]
}
```

Те саме для else:

```
else [інструкція]
```

або

```
else
{
        [блок інструкцій]
}
```

Example 5.2 if

```
if porta.0 is high then
{
        delay 200
}
else
{
        delay 300
}
```

5.2.2 alias

Створює альтернативну назву рядка. Синтаксис:

```
alias [початкова_назва] [альтернативна_назва]
```

5.2.3 repeat

Виконувати блок інструкцій повторно, аж доки не буде виконано контрольну умову. Перевірка виконання умови відбувається після виконання блоку інструкцій, отже блок інструкцій завжди буде виконано принаймні один раз. Синтаксис:

```
repeat
{
        [блок інструкцій]
}
until [вираз]
```

5.2.4 while

Подібно до repeat надає змогу повторно виконувати блок інструкцій. Втім, контрольна умова перевіряється до виконання, а не після нього. Отже, якщо контрольна умова не виконується на першому кроці, блок інструкцій не буде виконано. Синтаксис:

```
while [вираз]
{
        [блок інструкцій]
}
```

5.2.5 goto

Призводить до продовження виконання коду з інструкції, яка є наступною після вказаної мітки. Синтаксис:

```
goto [мітка]
```

Синтаксис мітки:

```
мітка:
```

Вважається, що варто уникати використання goto. Використання інструкцій перевірки або підпрограм робить програми набагато зручнішими для читання і розуміння.

Example 5.3 goto

```
goto MyLabel

...

[MyLabel]:
// Тут продовжуватиметься виконання коду
```

5.2.6 call

Викликає підпрограму. Синтаксис:

```
call [НазваПідпрограми]
```

де *НазваПідпрограми* — назва підпрограми, яку викликають.

5.2.7 delay

Ця команда призупиняє виконання коду на вказаний період часу. Проміжок часу задається у мілісекундах. Синтаксис:

```
delay [проміжок]
```

> Примітка
> У поточній версії Microbe припускає, що PIC працює на частоті 4 МГц, тобто виконання кожної інструкції триває 1 мікросекунду. Якщо ваш мікроконтролер працює на іншій частоті, проміжок слід пропорційно скоригувати.

5.2.8 sevenseg

Використовується для визначення прив'язки контактів для (типового катодного) сегментного індикатора, який з'єднано з PIC. Синтаксис:

```
sevenseg [назва] [a] [b] [c] [d] [e] [f] [g]
```

де [a]...[g] — контакти PIC, з якими з'єднано відповідні сегменти сегментного індикатора. Контакти можна записувати або у форматі PORTX.N, або у форматі RXN.

Щоб показати цифру на сегментному індикаторі, прив'язка контактів обробляється як придатна лише для запису змінна.

Example 5.4 Визначення даних для сегментного індикатора і виведення цих даних

```
sevenseg seg1 RB0 RB1 RB2 RB3 RB4 RB5 RB6
seg1 = x + 2
```

5.2.9 keypad

Використовується для визначення карти контактів для клавіатури, яку з'єднано з PIC. Синтаксис:

```
keypad [назва] [рядок 1] ... [рядок 4] [стовпчик 1] ... [стовпчик n]
```

де [рядок 1] ... [рядок 4] і [стовпчик 1] ... [стовпчик n] — контакти PIC, з якими з'єднано відповідні рядки і стовпчики клавіш на клавіатурі (у поточній версії кількість рядків змінити не можна). Докладніший опис карт контактів наведено у розділі Розділ 5.2.8.

Стовпчики клавіатури має бути з'єднано через резистори у 100 кОм із землею. Контакти рядків має бути налаштовано як виходи, а контакти стовпчиків — як входи. Після визначення клавіатури робота з нею здійснюється за допомогою придатної лише для читання змінної.

Example 5.5 Визначення клавіатури і читання з клавіатури

```
keypad keypad1 RB0 RB1 RB2 RB3 RB4 RB5 RB6
x = keypad1
```

Типово, значеннями, які повертає клавіатура є:

- значення цифри, якщо натиснуто клавішу цифри (від 1 до 3 у верхньому рядку; шістнадцяткові цифри від A до D у четвертому рядку і далі за додатковими стовпчиками).
- 253 для клавіші у рядку 4, стовпчику 1.
- 254 для клавіші у рядку 4, стовпчику 3.

Ці значення можна перевизначити за допомогою команди alias, використовуючи назву Keypad_x_y для клавіші у рядку x, стовпчику y (нумерація рядків і стовпчиків починається з 1). Наприклад, щоб надати клавіші із зірочкою на клавіатурі 4x3 нульовий номер, можна скористатися такою командою:

Example 5.6 Прив'язування клавіші на клавіатурі до значення

```
alias Keypad_4_1 0
```

5.3 Введення-виведення даних PIC

5.3.1 Напрямок порту

Напрямок порту встановлюється встановленням значення TRIS*, де «*» — літера порту. Приклад:

Example 5.7 Встановлення напрямків портів

```
TRISB = b'01111001'
```

Вище контакти RB1, RB2 і RB7 на PORTB визначено як виходи, а інші контакти на PORTB — як входи. У цьому прикладі b'01111001' — двійкове представлення типу виходів. 1 праворуч відповідає виходу на RB0, а 0 ліворуч — входу на RB7.

5.3.2 Введення-виведення з порту

З портом можна працювати як зі змінною. Приклад:

Example 5.8 Запис до порту

```
x = PORTA
```

Наведена вище команда встановлює значення PORTA для змінної x.

5.3.3 Введення-виведення з контактів

Кожен контакт на порту можна отримати дописуванням перед номером контакту назви порту. Наприклад, контакт 2 (починаємо з контакту чи ніжки 0) на PORTA позначається як *PORTA.0*. Синтаксис встановлення стану контакту:

```
PORTX.N = СТАН
```

, де *СТАН* може приймати значення *high* (відкрито) або *low* (закрито). Синтаксис перевірки стану контакту:

```
if PORTX.N is СТАН then
```

Поєднуючи ці приклади, маємо:

Example 5.9 Встановлення і тестування стану контакту

```
TRISA = 0
TRISB = 255
if PORTA.3 is high then
{
        PORTB.5 = low
}
else
{
        PORTB = PORTA + 15
}
```

5.4 Змінні

Усі змінні є 8-бітовими невід'ємними цілими числами у діапазоні від 0 до 255. У Microbe передбачено підтримку типових унарних (над однією змінною) і бінарних (над двома змінними) дій, підтримку яких передбачено у PIC. Крім того, у Microbe передбачено підтримку ділення і множення.

5.4.1 Унарні дії

- *rotateleft x* — виконує зсув ліворуч бітів змінної x.
- *rotateright x* — виконує зсув праворуч бітів змінної x.
- *increment x* — збільшує на одиницю змінну x. Якщо значенням x було число 255, після збільшення значення буде занулено.
- *decrement x* — зменшує на одиницю змінну x. Якщо значенням x було число 0, після зменшення значенням буде 255.

5.4.2 Арифметика

Підтримувані дії:

- *Додавання:* x + y
- *Віднімання:* x - y
- *Множення:* x * y
- *Ділення:* x / y
- *Двійкове виключне АБО:* x XOR y
- *Двійкове І:* x AND y
- *Двійкове АБО:* x OR y

5.4.3 Порівняння

Підтримувані дії:

- *Рівність:* x / y
- *Нерівність:* x != y
- *Є більшим:* x > y
- *Є меншим:* x < y
- *Є більшим або рівним:* x >= y
- *Є меншим або рівним:* x <= y

Приклад:

Example 5.10 Порівняння

```
if PORTA
>= 5 then
{
        ...
}
```

Розділ 6

Діагностика

6.1 Запуск засобу діагностики

Підтримку діагностики передбачено для коду асемблера, SDCC та Microbe, якщо код відкрито як текстовий документ. Після цього покрокове виконання керується меню **Діагностика**. Передбачено два способи запуску засобу діагностики.

Якщо програма PIC вже працює у схемі, подвійне клацання на компоненті PIC відкриє цю програму. Для програм PIC мовою асемблера засіб діагностики для текстового документа програми буде пов'язано з компонентом PIC. У цьому випадку меню діагностики не зможе зупинити виконання програми PIC, оскільки її власником є компонент PIC.

Якщо файл асемблерного коду вже відкрито, засіб діагностики можна запустити за допомогою меню **Діагностика**. Після компіляції програми засіб діагностики буде готовий до роботи, а програму PIC буде призупинено на першій команді. Зауважте, що під час роботи із кодом високорівневими мовами програмування поточну точку виконання не буде показано, якщо немає рядка, який відповідає першій команді асемблера, яку слід виконати. У цьому випадку натискання кнопки **Далі** переведе точку виконання на перший рядок програми.

6.2 Керування засобом діагностики

Засіб діагностики може працювати у одному з двох режимів: виконання і покрокове виконання. У режимі виконання робота програми PIC імітуватиметься у режимі реального часу. Для покрокового виконання роботу програми PIC має бути призупинено або натисканням пункту **Перервати** у меню **Діагностика**, або натисканням кнопки призупинення на компоненті PIC.

У режимі покрокового виконання наступний рядок, який має бути виконано буде позначено зеленою стрілочкою на полях редактора текстового документа (так, як цей рядок позначається і у KDevelop). Вам варто увімкнути показ полів (смужки піктограм) за допомогою меню **Перегляд** (увімкнути показ полів для типового документа можна за допомогою діалогового вікна **Параметри редактора**).

Передбачено три типи покрокового виконання:

- **Крок** — виконати поточну команду. Зелену стрілочку буде пересунуто на наступний рядок, який має бути виконано.
- **Переступити** — якщо наступною командою є виклик підпрограми або подібна команда, вибір цього варіанта призведе до «переступання» виклику і повернення до режиму покрокового виконання, щойно обробку виклику буде завершено. У інших випадках «переступання» працює як звичайне виконання кроку-команди. З технічної точки зору «переступання» виглядає так — записується початковий рівень стека і виконання програми призупиняється, аж доки стек на повернеться на початковий рівень.

- **Вийти** — якщо виконуються команди у межах виклику, неперервне виконання буде продовжено до завершення виклику. Подібно до переступання, ця дія еквівалентна до очікування моменту, коли рівень стека повернеться до значення, яке на одиницю менше за початковий рівень, якщо початковий рівень був ненульовим.

Точки зупинки надають змогу зупинити виконання, коли виконання програми PIC досягає заданої команди. Щоб увімкнути або вимкнути точку зупинки на рядку, де розміщено курсор, або скористайтеся меню **Діагностика** або клацніть на бічній смужці текстового документа.

На бічній панелі **Перегляд символів**, розташованій праворуч, буде показано значення спеціальних функціональних регістрів. Щоб визначити значення змінної у регістрах загального вжитку, ви можете навести вказівник миші на назву змінної у команді, яка працює з цим регістром. Зауважте, що вибір основи числення на панелі **Перегляд символів** визначить і спосіб показу значення, якщо ви наведете вказівник миші на змінну.

Розділ 7

Поширені питання

1. *KTechlab значно навантажує процесор*

   Причин може бути декілька. Імітація схем, які містять реактивні та нелінійні компоненти (зокрема конденсатори та транзистори), значно навантажує процесор. Ви можете призупинити або поновити імітацію за допомогою відповідних пунктів меню **Інструменти**.

   Малювання робочої області (зокрема перемальовування багатьох швидкооновлюваних смужок напруги на контактах) є доволі вибагливою до обчислювальних ресурсів процедурою. Ви можете зменшити частоту оновлення зображення або вимкнути індикатори напруги за допомогою діалогового вікна **Налаштування** програми. Частоту оновлення зображення **Осцилоскопа** також можна зменшити: клацніть правою кнопкою миші на ньому і виберіть бажану частоту.

   Зауважте, що наступна основна версія KTechlab може бути набагато швидшою як у показі робочої області, такі і в імітації активних та нелінійних компонентів.