МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Смірнов В.В., Смірнова Н.В., Пархоменко Ю.М.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ УПРАВЛЯЮЧИХ МІКРО-ЕОМ

Навчальний посібник

Кропивницький - 2021

УДК: 004.41

ББК 32.97

С50

Рекомендовано Вченою радою Центральноукраїнського національного технічного університету до друку та використанню навчального посібника у навчальному процесі здобувачами вищої освіти очної та заочної форми навчання за спеціальністю 123 «Комп'ютерна інженерія», протокол № 8 від 29 березня 2021 року

Рецензенти:

Павленко Іван Іванович, доктор технічних наук, професор, завідувач кафедрою Технології машинобудування Центральноукраїнського національного технічного університету.

Віхрова Лариса Григорівна, кандидат технічних наук, професор, декан факультету Автоматики та Енергетики Центральноукраїнського національного технічного університету.

С50

Смірнов В.В., Смірнова Н.В., Пархоменко Ю.М.
Програмне забезпечення управляючих мікро - ЕОМ : навчальний посібник ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2021. – 217 с.

У навчальному посібнику описані архітектура і вбудовані спеціалізовані модулі PIC – мікроконтролерів серії PIC18FXX2, методи та засоби для їх програмування. Представлені апаратно-програмні комплекси для розробки та програмування мікроконтролерних систем управління різними технологічними процесами, територіально-розподіленими системами, роботами, об'єктами і комплексами на базі PIC – мікроконтролерів.

Представлені практичні рішення навчальних завдань для кращого освоєння досліджуваного матеріалу, представлені варіанти навчальних завдань для самостійного придбання практичних навичок.

Навчальний посібник призначений для здобувачів вищої освіти очної та заочної форми навчання за спеціальністю 123 «Комп'ютерна інженерія».

Навчальне електронне видання комбінованого використання.
Можна використовувати в локальному та мережному режимах

УДК: 004.41

ББК 32.97

© В.В. Смірнов, Н.В. Смірнова, Ю.М. Пархоменко, 2021
© ЦНТУ, кафедра «Програмування комп'ютерних систем і мереж»

Зміст

ВСТУП

Дисципліна **«Програмне забезпечення управляючих мікро-ЕОМ»** займає важливе місце серед інших навчальних дисциплін, які розширюють і поглиблюють знання, вміння і навички, отримані студентами в результаті освоєння матеріалів даної дисципліни.

Завданням навчальної дисципліни є підготовка фахівців в області розробки і програмування мікроконтролерних систем управління різними технологічними процесами, територіально-розподіленими системами, роботами, об'єктами і комплексами.

В основі побудови сучасних технічних систем лежить автоматизація процесів, контроль та управління станом систем за допомогою однокристальних мікроконтролерів (МК, MCU - MicroController Unit).

На відміну від мікропроцесорів МК включають всі пристрої, потрібні для реалізації цифрових систем управління: процесор, оперативна пам'ять, пам'ять команд, електрично програмована постійна пам'ять, внутрішній генератор тактових сигналів, АЦП, ЦАП, пристрій для зв'язку із зовнішнім середовищем, а також інші спеціалізовані модулі.

Тому мікроконтролери дозволяють реалізувати широке коло завдань управління об'єктами різного призначення.

Існує велика кількість МК різного рівня складності, які виготовляються провідними фірмами. Одним з важливих чинників в освоєнні технології розробки та програмування систем на базі МК є так званий «поріг входження», який визначається ступенем складності в освоєнні апаратної і програмної складової оточення МК.

Найнижчим «порогом входження» для освоєння МК мають мікроконтролери фірми **Microchip Technology Inc.** спільно з інтегрованим середовищем розробки програм **(IDE) PSWH** фірми Custom Computer Services Inc.

Завданням навчального посібника є швидке навчання студентів навичкам роботи з мікроконтролерами у рамках виділених кредитів за допомогою спеціально розроблених апаратно-програмних комплексів які призначені для створення, програмування, налагодження мікроконтролерних систем управління і виконання лабораторних робіт в рамках дисципліни **«Програмне забезпечення управляючих мікро ЕОМ»**, а також для підготовки програмістів в області розробки і управління територіально-розподіленими системами, роботами, об'єктами і комплексами.

Навчальний посібник призначений для здобувачів вищої освіти очної та заочної форми навчання за спеціальністю 123 «Комп'ютерна інженерія».

РОЗДІЛ 1. ОПИС НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

«ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ УПРАВЛЯЮЧИХ МІКРО-ЕОМ»

Анотація до дисципліни

Дисципліна «**Програмне забезпечення управляючих мікро-ЕОМ**» викладається відповідно до навчального плану підготовки бакалаврів:

галузі знань: 12 Інформаційні технології;

спеціальності: 123 «Комп'ютерна інженерія»;

спеціалізації: «Комп'ютерні системи та мережі».

Дисципліна відноситься до категорії **вибіркових**.

Форма підсумкового контролю: залік.

Мета і завдання дисципліни

Основна мета дисципліни полягає в придбанні досконалих знань і навичок роботи з апаратним та програмним забезпеченням систем управління об'єктом на базі мікро-ЕОМ.

Компетентності, якими повинен оволодіти здобувач

В результаті засвоєння лекційного матеріалу і виконання лабораторних робіт студенти повинні отримати теоретичні знання та методику ефективної роботи з сучасними системами управління на базі мікро-ЕОМ.

Завдання вивчення дисципліни

- вивчення теоретичних основ систем управління;
- вивчення теоретичних основ методів управління;
- вивчення теоретичних основ методів перетворення сигналів;
- вирішення завдань введення - виведення дискретних і аналогових сигналів;
- вирішення завдань управління об'єктами;

- набуття практичних навиків в сфері програмування систем управління на базі мікро-ЕОМ.

Предметом навчальної дисципліни є архітектура мікроконтролерів і програмне забезпечення мікроконтролерів для систем управління об'єктом.

Результати навчання

У результаті вивчення навчальної дисципліни студент повинен:

знати:

- особливості роботи з дискретними і аналоговими сигналами;
- процес управління об’єктами на базі мікро-ЕОМ;
- процедуру програмування систем управління на базі мікро-ЕОМ;

вміти:

- вирішувати завдання введення/виведення дискретних і аналогових сигналів;
- проводити управління об’єктами на базі мікро-ЕОМ;
- розробляти прикладні та системні програми для систем управління об’єктами на базі мікро-ЕОМ;

набути соціальних навичок (soft-skills):

- здійснювати професійну комунікацію;
- ефективно пояснювати і презентувати матеріал;
- взаємодіяти в проектній діяльності.

Теми лабораторних робіт

Таблиця 1.1 - Теми лабораторних робіт

№ з/п	Назва теми
№ 1 - «IN_OUT»	Введення/виведення дискретних сигналів
№ 2 - «INTERRUPTS»	Робота з перериваннями мікроконтролера
№ 3 - «ADC»	Робота з АЦП
№ 4 - «DAC»	Робота з ЦАП

КРИТЕРІЇ ОЦІНЮВАННЯ ЗНАНЬ ЗДОБУВАЧІВ ВИЩОЇ ОСВІТИ

Реалізація основних завдань контролю знань здобувачів вищої освіти у ЦНТУ досягається системними підходами до оцінювання та комплексністю застосування різних видів контролю.

Згідно з діючою в університеті системою комплексної діагностики знань, з метою стимулювання планомірної та систематичної навчальної роботи, оцінка знань здобувачів вищої освіти здійснюється за 100-бальною системою.

Форми контролю знань здобувачів вищої освіти:

- поточний;
- рубіжний;
- семестровий підсумковий (залік, екзамен).

Оцінювання знань здобувачів вищої освіти в університеті здійснюється за 100-бальною шкалою, яка переводиться відповідно у:

національну шкалу:

- **«відмінно»**;
- **«добре»**;
- **«задовільно»**;
- **«незадовільно»**,

та шкалу європейської кредитно-трансферної системи ЄКТС:

- **A**; **B**; **C**; **D**; **E**; **FX**; **F**.

Поточний контроль проводиться на кожному семінарському, практичному/лабораторному занятті та за результатами виконання завдань самостійної роботи. Він передбачає оцінювання теоретичної підготовки здобувачів вищої освіти із зазначеної теми (у тому числі, самостійно опрацьованого матеріалу) під час роботи на семінарських заняттях та набутих практичних навичок під час виконання завдань лабораторних/практичних робіт.

Критерії поточного оцінювання знань здобувачів вищої освіти

Таблиця 1.2 - Критерії поточного оцінювання знань здобувачів вищої освіти

Усний виступ та виконання письмового завдання, тестування (бали)	**Критерії оцінювання**
5	В повному обсязі володіє навчальним матеріалом, вільно самостійно та аргументовано його викладає під час усних виступів та письмових відповідей, глибоко та всебічно розкриває зміст теоретичних питань та практичних завдань, використовуючи при цьому обов'язкову та додаткову літературу. Правильно вирішив усі тестові завдання.
4	Достатньо повно володіє навчальним матеріалом, обґрунтовано його викладає під час усних виступів та письмових відповідей, в основному розкриває зміст теоретичних питань та практичних завдань, використовуючи при цьому обов'язкову літературу. Але при викладанні деяких питань не вистачає достатньої глибини та аргументації, допускаються при цьому окремі несуттєві неточності та незначні помилки. Правильно вирішив більшість тестових завдань.

Продовження таблиці 1.2

Усний виступ та виконання письмового завдання, тестування (бали)	**Критерії оцінювання**
3	В цілому володіє навчальним матеріалом викладає його основний зміст під час усних виступів та письмових відповідей, але без глибокого всебічного аналізу, обґрунтування та аргументації, без використання необхідної літератури допускаючи при цьому окремі суттєві неточності та помилки. Правильно вирішив половину тестових завдань.
2	Не в повному обсязі володіє навчальним матеріалом. Фрагментарно, поверхово (без аргументації та обґрунтування) викладає його під час усних виступів та письмових відповідей, недостатньо розкриває зміст теоретичних питань та практичних завдань, допускаючи при цьому суттєві неточності, правильно вирішив меншість тестових завдань.
1	Частково володіє навчальним матеріалом не в змозі викласти зміст більшості питань теми під час усних виступів та письмових відповідей, допускаючи при цьому суттєві помилки. Правильно вирішив окремі тестові завдання.
0	Не володіє навчальним матеріалом та не в змозі його викласти, не розуміє змісту теоретичних питань та практичних завдань. Не вирішив жодного тестового завдання

Доповнення виступу:

2 бали – отримують здобувачі вищої освіти, які глибоко володіють матеріалом, чітко визначили його зміст; зробили глибокий системний аналіз змісту виступу, виявили нові ідеї та положення, що не були розглянуті, але суттєво впливають на зміст доповіді, надали власні аргументи щодо основних положень даної теми.

1 бал - отримують здобувачі вищої освіти, які виклали матеріал з обговорюваної теми, що доповнює зміст виступу, поглиблює знання з цієї теми та висловили власну думку.

Суттєві запитання до доповідачів:

1 бал - отримують здобувачі, які своїм запитанням до виступаючого суттєво і конструктивно можуть доповнити хід обговорення теми.

0,5 балів - отримують здобувачі вищої освіти, які у своєму запитанні до виступаючого вимагають додаткової інформації з ключових проблем теми, що розглядається.

Експрес-контроль:

1 бал - нараховуються здобувачам вищої освіти, які вільно володіють усім навчальним матеріалом, орієнтуються в темі та аргументовано висловлюють свої думки.

0,5 балів - отримують здобувачі вищої освіти, які частково володіють матеріалом та можуть окреслити лише деякі проблеми теми.

Ведення опорного конспекту лекції:

Опорний конспект лекції *(ОКЛ)* – вид навчально-методичного посібника, в якому у стисло і системно викладено основний теоретичний матеріал у формі основних понять і положень, що структурно й логічно пов’язані між собою.

Кожен здобувач повинен мати ОКЛ на лекціях і вести в ньому записи власноруч. Під час аудиторної роботи з ОКЛ здобувачі вищої освіти записують

основні тези лекції та пояснення викладача. Під час самостійної роботи рекомендується доповнити записи лекції.

1 бал нараховується здобувачам вищої освіти, які в повному обсязі самостійно і творчо опрацювали всі питання лекції і вільно володіють її змістом.

0,5 балів нараховується здобувачам вищої освіти, які опрацювали лише окремі питання лекції і не достатньо вільно володіють її змістом.

Рубіжний контроль знань здобувачів вищої освіти

Рубіжний контроль успішності здобувачів вищої освіти − це об'єктивна оцінка міри освоєння здобувачами вищої освіти денної форми навчання програм навчальних дисциплін; результатів у здобутті знань, дотримання навчальної дисципліни.

Рубіжний контроль успішності має на меті підвищення мотивації до навчання і свідомої навчальної дисципліни здобувачів вищої освіти.

Рубіжний контроль успішності здобувачів вищої освіти проводиться науково-педагогічними працівниками під час проведення всіх видів аудиторних занять з усіх дисциплін по завершеним темам всередині семестру та в останній тиждень семестру.

Оцінка рубіжного контролю носить комплексний характер і враховує досягнення здобувача вищої освіти по основних компонентах, які визначені робочою програмою навчальної дисципліни:

- рівень засвоєння навчального матеріалу;
- повнота виконання здобувачем вищої освіти усіх видів робіт, передбачених навчальною програмою дисципліни;
- відвідування занять;
- робота з дистанційними курсами на сайті дистанційної освіти ЦНТУ;
- самостійна робота здобувача вищої освіти;
- дослідницька робота тощо.

Результати поточних та рубіжних контролів є складовими оцінки семестрового підсумкового контролю.

Результати рубіжного контролю успішності з усіх дисциплін фіксуються викладачами двічі на семестр у встановлені графіком освітнього процесу терміни у факультетських журналах результатів рубіжного контролю і доводяться до відома кураторів академічних груп, обговорюються на засіданнях кафедр, рад факультетів

Загальна максимальна кількість балів, виділених для оцінки результатів під час одного рубіжного контролю робочою програмою навчальної дисципліни, при семестровому підсумковому контролі:

- у формі заліку складає 50 балів;
- у формі екзамену складає 30 балів.

Критерії рубіжного оцінювання знань здобувачів вищої освіти

Таблиця 1.3 - Критерії рубіжного оцінювання знань здобувачів вищої освіти

Загальна кількість балів	**Критерії оцінювання**
25 - 30	В повному обсязі володіє навчальним матеріалом, вільно самостійно та аргументовано його викладає під час усних та письмових відповідей глибоко та всебічно розкриває зміст теоретичних питань та практичних завдань, використовуючи при цьому обов’язкову та додаткову літературу. Правильно вирішив усі тестові завдання.

Продовження таблиці 1.3

Загальна кількість балів	**Критерії оцінювання**
21 - 24,5	Достатньо повно володіє навчальним матеріалом, обґрунтовано його викладає під час усних виступів та письмових відповідей, в основному розкриває зміст теоретичних питань та практичних завдань, використовуючи при цьому обов'язкову літературу. Але при викладанні деяких питань не вистачає достатньої глибини та аргументації, допускаються при цьому окремі несуттєві неточності та незначні помилки. Правильно вирішив більшість тестових завдань.
17 - 20,5	В цілому володіє навчальним матеріалом викладає його основний зміст під час усних виступів та письмових відповідей, але без глибокого всебічного аналізу, обґрунтування та аргументації, без використання необхідної літератури допускаючи при цьому окремі суттєві неточності та помилки. Правильно вирішив половину тестових завдань
12 - 16,5	Не в повному обсязі володіє навчальним матеріалом. Фрагментарно, поверхово (без аргументації та обґрунтування) викладає його під час усних виступів та письмових відповідей, недостатньо розкриває зміст теоретичних питань та практичних завдань, допускаючи при цьому суттєві неточності, правильно вирішив меншість тестових завдань.
10 - 15	Частково володіє навчальним матеріалом не в змозі викласти зміст більшості питань теми під час усних виступів та письмових відповідей, допускаючи при цьому суттєві помилки. Правильно вирішив окремі тестові завдання.
0	Не володіє навчальним матеріалом та не в змозі його викласти, не розуміє змісту теоретичних питань та практичних завдань. Не вирішив жодного тестового завдання.

Сума балів, накопичених здобувачем вищої освіти за виконання всіх видів поточних навчальних завдань (робіт) на практичних (семінарських) заняттях та на підсумковому рубіжному контролі, свідчить про ступінь оволодіння ним програмою навчальної дисципліни на конкретному етапі її вивчення.

Протягом семестру здобувачі вищої освіти можуть набрати від 0 до 100 балів, що переводяться у національну шкалу оцінювання і відповідно у шкалу ЄКТС.

Кількість балів відповідає певному рівню засвоєння дисципліни:

Таблиця 1.4 - Шкала оцінювання: національна та ЄКТС

За системою ЦНТУ	**За шкалою ECTS**	**За національною системою**	**Визначення**
90 - 100	A	5 (відмінно)	Повно та ґрунтовно засвоїв всі теми навчальної програми вміє вільно та самостійно викласти зміст всіх питань програми навчальної дисципліни, розуміє її значення для своєї професійної підготовки, повністю виконав усі завдання кожної теми та рубіжного контролю в цілому. Брав участь в олімпіадах, конкурсах, конференціях.

Продовження таблиці 1.4

За системою ЦНТУ	**За шкалою ECTS**	**За національною системою**	**Визначення**
82 - 89	B	4 (дуже добре)	Недостатньо повно та ґрунтовно засвоїв окремі питання робочої програми. Вміє самостійно викласти зміст основних питань програми навчальної дисципліни, виконав завдання кожної теми та рубіжного контролю в цілому.
74 - 81	C	4 (добре)	Недостатньо повно та ґрунтовно засвоїв деякі теми робочої програми, не вміє самостійно викласти зміст деяких питань програми навчальної дисципліни. Окремі завдання кожної теми та рубіжного контролю в цілому виконав не повністю
64 - 73	D	3 (задовільно)	Засвоїв лише окремі теми робочої програми. Не вміє вільно самостійно викласти зміст основних питань навчальної дисципліни, окремі завдання кожної теми рубіжного контролю не виконав.
60 - 63	E	3 (достатньо)	Засвоїв лише окремі питання навчальної програми. Не вміє достатньо самостійно викласти зміст більшості питань програми навчальної дисципліни. Виконав лише окремі завдання кожної теми та рубіжного контролю в цілому.

Продовження таблиці 1.4

За системою ЦНТУ	За шкалою ECTS	За національною системою	Визначення
> 60	Fx	2 (незадовільно)	Не засвоїв більшості тем навчальної програми не вміє викласти зміст більшості основних питань навчальної дисципліни. Не виконав більшості завдань кожної теми та рубіжного контролю в цілому.

Підсумковий семестровий контроль

Семестровий підсумковий контроль проводиться з метою визначення рівня досягнення здобувачами вищої освіти запланованих результатів навчання, що визначені робочою програмою навчальної дисципліни (практики). Здобувач вищої освіти вважається допущеним до семестрового підсумкового контролю з конкретної навчальної дисципліни (семестрового екзамену, диференційованого заліку або заліку), якщо він виконав усі види робіт, які передбачені навчальним планом на відповідний семестр з цієї навчальної дисципліни, та виконав умови контракту.

Семестровий підсумковий контроль проводиться у формі екзамену, диференційованого заліку чи заліку, що визначено навчальним планом, у терміни, передбачені графіком освітнього процесу. Зміст екзаменів і заліків визначається робочими навчальними програмами дисциплін.

У випадку проведення семестрового підсумкового контролю у формі заліку, кожен з видів роботи (завдань), виконаних здобувачем вищої освіти протягом семестру, оцінюється визначеною кількістю балів відповідно до схеми нарахування балів, що представлена в робочій програмі навчальноїдисципліни. Здобувачі вищої освіти мають бути повідомлені про кількість набраних ними балів до початку екзаменаційної сесії.

Семестровий екзамен − це форма підсумкового семестрового контролю, що полягає в оцінці засвоєння здобувачем вищої освіти теоретичного та практичного навчального матеріалу з певної навчальної дисципліни протягом семестру, результати навчання за яким оцінюються за стобальною та чотирьохбальною шкалами оцінювання.

Екзамени складаються здобувачами вищої освіти з відповідних дисциплін, які передбачені навчальним планом, в період екзаменаційних сесій. Семестрові екзамени проводяться в письмовій, усній та тестовій формі.

Екзамен може завершуватись усною співбесідою зі здобувачами вищої освіти, їх відповідями на додаткові запитання.

Зміст, обсяг, структура, форма екзаменаційної роботи, система і критерії її оцінювання визначаються робочою програмою дисципліни. На початку семестру науково-педагогічний працівник повинен ознайомити здобувачів вищої освіти зі змістом, структурою, формою екзаменаційної (залікової) роботи та прикладами завдань. Обсяг матеріалу, що виноситься на підсумковий контрольний захід, має охоплювати весь зміст дисципліни відповідно до її робочої програми.

Оцінку підсумкового семестрового контролю у формі екзамену становить сума балів за результатами рубіжних контролів та балів, набраних здобувачем вищої освіти при складанні семестрового екзамену. Загальна кількість балів, виділених на проведення семестрового екзамену робочою програмою навчальної дисципліни, складає 40 балів. Кількість балів, одержана здобувачем вищої освіти на екзамені, додається до результатів рубіжних контролів, що разом складає оцінку знань здобувача вищої освіти з навчальної дисципліни за 100-бальною шкалою та переводиться в оцінку за шкалою ЄКТС і національною шкалою (“Відмінно”, “Добре”, “Задовільно”, “Незадовільно”).

Семестровий залік полягає в оцінці рівня засвоєння здобувачем вищої освіти навчального матеріалу на лекційних, практичних, семінарських або лабораторних заняттях і виконання індивідуальних завдань за стобальною та дворівневою («зараховано», «не зараховано») шкалою оцінювання результатів

навчання. Семестровий залік планується при відсутності екзамену. Семестровий залік з окремої дисципліни проводиться на останньому занятті, до початку екзаменаційної сесії.

Навчальний план передбачає при вивченні навчальної дисципліни виконання певних видів робіт на лекційних, практичних, семінарських, лабораторних заняттях, виконання індивідуальних завдань, інших видів навчальної діяльності, тому оцінка здобувачам вищої освіти вище 60 балів може виставлятися без виконання ними підсумкової залікової роботи. В такому разі виставлення оцінки підсумкового семестрового контролю не передбачає обов'язкової присутності здобувача вищої освіти на заліку. У разі, якщо сума рейтингових балів менша ніж 60, але виконані умови допуску до семестрового контролю, здобувач вищої освіти виконує на останньому за розкладом занятті залікову контрольну роботу. За бажанням, здобувач вищої освіти має право на виконання залікової контрольної роботи з метою підвищення кількості балів, які були набрані ним протягом семестру.

Заліки приймаються науково-педагогічними працівниками, які проводили практичні, семінарські та інші заняття в академічній групі або читали лекції з даної дисципліни.

Семестровий диференційований залік − це форма підсумкового контролю, що полягає в оцінці засвоєння здобувачем вищої освіти навчального матеріалу з певної дисципліни виключно на підставі результатів виконаних індивідуальних завдань (розрахункових, графічних, під час проходження практики тощо).

Семестровий диференційований залік може плануватися при відсутності екзамену з даної навчальної дисципліни. Здобувачі вищої освіти, які набрали за результатами поточного контролю менше мінімальної кількості балів, необхідної для виставлення заліку, допускаються до семестрового контролю після перескладання контрольних заходів, що проводилися в межах рубіжних контролів.

Здобувачі вищої освіти заочної форми навчання допускаються до семестрового контролю, якщо вони своєчасно виконали завдання із самостійної роботи з навчальних дисциплін семестру.

При складанні заліку оцінка підсумкового семестрового контролю виставляється як сума балів, набраних здобувачем вищої освіти за рубіжними контролями. У разі, якщо сума рентингових балів менша за 60, але виконані умови допуску до семестрового контролю з цієї навчальної дисципліни, здобувач вищої освіти виконує на останньому за розкладом занятті залікову контрольну роботу.

РОЗДІЛ 2. ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МІКРОКОНТРОЛЕРІВ

АПАРАТНО-ПРОГРАМНІ КОМПЛЕКСИ

Для виконання лабораторних робіт на базі **PIC – мікроконтролерів**, були розроблені **апаратно-програмні комплекси (АПК)** (рис 2.1, 2.2).

Рисунок 2.1 – Апаратно-програмний комплекс № 1

Апаратно-програмні комплекси призначені для створення та відлагодження програмного забезпечення для різних мікроконтролерних систем у рамках дисципліни **«Програмне забезпечення управляючих мікро-ЕОМ»** та **«Програмування мікроконтролерних систем»**.

Рисунок 2.2 – Апаратно-програмний комплекс № 2

До складу Апаратно-програмного комплексу № 1 (рис. 2.1) входить:

- базовий інтегрований апаратно-програмний модуль;
- автономний апаратно-програмний модуль;
- модуль-емулятор об'єкта управління;
- навчальна програма дисципліни;
- завдання і методичні вказівки до виконання лабораторних робіт;
- приклади виконання завдань;
- інтегроване середовище розробки програмного забезпечення на мові програмування **C**;
- драйвер для завантаження бінарного коду у FLASH-пам'ять програм мікроконтролера по інтерфейсу **ICSP**.

Завдання, які вирішуються за допомогою Апаратно-програмного комплексу № 1:

1) ознайомлення з мікроконтролерами і засобами розробки:
   - введення/виведення дискретних сигналів;
2) виведення інформації на LCD-дисплей;
3) робота з перериваннями мікроконтролера;
4) введення/виведення даних по інтерфейсу **RS-232**;
5) введення і обробка аналогових сигналів за допомогою АЦП:
   - організація введення з цифрової клавіатури;
6) виведення даних через зовнішній ЦАП по інтерфейсу **SPI**:
   - створення драйвера інтерфейсу **SPI**;
7) виведення інформації на семисегментний LED-дисплей:
   - створення драйвера динамічної індикації;
8) робота з модулями **PWM** (ШІМ):
   - управління потужністю у навантаженні;
   - управління двигуном постійного струму. Напрямок та швидкість обертання;
9) робота з зовнішньої **EEPROM** пам'яттю по інтерфейсу **I^2C**:
   - запис/читання байтів і блоків даних;
10) розробка ПІД-регулятора:
   - розрахунок параметрів ПІД-регулятора за методикою Ніколса;
   - оптимізація ПІД-регулятора;
   - управління об'єктом;
11) управління сервоприводом.

Структура Апаратно-програмного комплексу № 2

Апаратно-програмний комплекс № 2 (рис. 2.2) має наступну структуру (рис. 2.3):

- контролери;
- пристрої введення/виведення;
- зовнішні пристрої;
- операційні системи;
- мережеві протоколи;
- інтерфейси зв'язку з об'єктами і пристроями;
- бездротові інтерфейси;
- об'єкти управління.

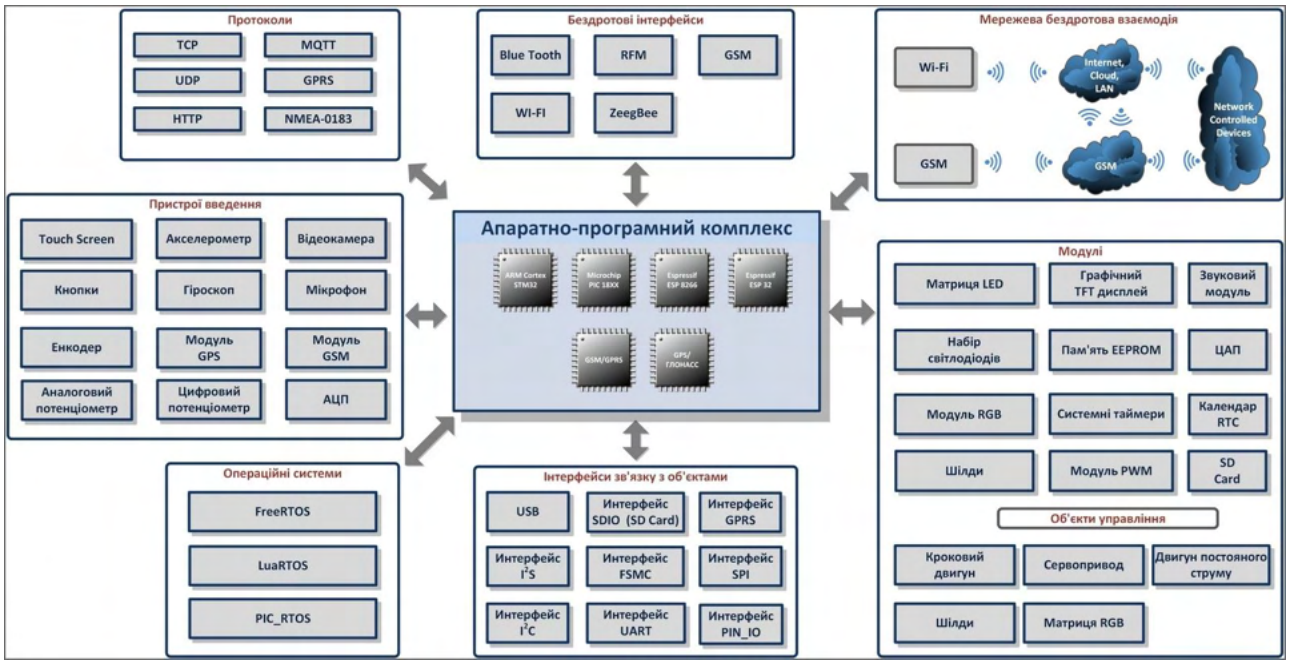

Рис. 2.3 - Структура Апаратно-програмного комплексу № 2

Склад апаратної частини комплексу № 2 (рис. 2.2)

1. Модулі контролерів:

- PIC 18F25K22;
- STM 32F103C8T6;
- ESP 32;
- ESP 8266 (рис. 2.4).

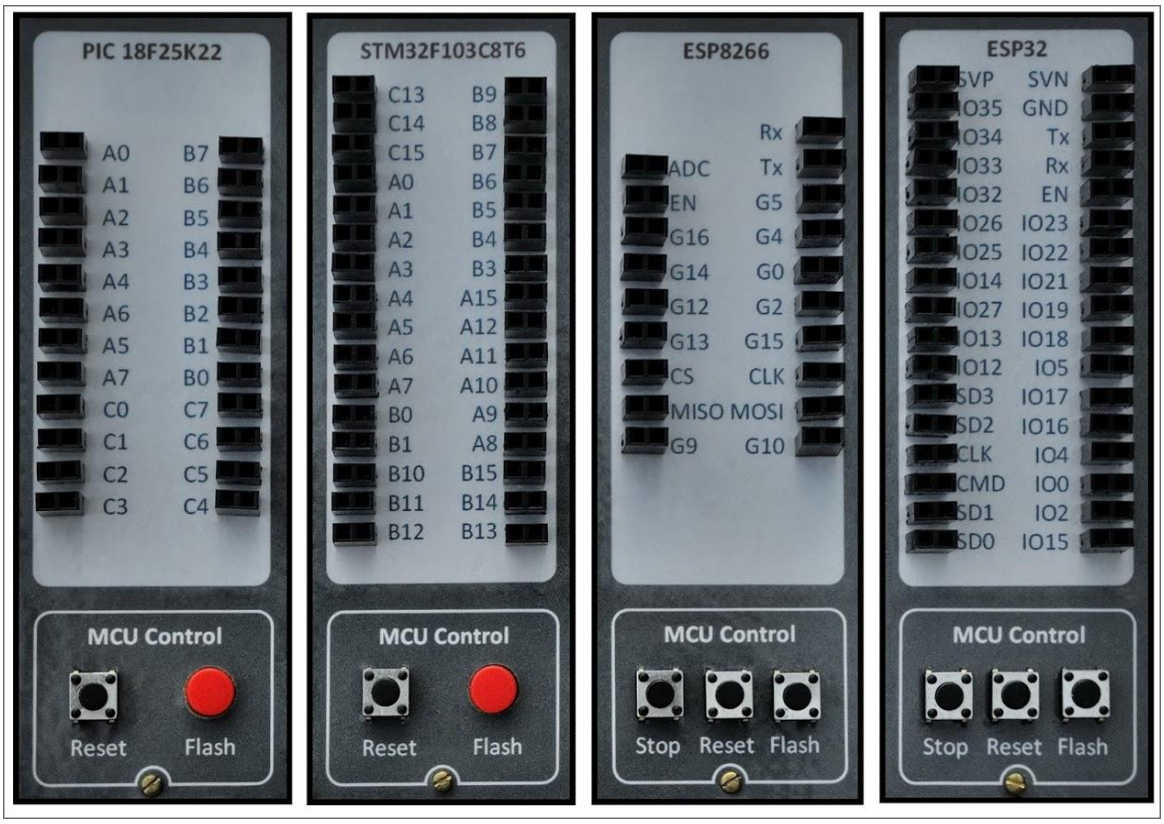

Рис. 2.4 - Змінні модулі контролерів

Передбачене встановлення будь-яких інших контролерів і програмованої логічної матриці. (ПЛМ).

2. Мережеві комунікаційні модулі:

- модуль Wi-Fi: ESP 8266;
- модуль Wi-Fi: ESP 32;
- модуль GSM: SIM800L.

3. Допоміжні модулі:

- гіроскоп: L3G4200D;
- акселерометр: ADXL345;
- модуль GPS: M8030 NEO-M8N;
- модуль SD Card (read/write).

4. Пристрої введення:

- TouchScreen SPI (TFT дисплей);
- аналоговий потенціометр;
- цифровий потенціометр: X9C103SZI;
- цифровий енкодер;
- набір кнопок;
- набір перемикачів;
- модуль UART;
- модуль Wi-Fi;
- модуль GSM;
- модуль GPS.

5. Пристрої виведення:

- TFT SPI дисплей;
- OLED I^2C serial дисплей;
- набір світлодіодів;
- набір RGB світлодіодів;
- матриця LED 8x8 SPI-UART;
- модуль Wi-Fi;
- модуль GSM.

6. Об'єкти управління:

- двигун постійного струму;
- кроковий двигун;
- сервопривід.

Розширення:

Для розробки і/або підключення інших модулів передбачена макетна панель і клеми розширення.

Склад програмної частини комплексу № 2 (рис. 2.2)

1. Мережеві модулі:

- FTP Server, Client;
- HTTP Server, Client;
- MQTT Client;
- Websocket Client;
- Redis Client;
- Net (UDP, TCP);
- CJSON;
- CoAP;
- IMAP (e-mail.);
- WiFi (Station, Access Point);
- WiFi Monitor;
- Sqlite3 (SQL);
- Crypto;
- TLS.

2. Бібліотеки та драйвери для:

- управління апаратними модулями комплексу і зовнішніми шилдами;
- управління модулем GPS / ГЛОНАСС;
- управління модулем GSM;
- бібліотека графічного інтерфейсу з елементами дискретного і пропорційного управління.

3. Інші бібліотеки і драйвери:

При необхідності легко встановлюються/використовуються необхідні бібліотеки і драйвери.

СИСТЕМА АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ PROTEUS

Proteus Design Suite – пакет програм для автоматизованого проектування (САПР) електронних схем. Розробка компанії Labcenter Electronics (Великобританія).

Пакет являє собою систему схемотехнічного моделювання, що базується на основі моделей електронних компонентів, прийнятих у **PSpice** (Personal Simulation Program with Integrated Circuit Emphasis – програма симуляції аналогової і цифрової логіки).

Відмінною рисою пакета PROTEUS VSM є можливість моделювання роботи програмованих пристроїв: мікроконтролерів, мікропроцесорів, DSP NF та ін. Бібліотека компонентів містить довідкові дані. Додатково у пакет PROTEUS VSM входить система проектування друкованих плат.

Пакет **Proteus** складається з двох частин, двох підпрограм:

- **ISIS** – програма синтезу та моделювання безпосередньо електронних схем;
- **ARES** – програма розробки друкованих плат.

Разом з програмою встановлюється набір демонстраційних проектів для ознайомлення.

Також до складу восьмої версії входить середовище розробки **VSM Studio**, що дозволяє швидко написати програму для мікроконтролера, використовуваного у проекті, NF скомпілювати.

Пакет є комерційним. Безкоштовна ознайомча версія характеризується повною функціональністю, але не має можливості збереження файлів.

Примітною особливістю є те, що у ARES можна побачити 3D-модель друкованої плати, що дозволяє розробнику оцінити свій пристрій ще на стадії розробки.

Система підтримує підключення нових елементів (SPICE) і підключення різних компіляторів (PICOLO, ARM-подібні, AVR і т.д.).

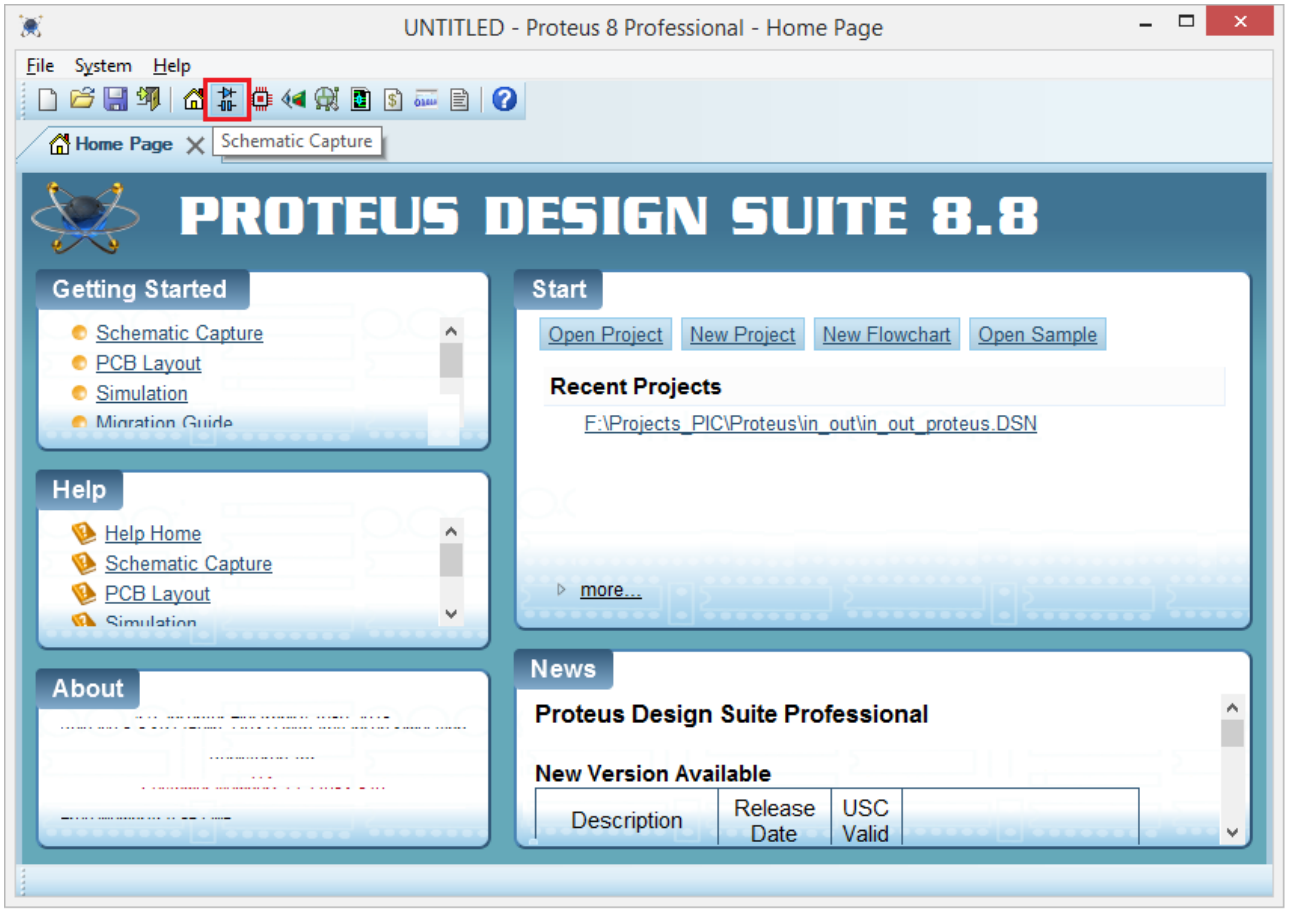

Рисунок 2.5 – Середовище моделювання «**Proteus**»

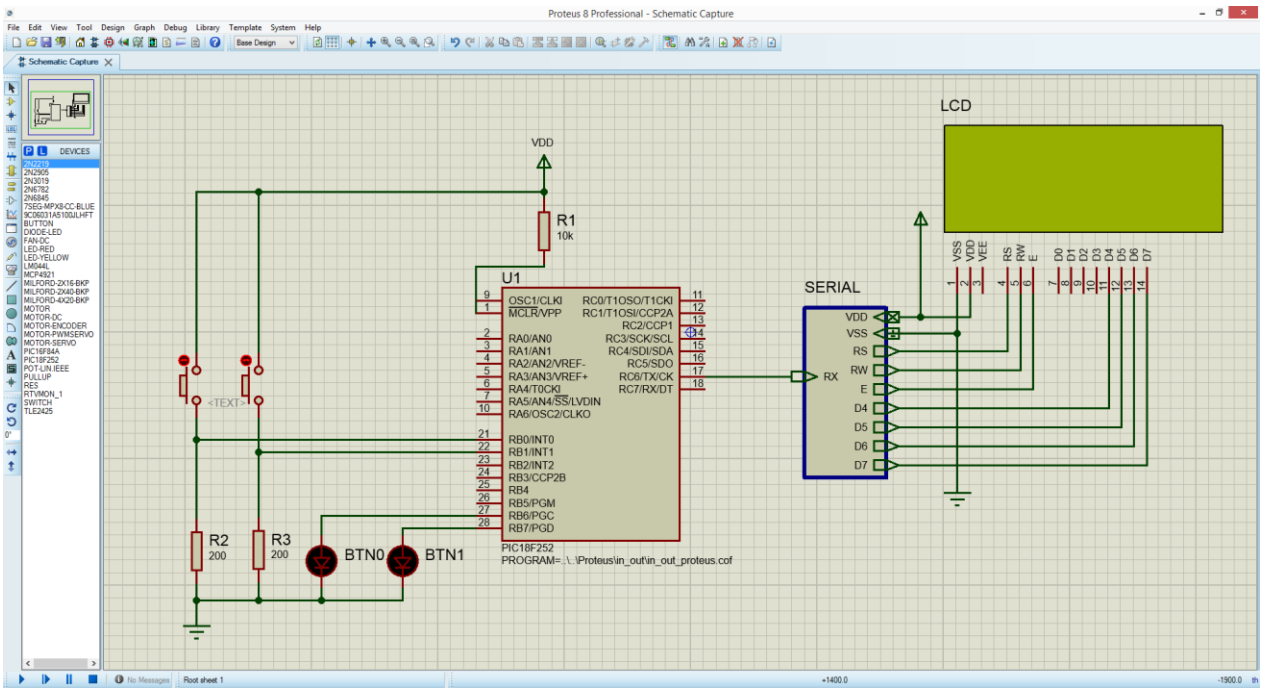

Рисунок 2.6 – Виконання лабораторної роботи
у середовищі моделювання «**Proteus**»

ОПИС РОБОТИ З ІНТЕГРОВАНИМ СЕРЕДОВИЩЕМ РОЗРОБКИ ПРОГРАМ PCWH

До складу інтегрованого середовища розробки **(IDE) PCWH** входить компілятор **CCS-PICC**, який дозволяє користувачу створювати проекти з одного або декількох файлів вихідного коду, та компілювати вихідний код у виконувані файли, призначені для завантаження у цільовий мікроконтролер.

За замовчуванням, після установки **IDE PCWH** на **Робочому столі Windows** буде розміщений ярлик **«PIC C Compiler»**, який і використовується для запуску інтегрованого середовища розробки.

PCWH можна запустити по команді меню **Пуск › Все программы › PIC-C › PIC C Compiler** або **c:\Program Files\ PICC\PCW.exe**.

Вікно інтегрованого середовища розробки **PCWH** при першому запуску показане на рис. 2.7.

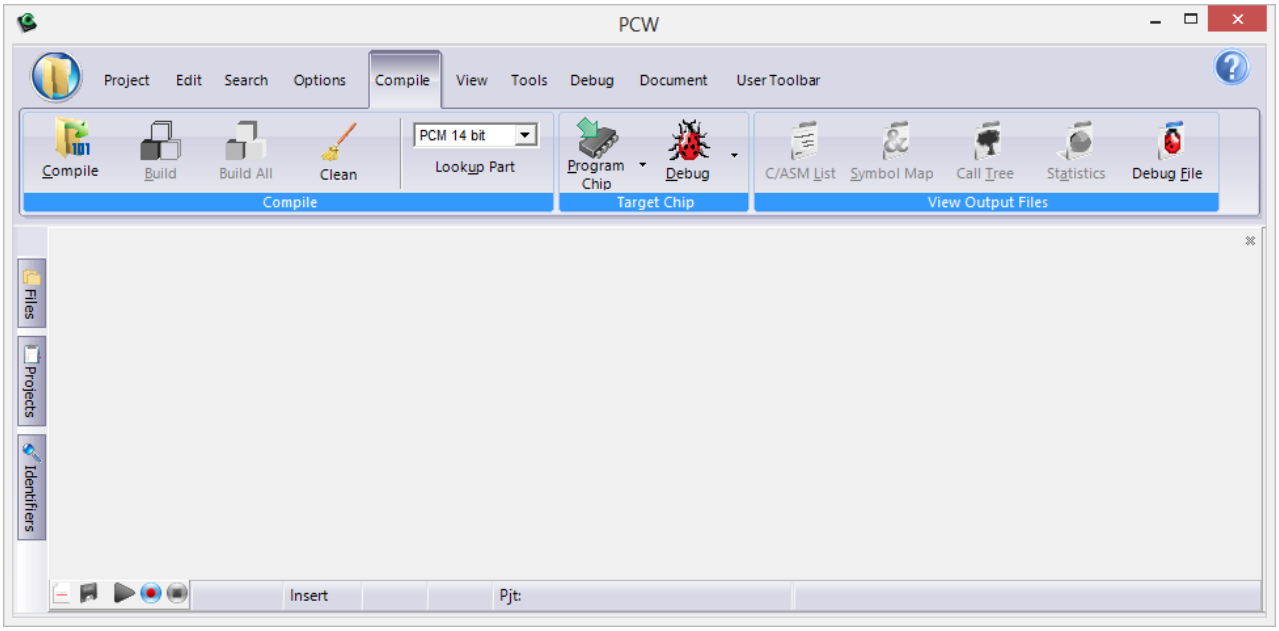

Рисунок 2.7 – Вікно інтегрованого середовища розробки **PCWH** при першому запуску

Порядок створення проекту

Проект складається з одного або більше файлів з вихідним кодом програми (головний файл проекту має розширення ***.pjt**). Новий проект можна створювати вручну за допомогою команди меню **Project › Create** (рис. 2.8),

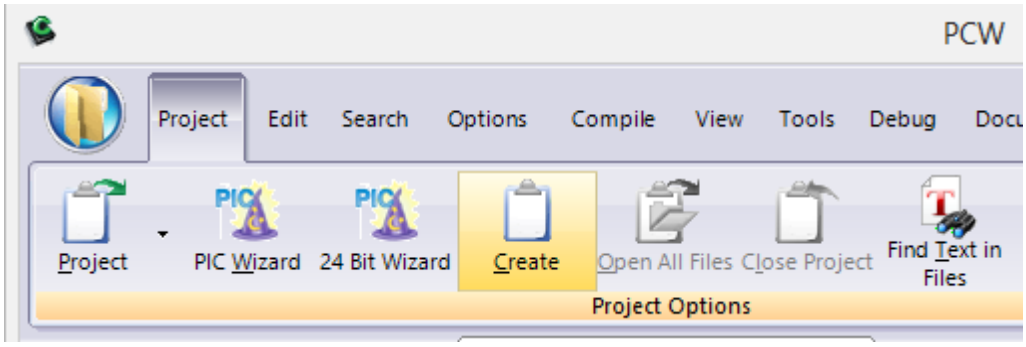

Рисунок 2.8 – Створення проекту у ручному режимі
за допомогою команди меню **Project / Create**

або згенерувати його автоматично за допомогою майстра **PIC Wizard**, який викликається по команді меню **Project › PIC Wizard або Project › 24 Bit Wizard** (рис. 2.9).

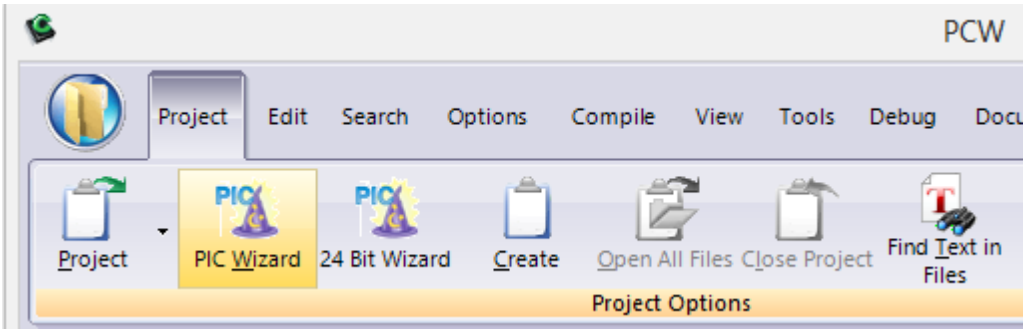

Рисунок 2.9 – Створення проекту автоматично
за допомогою майстра **PIC Wizard**

Обидва методи створення нового проекту запитують у користувача ім'я головного вихідного файлу для проекту і цільовий пристрій.

Створення проекту у інтегрованому середовищі розробки PCWH за допомогою команди меню Project/Create

Якщо було вибрано створення проекту за допомогою команди меню **Project/Create**, то спочатку з'явиться стандартне діалогове вікно, яке пропонує вибрати ім'я головного вихідного файлу проекту з розширенням ***.c** (рис. 2.10).

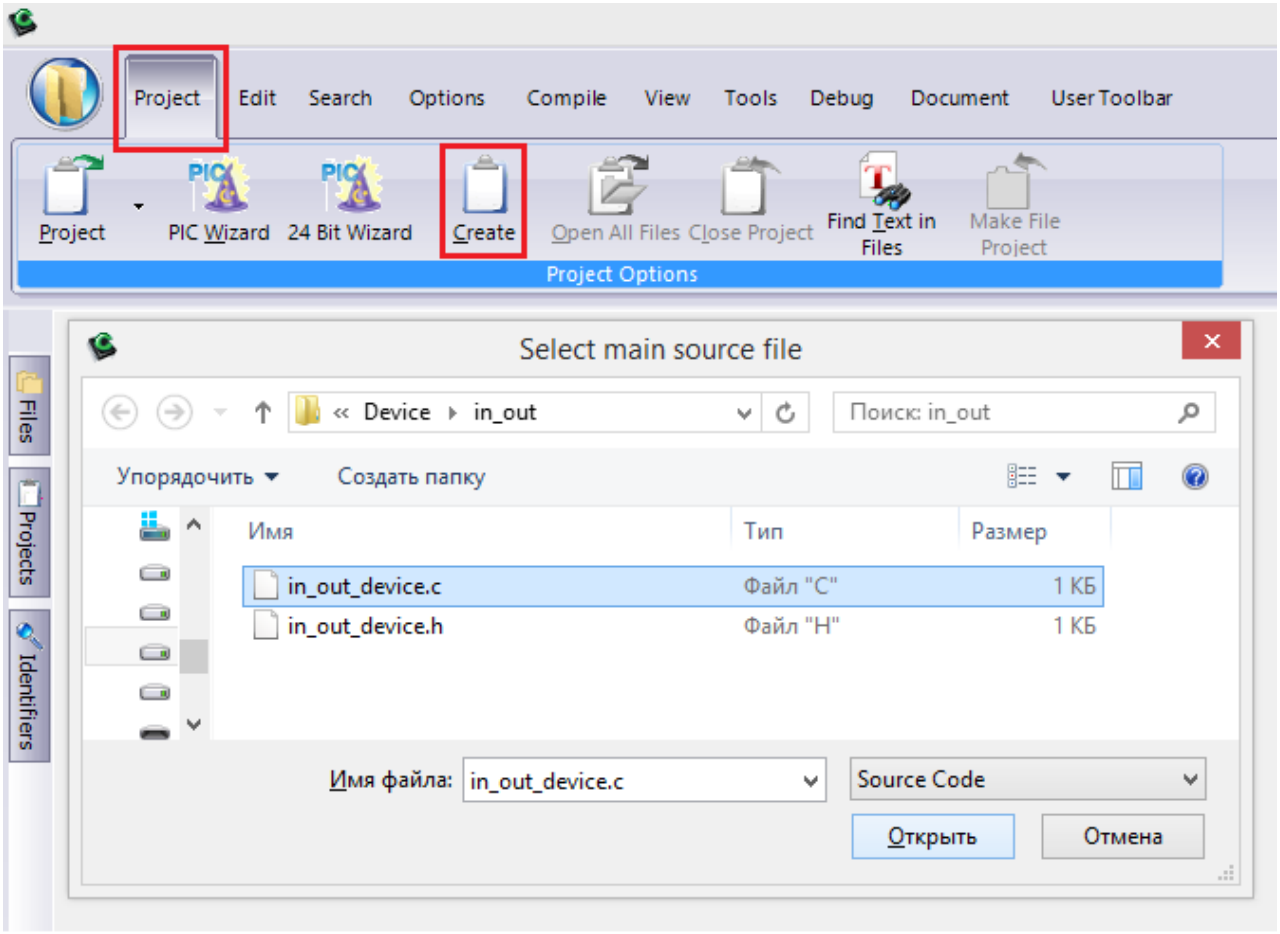

Рисунок 2.10 – Створення проекту в інтегрованому середовищі розробки **PCWH** у ручному режимі

Після вибору такого файлу відкриється діалогове вікно, яке пропонує вказати цільовий мікроконтролер. Наприклад: мікроконтролер **PIC18F252** (рис. 2.11).

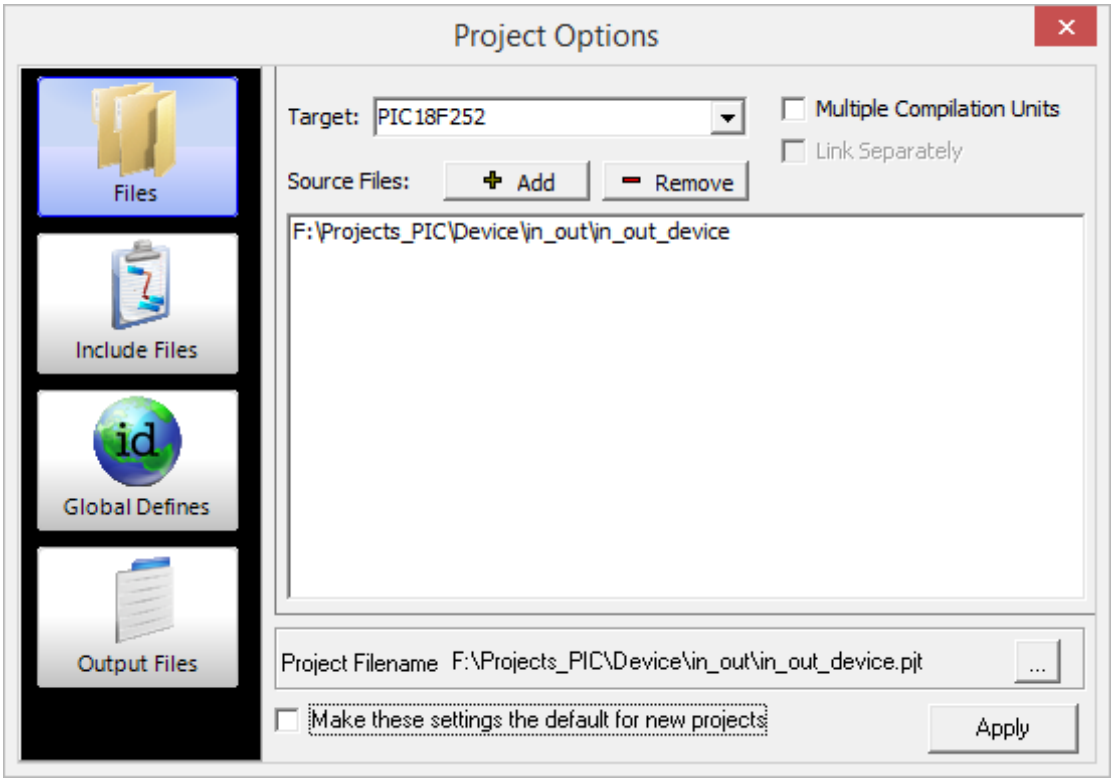

Рисунок 2.11 – Діалогове вікно **Project Options**

Це ж діалогове вікно можна відкрити у будь-який момент і після створення проекту по команді меню **Options › Project Options** (рис. 2.12).

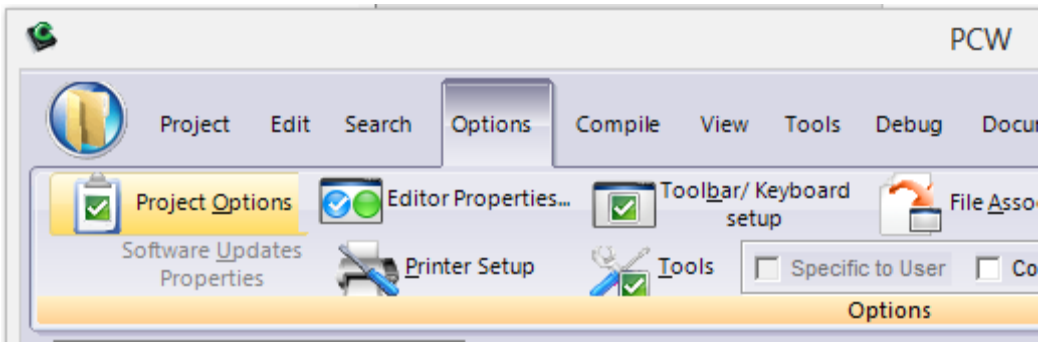

Рисунок 2.12 – Діалогове вікно **Project Options**

Якщо готового вихідного коду немає, і необхідно почати створення проекту «з нуля», то можна вибрати будь-який файл ***.c** (наприклад, з папки `\Program Files\ PICC\Examples`) (рис. 2.13).

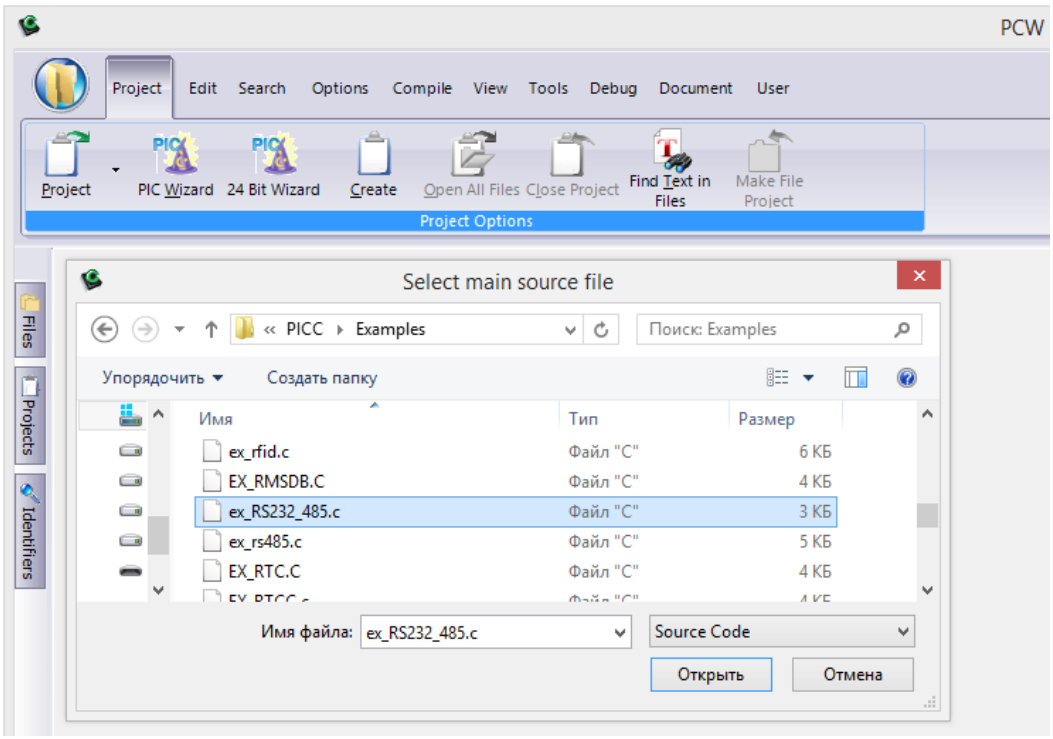

Рисунок 2.13 – Створення проекту «з нуля»

Вибрати тип мікроконтролера (наприклад, для виконання лабораторних робіт вибрати тип мікроконтролера **PIC18F252)**. Потім виділити будь-який файл ***.c** у списку діалогового вікна **Project Options** і натиснути кнопку **Remove** (Видалити) (рис. 2.14).

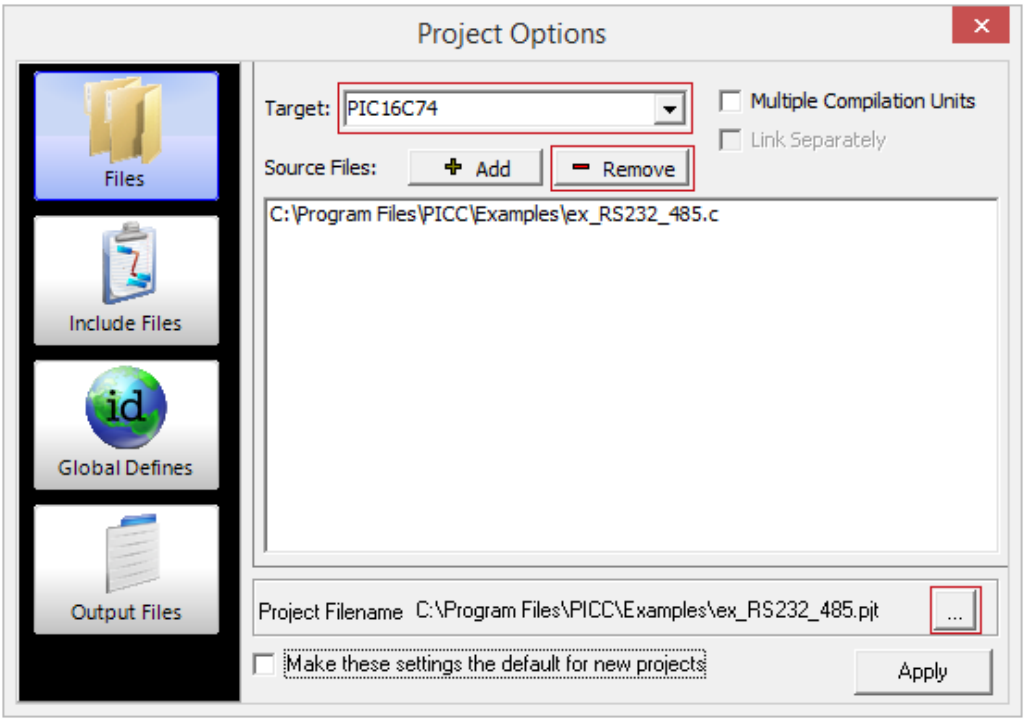

Рисунок 2.14 – Діалогове вікно **Project Options**

Надалі файл вихідного коду буде створений автоматично. У такому випадку знадобиться також змінити папку розміщення та ім'я проектного файлу. Для цього слід натиснути кнопку [...] праворуч від поля **Project FileName** і задати нове розташування та ім'я файлу ***.pjt**.

Для того щоб задати розміщення файлів з описом підтримуваних компілятором мікроконтролерів, відмінним від обраного за замовчуванням, у вікні **Project Options** слід натиснути кнопку **Include Files**. В результаті відкриється відповідний розділ параметрів проекту (рис. 2.15).

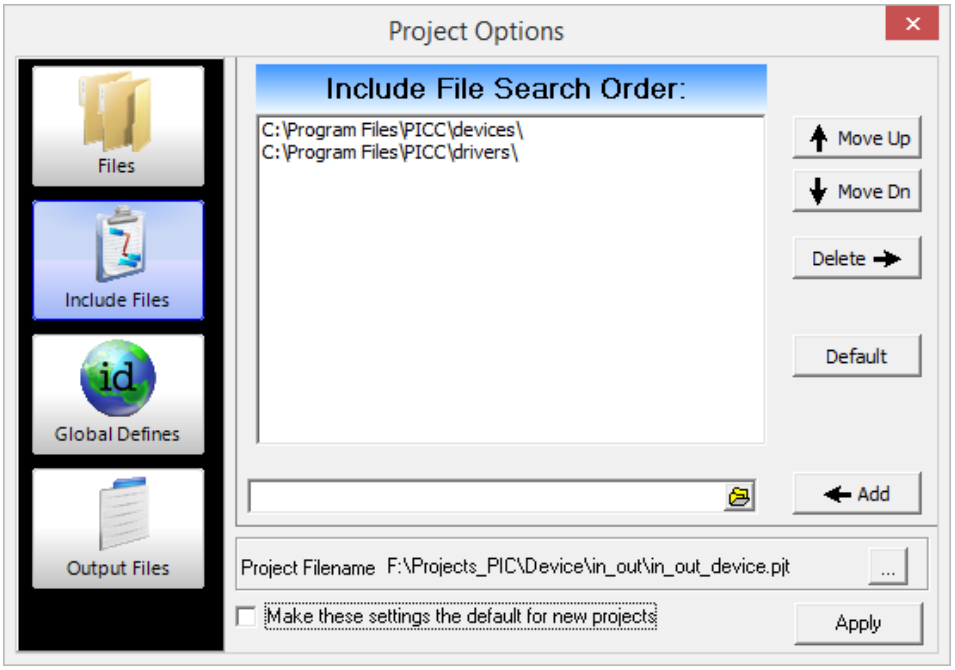

Рисунок 2.15 – Розділ **Include Files** діалогового вікна **Project Options**

Для того щоб додати новий шлях у список, слід ввести його у розташоване внизу поле (або знайти за допомогою діалогового вікна пошуку, натиснувши кнопку з зображенням папки) і натиснути кнопку **Add**.

Для видалення поточного елементу зі списку призначена кнопка **Delete**.

Після того як список шляхів пошуку сформований, можна натиснути кнопку **Apply** (Застосувати), щоб завершити створення проекту.

Якщо проект створюється на основі існуючого файлу ***.c**, то в одній з ним папці буде створений проектний файл з тим же ім'ям, але з розширенням ***.pjt**. В іншому випадку з'явиться запит на створення файлу **main.c**.

Після ствердної відповіді на цей запит буде створений проект з ім'ям, заданим у діалоговому вікні **Project Options**. У будь-якому випадку у **(IDE) PCWH** відкриється вихідний код головного файлу ***.c**.

Заголовні файли з розширенням ***.h** – це зовнішні файли описів, що підключаються до програмного модулю за допомогою директиви **#include**.

Створення проекту у інтегрованому середовищі розробки PCWH за допомогою майстра PIC Wizard

Для запуску майстра **PIC Wizard** слід виконати відповідну команду меню **Project**, а потім вказати розміщення і ім'я головного файлу проекту. В результаті відкриється вікно майстра, що складається з розділів з параметрами проекту (рис. 2.16). Зовнішній вигляд вікна майстра може відрізнятися в залежності від версії **IDE** і обраного типу мікроконтролера.

У розділі **General** (рис. 2.16) вибирається цільовий мікроконтролер (список **Device**, що розкривається), його робоча частота (поле **Oscillator Frequency**), а також настроюються загальні параметри, на зразок розрядів запобігання, типу джерела системної синхронізації, порогової напруги для скидання та ін.

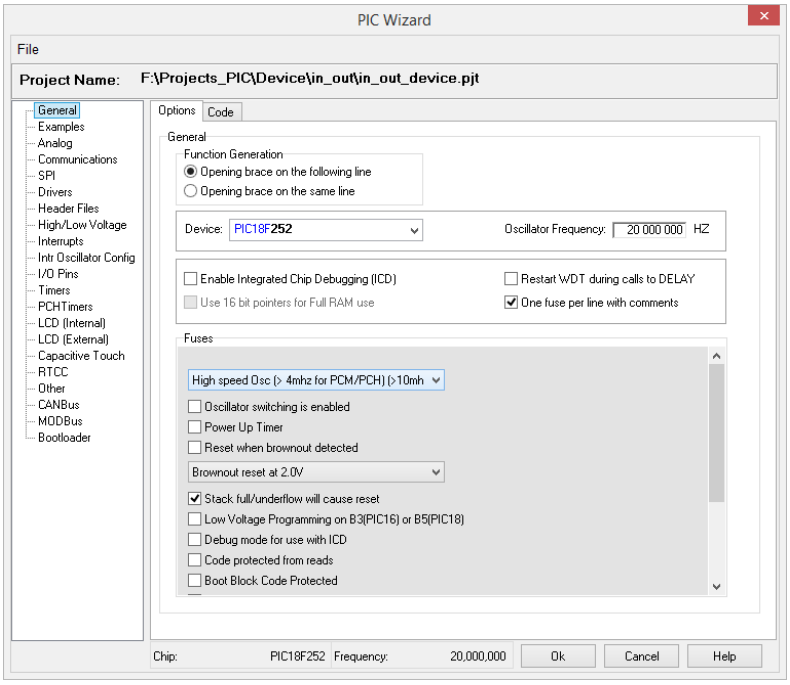

Рисунок 2.16 – Розділ **General** майстра **PIC Wizard**

Для перегляду програмного коду, який буде згенерований і доданий у вихідний файл відповідно до параметрів на поточній вкладці, слід вибрати вкладку **Code** (рис. 2.17).

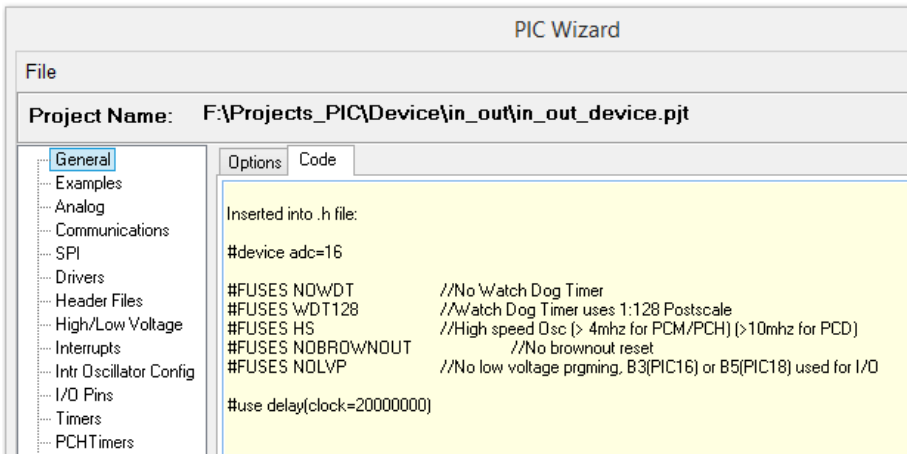

Рисунок 2.17 – Попередній перегляд коду, що генерується майстром **PIC Wizard** для поточного розділу

У розділі **Communications** (рис. 2.18) налаштовуються параметри введення/виведення для інтерфейсів **RS-232** і **I²C** (швидкість обміну даними, відповідні лінії портів введення/виведення, перевірка помилок, режим **Master/Slave** і т.д.), а також активізується/відключається апаратний порт **PSP**.

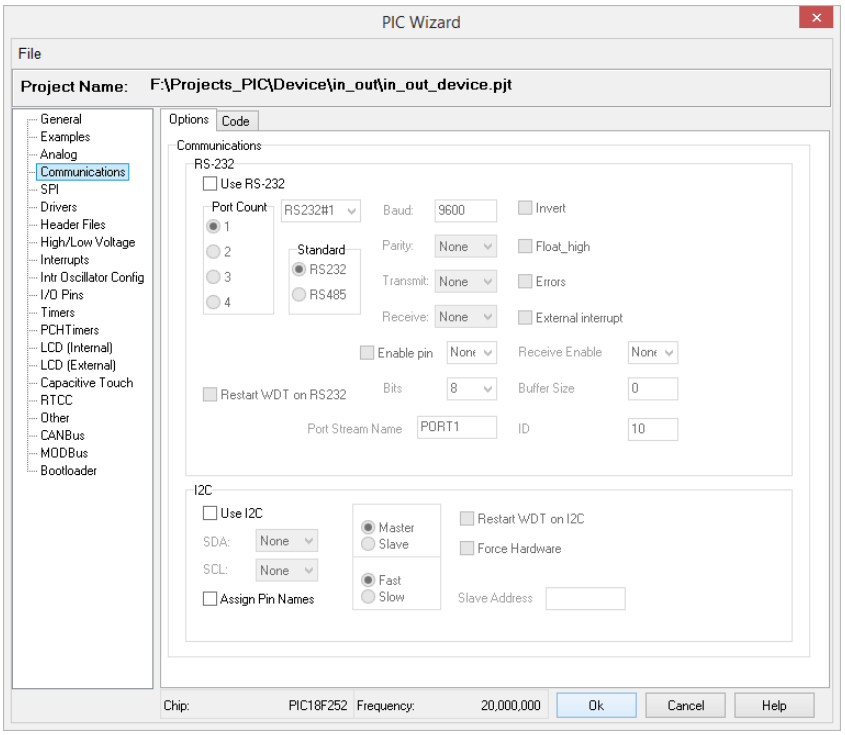

Рисунок 2.18 – Розділ **Communications** майстра **PIC Wizard**

У розділі **SPI** (рис. 2.19) користувач може активізувати апаратний інтерфейс **SPI** і налаштувати відповідні параметри обміну даними (режим **Master/Slave**, активний фронт тактового імпульсу, коефіцієнт ділення частоти системної синхронізації, використання виводу / **SS**).

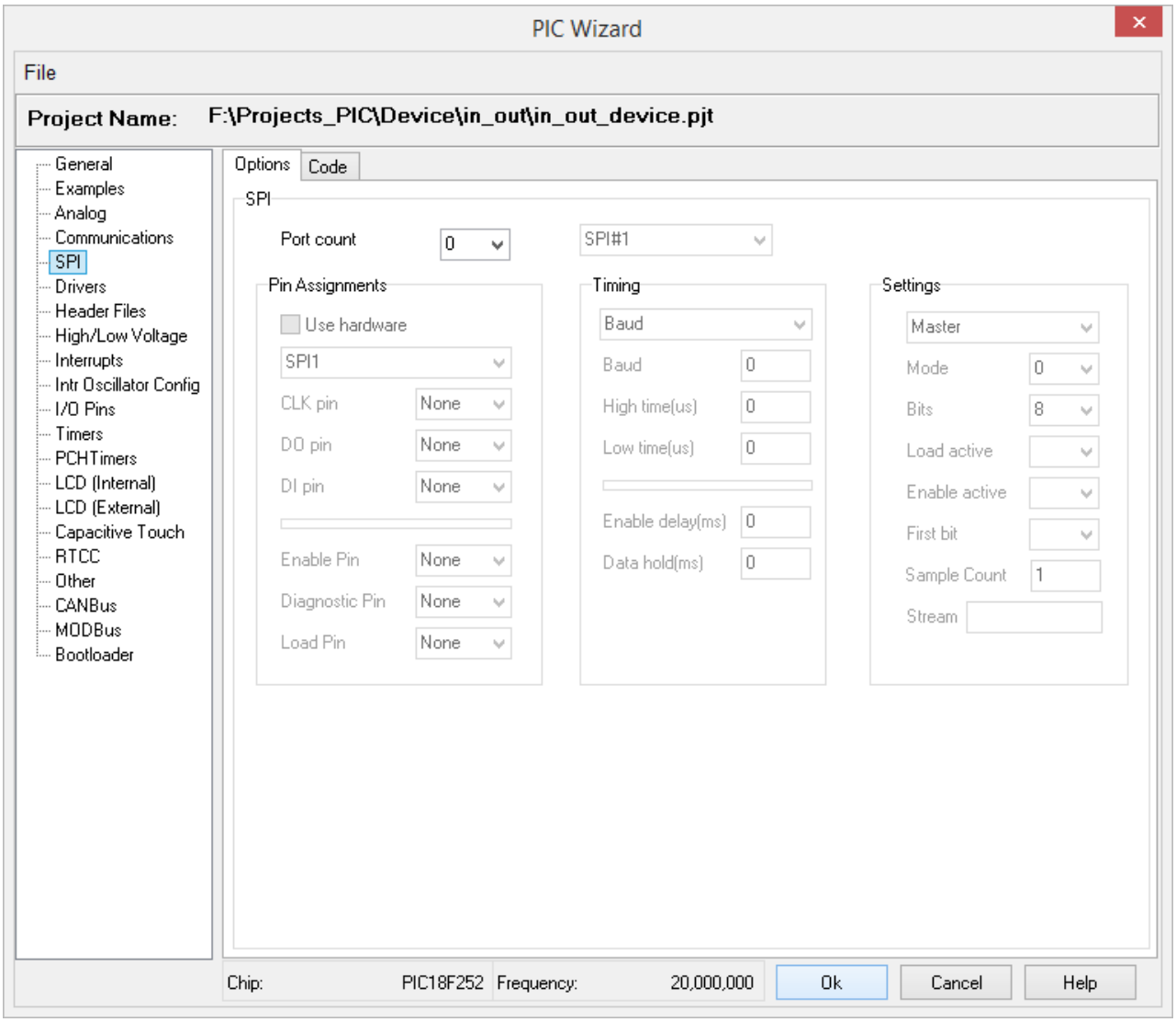

Рисунок 2.19 – Розділ **SPI** майстра **PIC Wizard**

У розділі **Timers** (рис. 2.20) налаштовуються параметри таймерів, включаючи сторожовий.

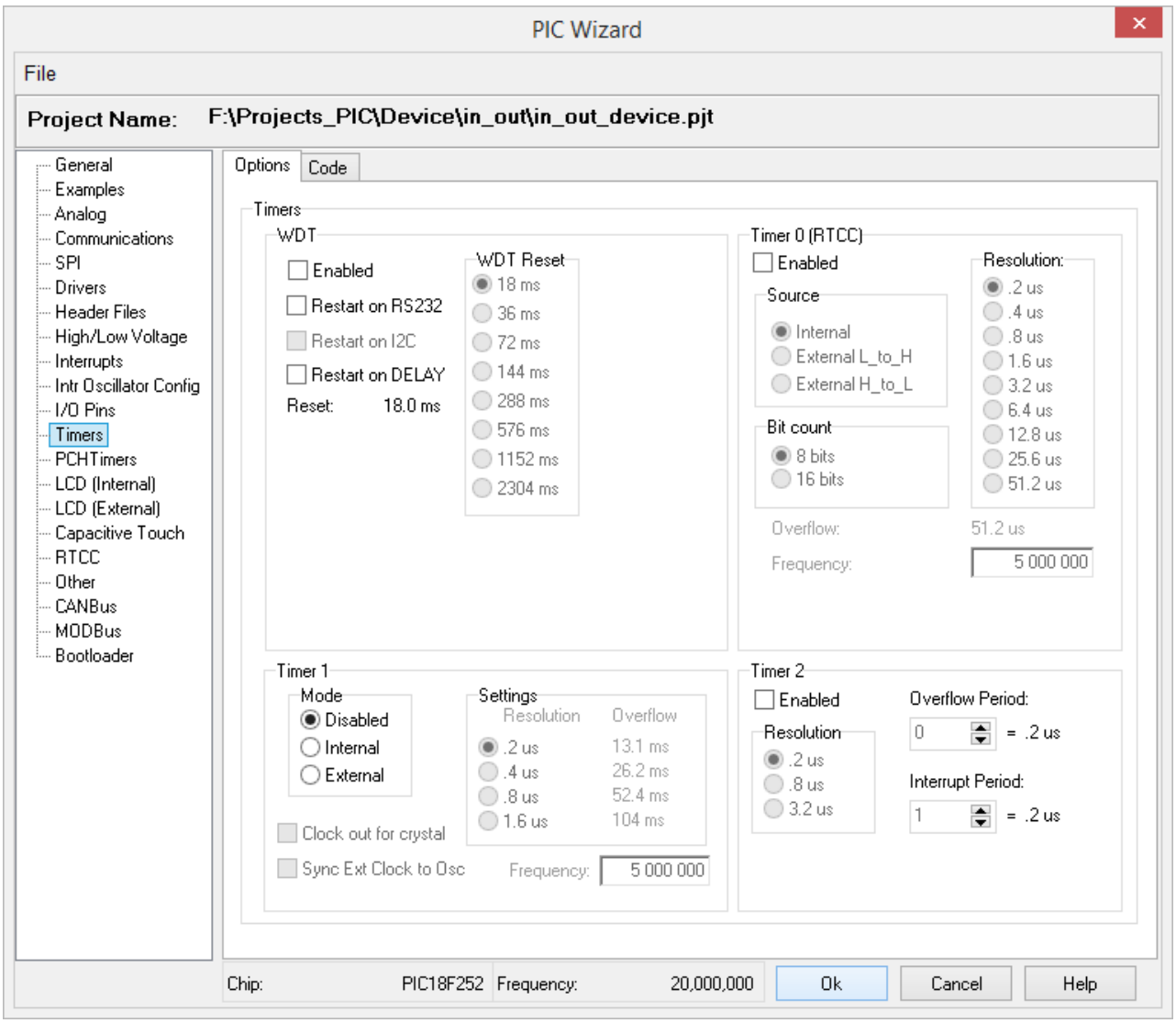

Рисунок 2.20 – Розділ **Timers** майстра **PIC Wizard**

Значення деяких параметрів:

- **WDT Reset** – період між сигналами скидання від сторожового таймера;
- **Source** – вибір джерела тактування таймера TMR0: внутрішній або зовнішній (по наростаючому або спадаючому фронту сигналу);
- **Frequency** – встановлення частоти у випадку вибору зовнішнього тактування таймера TMR0;
- **Resolution** – дозвіл таймера, що впливає на період між переповненнями лічильного регістру (для таймерів TMR0 і TMR1 обчислюється автоматично і відображається у полі **Overflow**);

- **Interrupt Period** – період між запитами на переривання від таймера TMR2.

Якщо обраний мікроконтролер надає додаткові таймери, їх параметри налаштовуються у розділі **PCH Timers** майстра **PIC Wizard**.

Розділ **Analog** (рис. 2.21) служить для налаштування вбудованого АЦП (конфігурація аналогових входів, частота і розрядність перетворення).

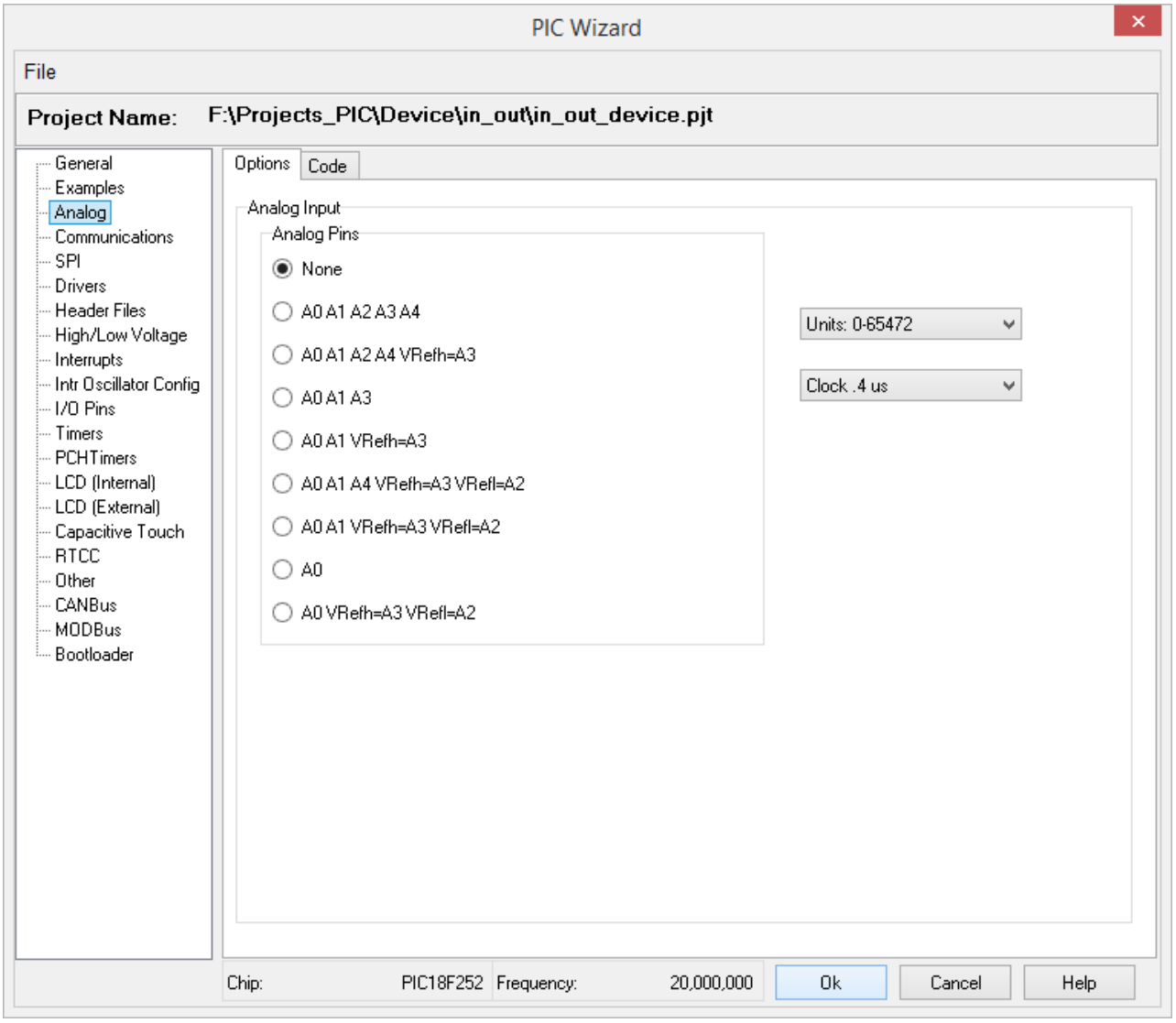

Рисунок 2.21 – Розділ **Analog** майстра **PIC Wizard**

У розділі **Other** (рис. 2.22) активізується модуль **CCP** і налаштовується режим його роботи, а також вибирається конфігурація входів компараторів напруги.

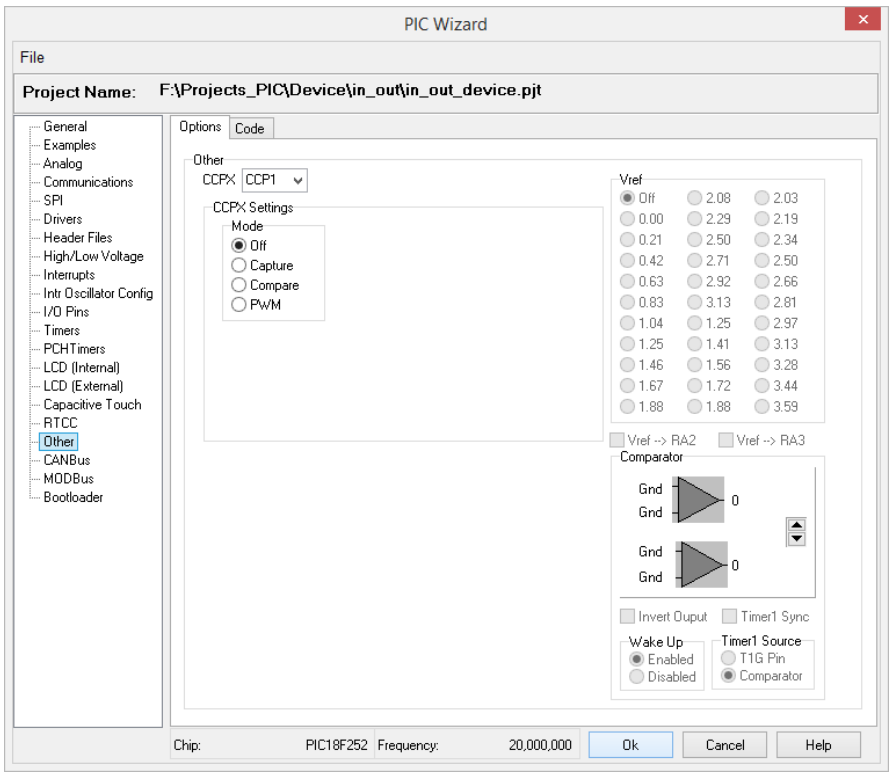

Рисунок 2.22 – Розділ **Other** майстра **PIC Wizard**

Розділ **Interrupts** (рис. 2.23) дозволяє за допомогою набору прапорців дозволити або заборонити те чи інше переривання, доступне для вибраного пристрою.

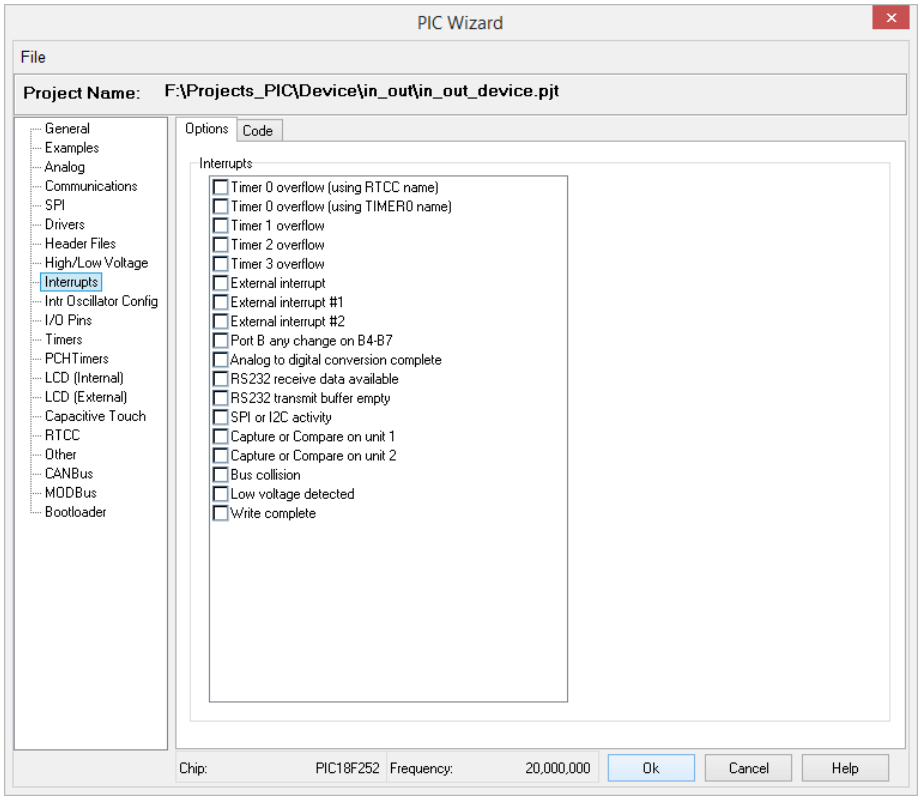

Рисунок 2.23 – Розділ **Interrupts** майстра **PIC Wizard**

Для кожного вибраного переривання у головну процедуру програми `main()` додається виклик відповідної підпрограми обробки переривання `enble_interrupts()`, а тіло самої підпрограми розміщується вище `main()` (приклад: рис. 2.24).

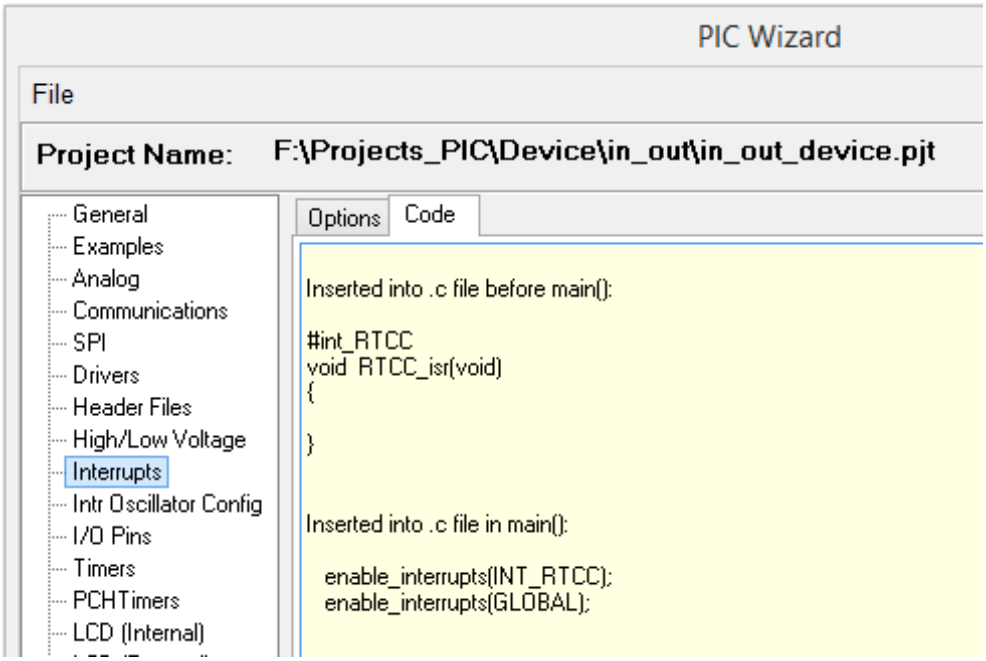

Рисунок 2.24 – Приклад програмного коду, згенерованого майстром **PIC Wizard** у розділі **Interrupts**

У розділі **Drivers** міститься список програмних драйверів, доступних для вибраного мікроконтролера.

Вибір конкретного драйверу призводить до того, що у проект включається відповідний заголовний файл, і викликається підпрограма для ініціалізації цього драйверу (за замовчуванням драйвери розміщені у папці `\Program Files\PICC\Drivers`).

Конфігурація виводів портів мікроконтролера налаштовується у розділі **I/O Pins** (рис. 2.25).

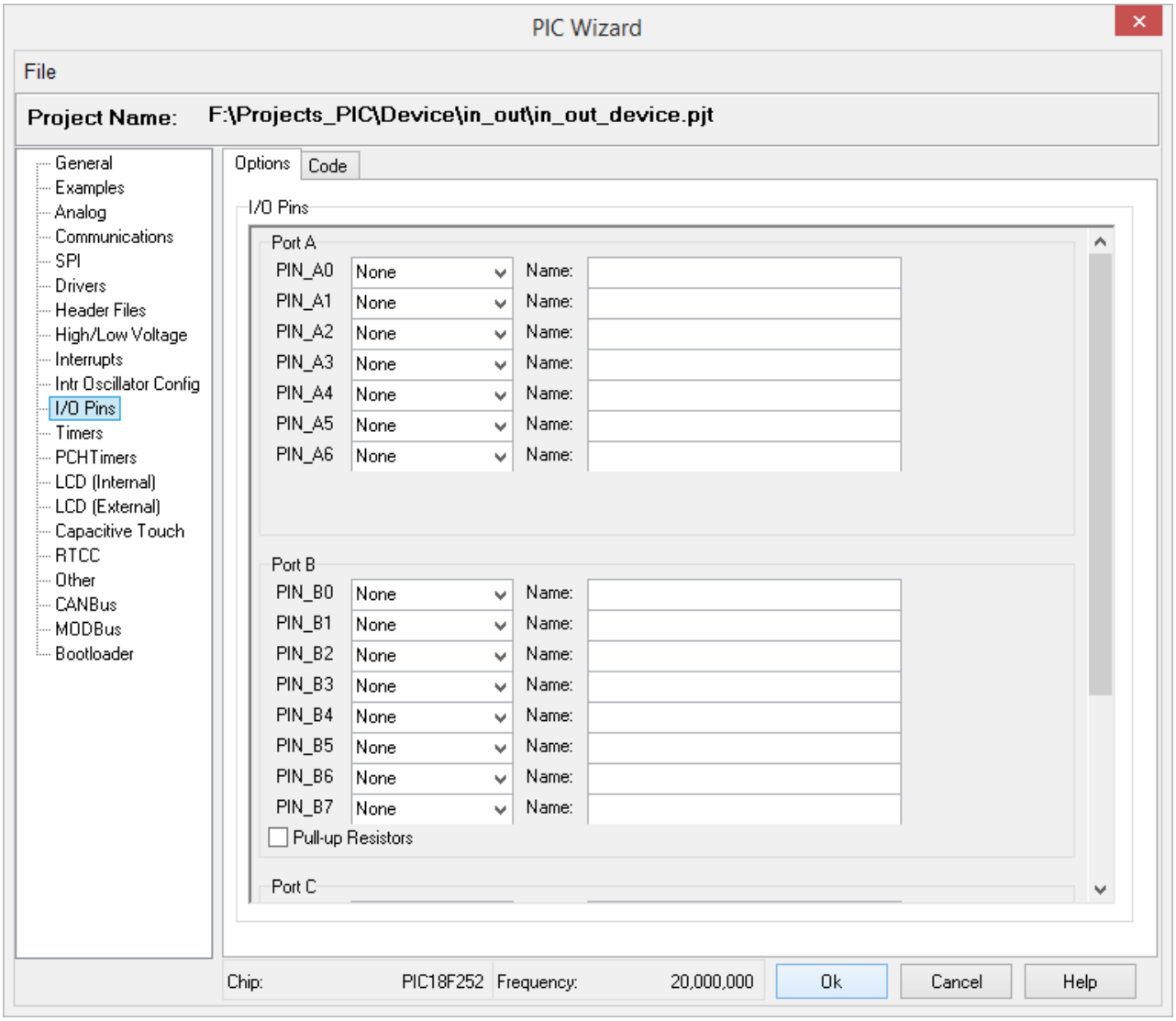

Рисунок 2.25 – Розділ **I/O Pins** майстра **PIC Wizard**

При цьому для кожного виводу можливі значення:

- **Input** (вхід);
- **Output** (вихід);
- **Input/Output** (вхід/вихід);
- **Analog** (аналоговий);
- **Not used** (не використовується).

Кожному виводу у програмі будуть відповідати ідентифікатори, зазначені через кому у стовпці **Identifiers**.

Якщо встановити прапорець **Pull-up Resistors (Port B)** у розділі **I/O Pins**, та прапорець **Timer 0 overlow (using RTCC name)** у розділі **Interrupts** то виводи порту **B** будуть зконфігуровані з підтягуючим резистором:

```
Inserted into .c file in main():
port_B_pullups(0xFF);
```

У розділі майстра **PIC Wizard Header Files** – за допомогою прапорців можна включити у проект додаткові заголовки, які використовуються, наприклад, для роботи з термінами або з числами з плаваючою комою.

Інші розділи майстра служать для налаштування різних апаратних функцій і інтерфейсів, наприклад:

- **High / Low Voltage** – виявлення підвищень і падінь рівня робочої напруги;
- **Internal Oscillator Configuration** – конфігурація внутрішнього осцилятора;
- **CAN Bus** – параметри шини CAN;
- **LCD** – параметри інтерфейсу для підключення ЖК-дисплея;
- **MOD Bus** – параметри шини MOD;
- **Bootloader** – активізація та параметри розміщення завантажувача.

Після налаштування всіх необхідних параметрів, у вікні майстра можна натиснути кнопку **ОК**, і в одній папці з самого початку заданим проектним файлом буде створений файл з розширенням ***.c** з тим же ім'ям, а також – всі необхідні файли, що включаються до заголовочного файлу ***.h**.

Компіляція проекту

Для компіляції поточного проекту можна виконати команду меню **Compile › Compile** або натиснути клавішу **<F9>** (рис. 2.26, рис 2.27).

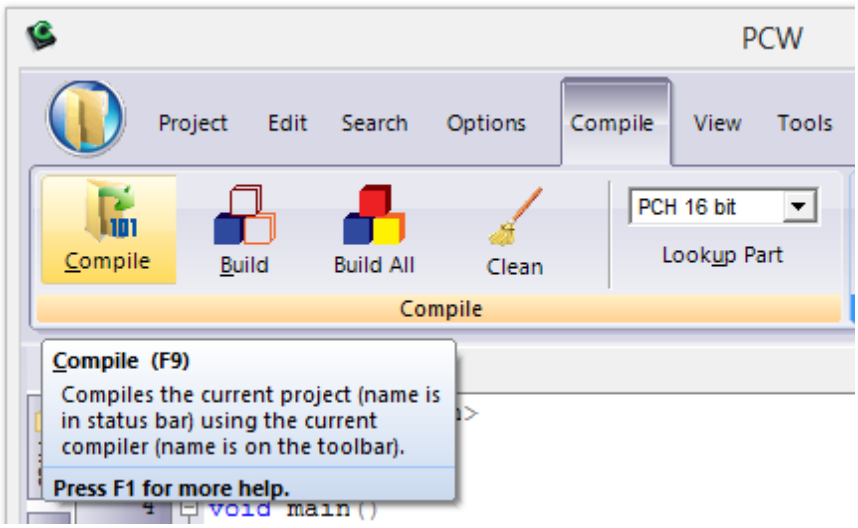

Рисунок 2.26 – Компіляція проекту

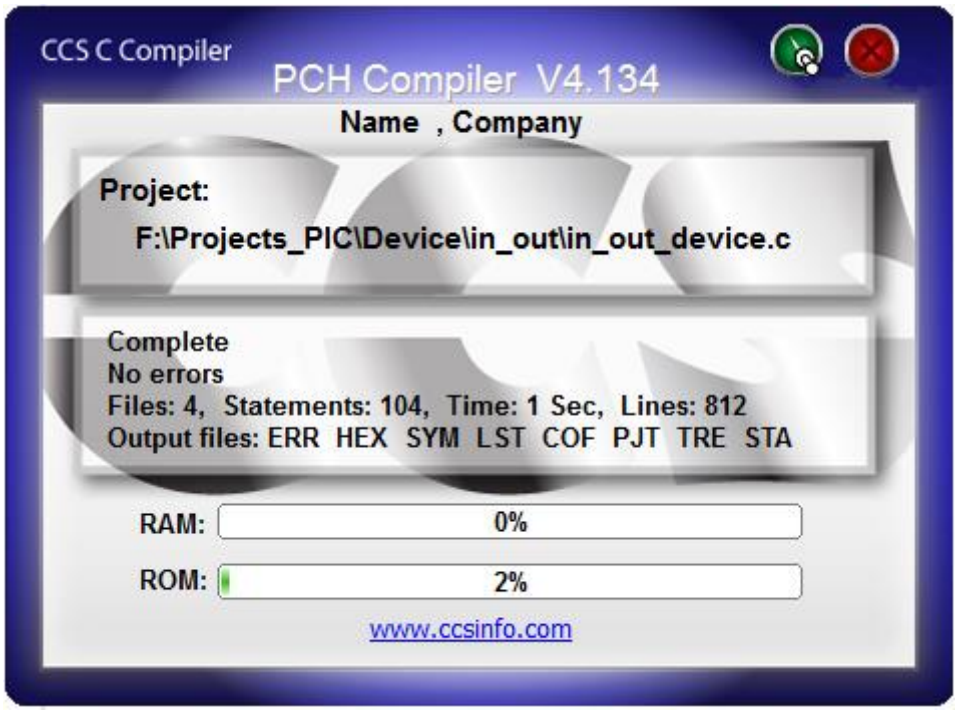

Рисунок 2.27 – Результати роботи компілятора **CCS-PICC**

У результаті компіляції у папці розміщення вихідних файлів будуть створені ще кілька файлів з тим же ім'ям, але іншими розширеннями:

- `.cof` – файл, який використовується у середовищі відлагодження;
- `.err` – файл з переліком помилок (якщо були виявлені); крім того, перша виявлена помилка буде виділена у початковому тексті програми, а її опис – відображений на червоному фоні у рядку стану;

- `.hex` – бінарний файл для завантаження у мікроконтролер;
- `.lst` – файл лістингу, в якому відображені відповідності між операторами на мові **C** і асемблерними наборами команд;
- `.sta` – файл статистики за результатами останньої компіляції;
- `.sym` – файл символів, що містить адреси змінних у пам'яті RAM;
- `.tre` – дерево викликів підпрограм.

У файлі з розширенням `*.h` містяться установки згенеровані **IDE PCWH:**

```
#include <18F252.h>
#device adc=16

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES WDT128                   //Watch Dog Timer uses 1:128 Postscale
#FUSES HS                       //High speed Osc (> 4mhz for PCM/PCH)
//(>10mhz for PCD)
#FUSES NOBROWNOUT               //No brownout reset
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or
//B5(PIC18) used for I/O

#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8,stream=PORT1)
```

Відкриття створеного проекту

Відкрити проект можна наступним чином:

1. Запустити **IDE PCWH**, натиснути кнопку **Projects › Project** (рис. 2.28):

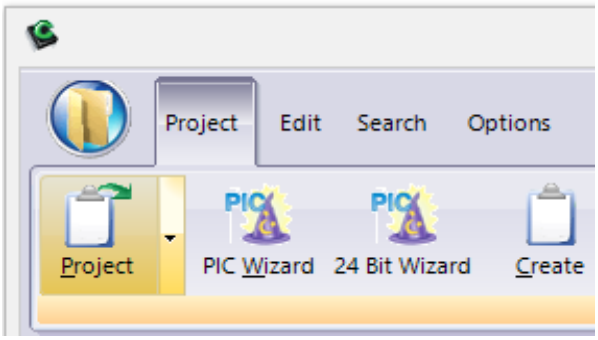

Рисунок 2.28 – Відкриття проекту у **IDE PCWH**
за допомогою майстра **PIC Wizard**

або натиснути кнопку у вигляді відкритої папки  .

2. Вибрати: **Open › Project** (рис. 2.29, рис. 2.30).

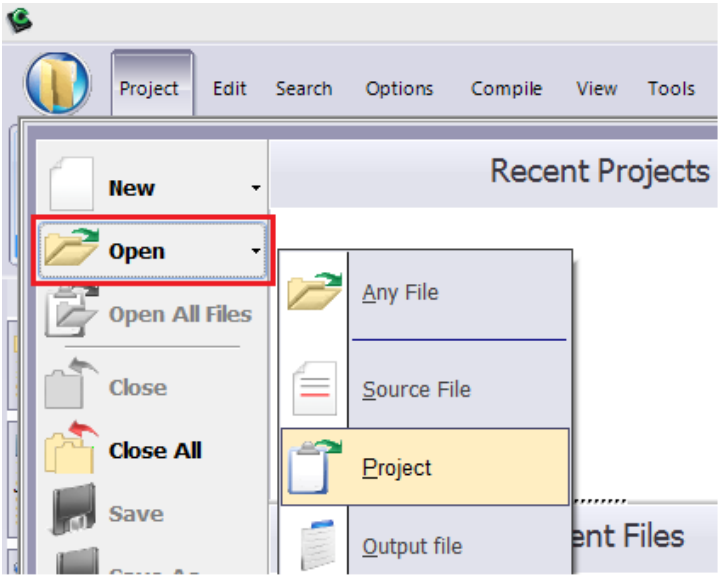

Рисунок 2.29 – Відкриття проекту у **IDE PCWH** за допомогою майстра **PIC Wizard**

3. Знайти і відкрити свій директорій і вибрати проект з розширенням ***.pjt** (рис. 2.30):

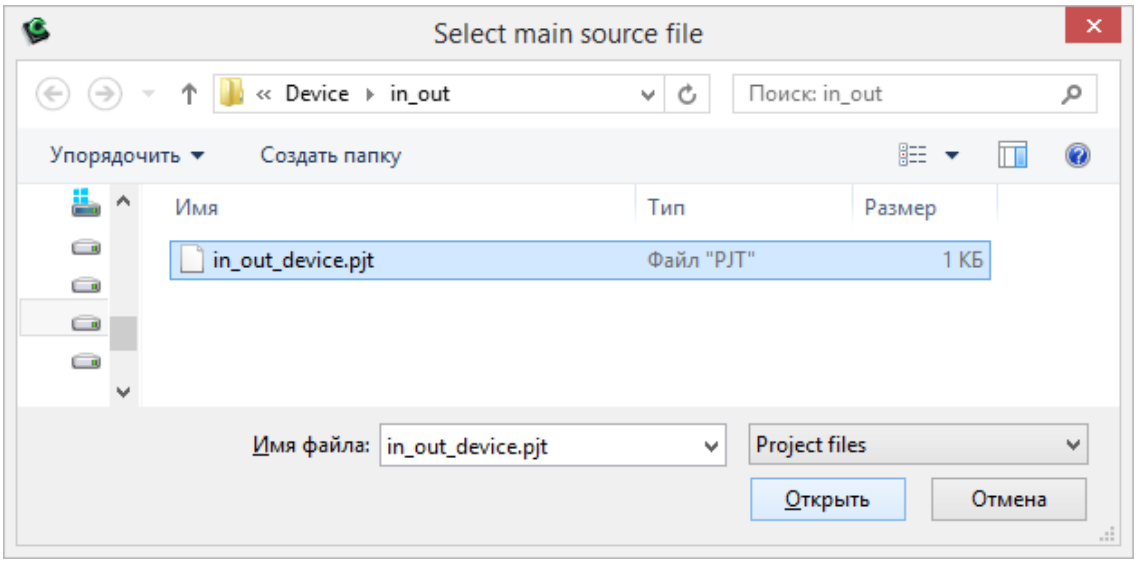

Рисунок 2.30 – Відкриття проекту у **IDE PCWH** за допомогою майстра **PIC Wizard**

Утиліти меню Tools

Меню **Toole** містить команди доступу до різних корисних утиліт:

- **Device Editor** – доступ до бази даних властивостей кожного підтримуваного мікроконтролера PIC;

- **Device Selector** – вибір цільового мікроконтролера і його властивостей;
- **File Compare** – утиліта порівняння файлів (вихідних кодів або лістингів). Якщо вибрано порівняння вихідних файлів, то виконується звичайне порядкове порівняння, якщо ж вибрано порівняння лістингів, то його можна налаштувати таким чином, щоб адреси пам'яті не враховувалися;
- **Numeric Converter** – засіб перетворення цілих і дійсних чисел в десятковому представленні в шістнадцяткове і навпаки;
- **Serial Port Monitor** – монітор послідовного порту для налагодження вбудованих систем, призначених для обміну через інтерфейс RS-232, RS-442, RS-485.

Завантаження бінарного коду у FLASH-пам'ять мікроконтролера

Двійковий файл з розширенням ***.HEX** завантажується у FLASH-пам'ять програм мікроконтролера через порт USB за допомогою програми **PICkit** (рис. 2.31).

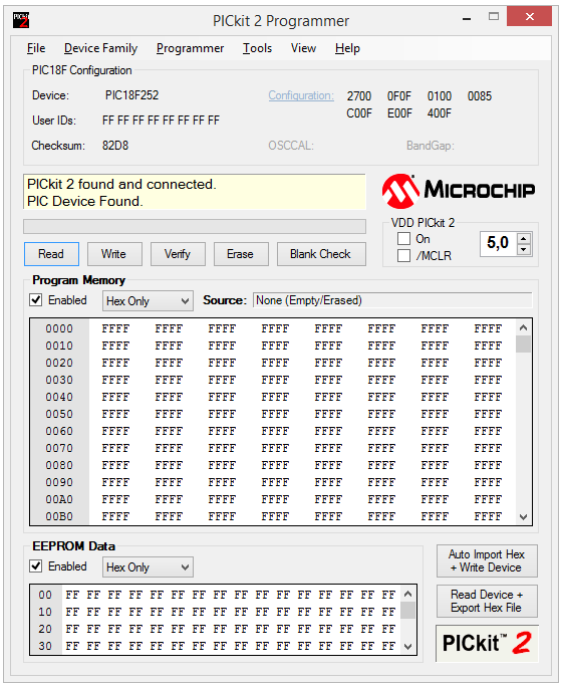

Рисунок 2.31 – Завантаження бінарного коду у FLASH-пам'ять мікроконтролера за допомогою програми **PICkit**

Для завантаження бінарного коду у FLASH-пам'ять мікроконтролера на передній панелі **АПК** натиснути кнопку **Pgm**.

Мікроконтролер готовий до приймання і завантаженню програми.

Потім у меню **File › Import HEX** ***(ctrl+I)*** вибрати ***.HEX**-файл і натиснути кнопку **Write** (рис. 2.32 – рис. 2.35):

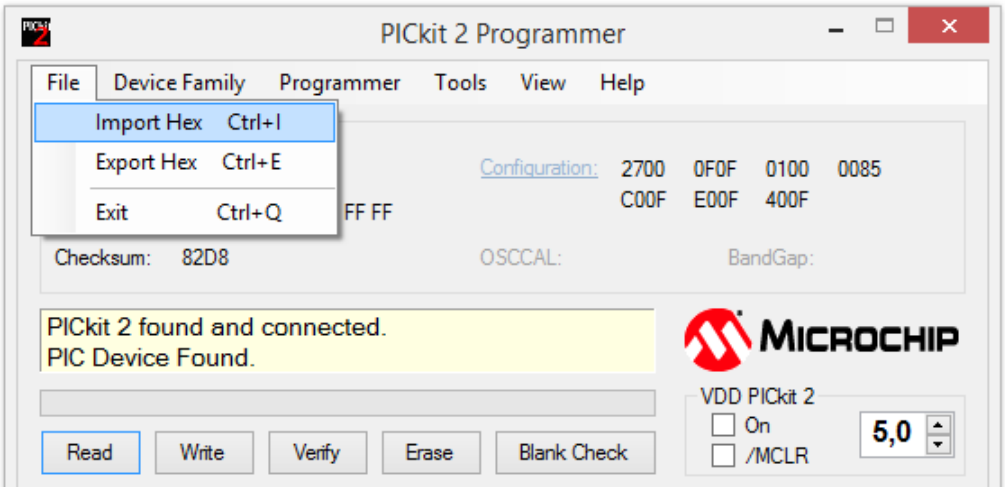

Рисунок 2.32 – Імпорт бінарного файлу «***.HEX**»
за допомогою програми **PICkit**

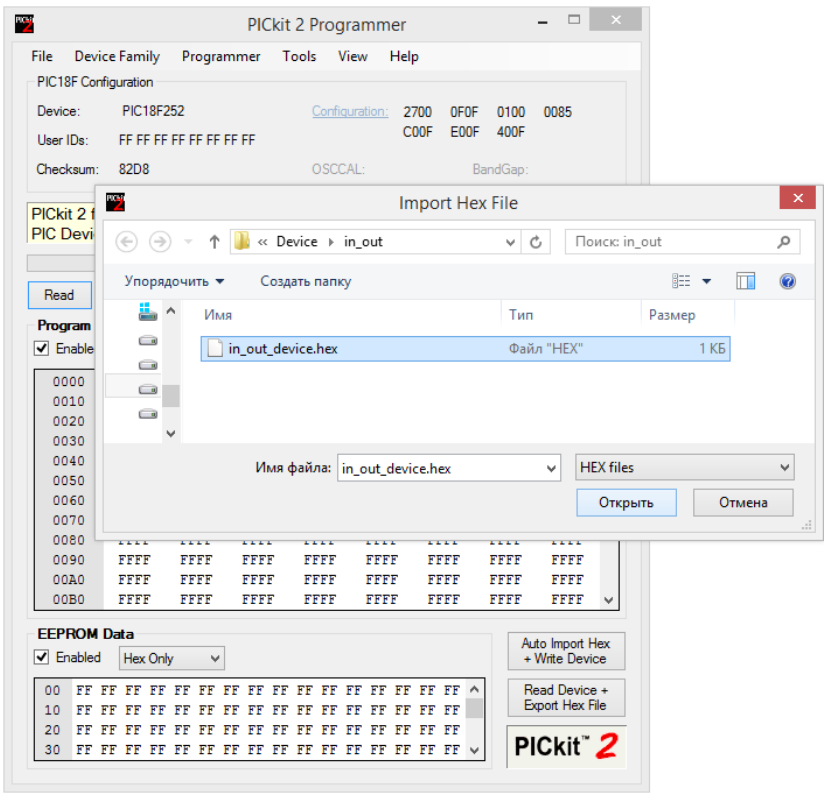

Рисунок 2.33 – Імпорт бінарного файлу «***.HEX**»
за допомогою програми **PICkit**

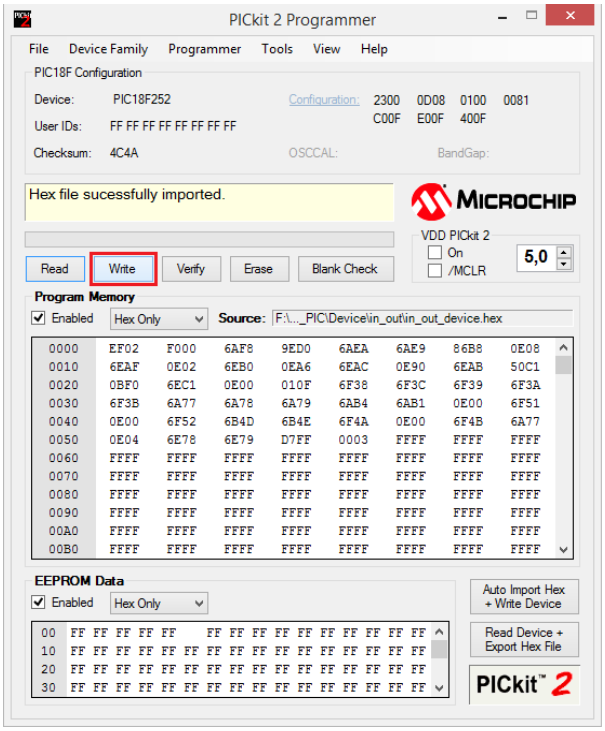

Рисунок 2.34 – Завантаження бінарного файлу «***.HEX**»
у FLASH-пам'ять мікроконтролера

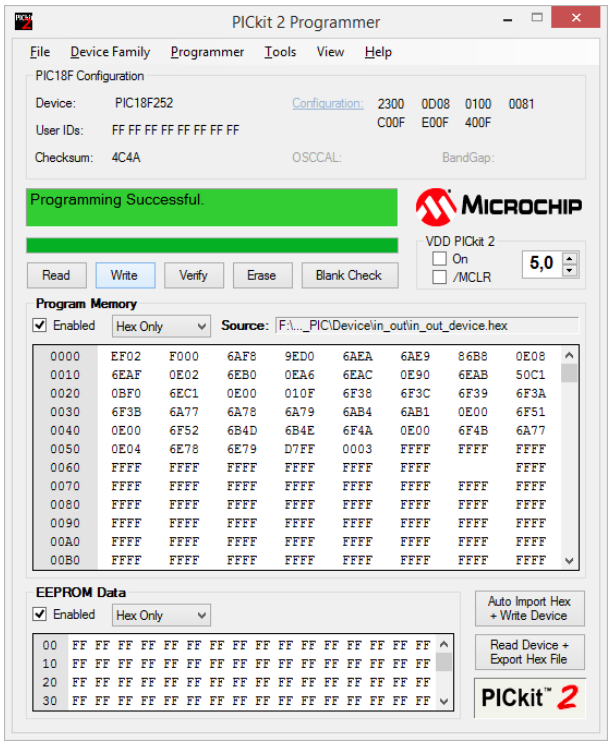

Рисунок 2.35 – Результат завантаження бінарного файлу «***.HEX**»
у FLASH-пам'ять мікроконтролера

Після завантаження програми на **АПК** натиснути кнопки **Pgm**, потім **Reset** і вона відразу починає виконуватися.

ОПИС РОБОТИ З СЕРЕДОВИЩЕМ МОДЕЛЮВАННЯ «PROTEUS»

Порядок виконання лабораторних робіт для варіанту «Proteus»

1. У папці **«Proteus_students»** вибрати папку відповідної лабораторної роботи (наприклад, **«IN_OUT»**).

2. Відкрити файл проекту з розширенням ***.DSN**.

або:

1. Відкрити середовище моделювання **«Proteus»** і натиснути на кнопку **«Schematic Capture»** або **«Open Progect»** (рис. 2.36).

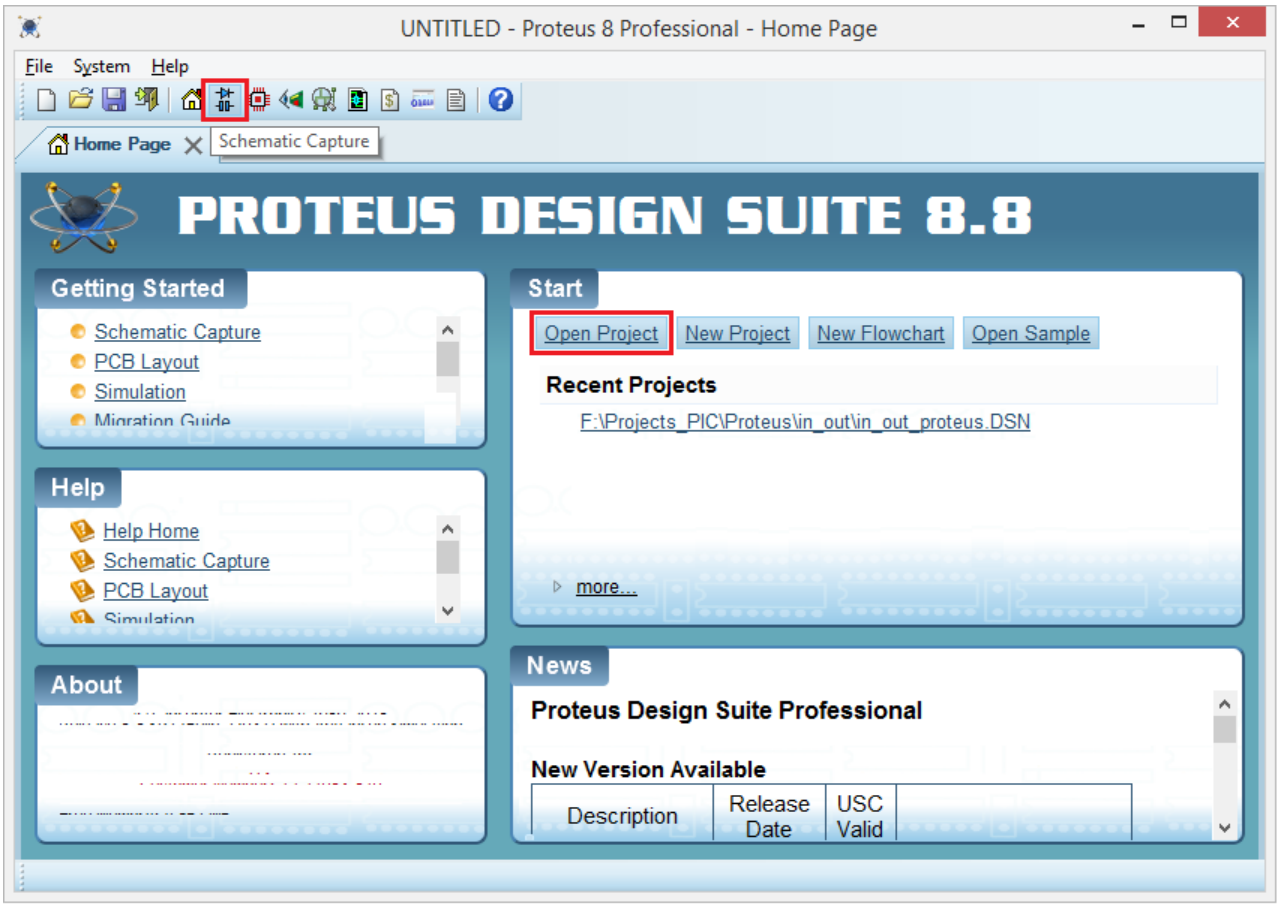

Рисунок 2.36 – Середовище моделювання **«Proteus»**

2. Відкрити файл проекту **File › Open Project (Ctrl+O)** (рис. 2.37) або натиснути на кнопку  .

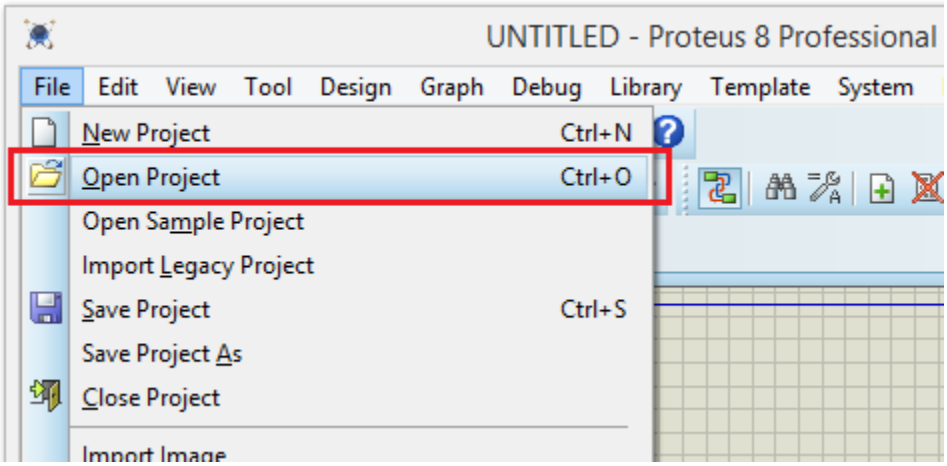

Рисунок 2.37 – Відкриття файлу проекту
у середовищі моделювання **«Proteus»**

3. У папці **«Proteus_students»** вибрати папку відповідної лабораторної роботи (наприклад, **«IN_OUT»**).

4. Вибрати **«Тип файлов: » All files**.

5. Відкрити файл проекту з розширенням ***.DSN** (рис. 2.38).

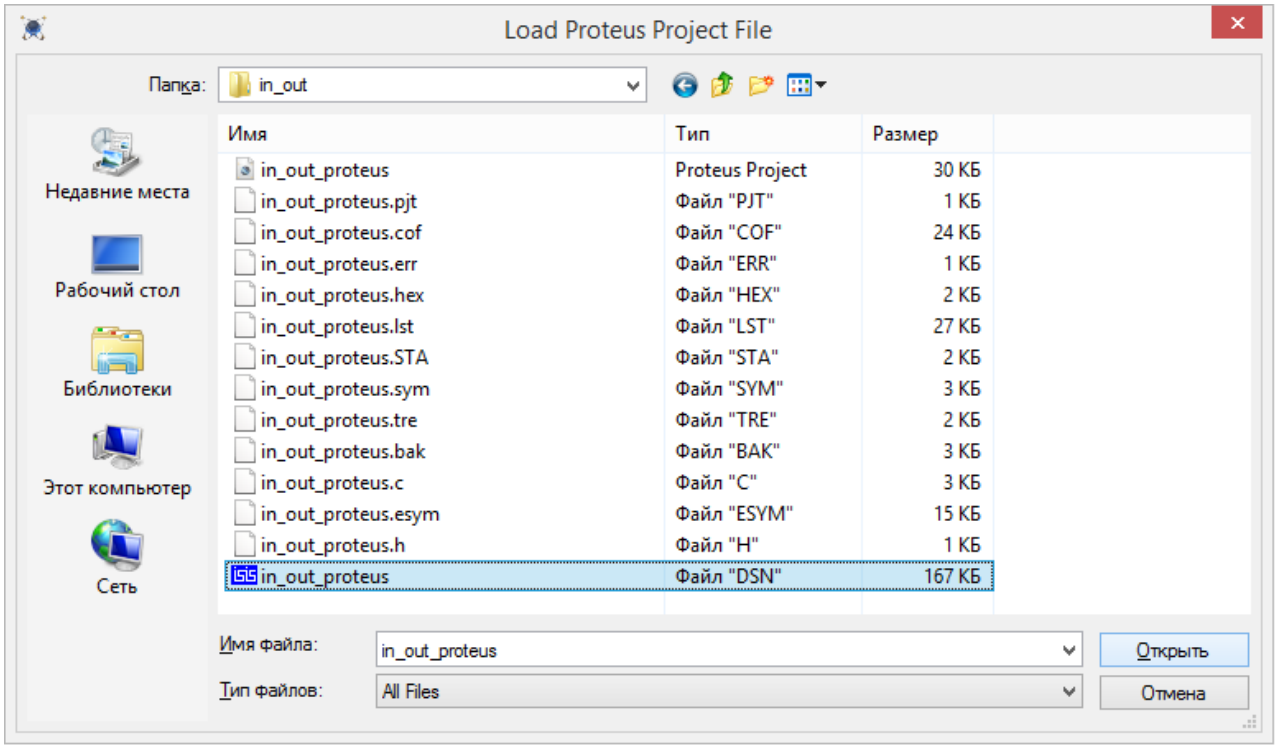

Рисунок 2.38 – Відкриття файлу проекту у
середовищі моделювання **«Proteus»**

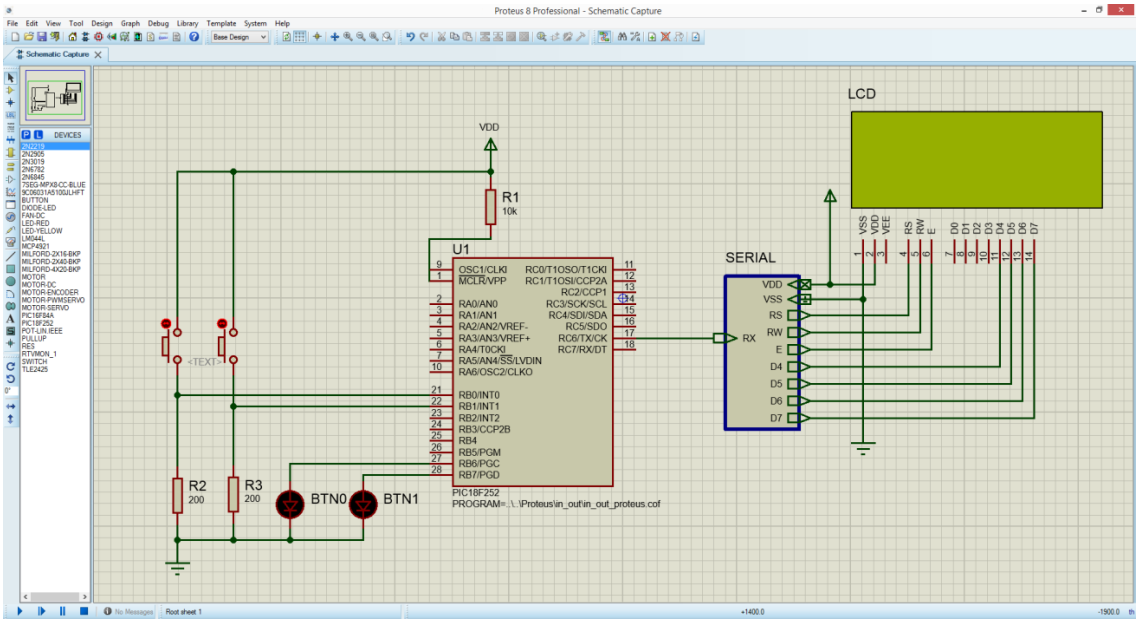

Рисунок 2.39 – Середовище моделювання **«Proteus»**

6. Змінити схему підключень відповідно до варіанту для виконання лабораторної роботи (рис. 2.39).

7. Натиснути лівою кнопкою миші двічі по мікроконтролеру на схемі.

8. У вікні **Edit Component › Program File** натиснути кнопку (рис. 2.40).

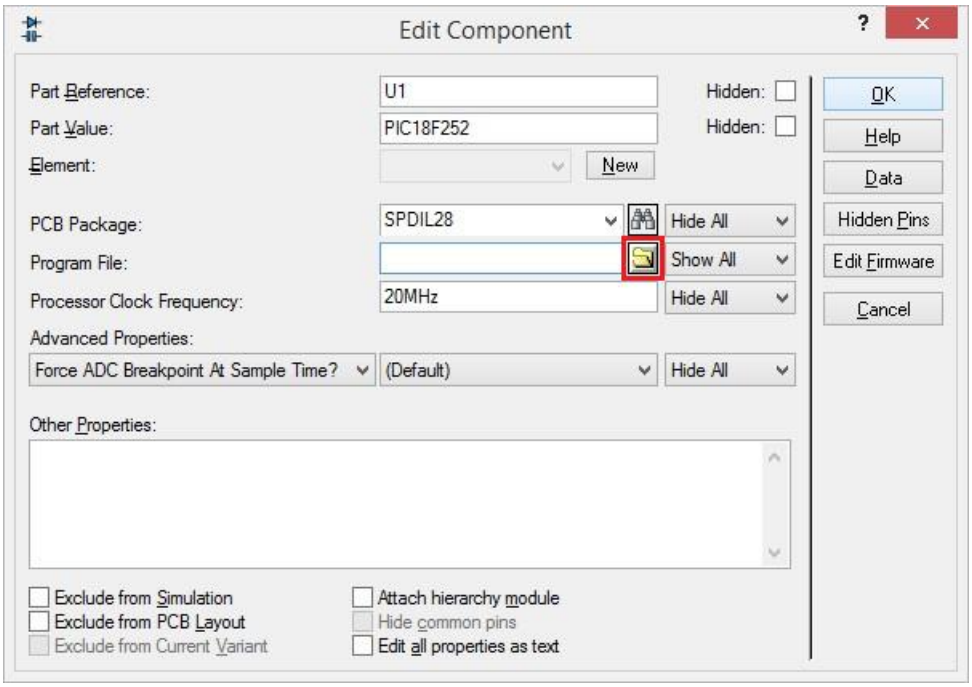

Рисунок 2.40 – Відкриття бінарного файлу з розширенням ***.HEX** або ***.COF** у середовищі моделювання **«Proteus»**

9. Завантажити заздалегідь скомпільований бінарний файл з розширенням ***.HEX** або ***.COF** у мікроконтролер (рис. 2.41).

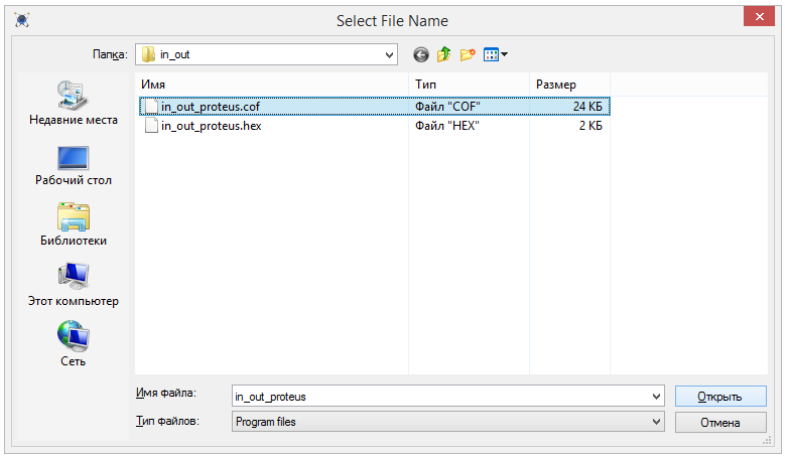

Рисунок 2.41 – Завантаження бінарного файлу з розширенням ***.HEX** або ***.COF** у середовищі моделювання **«Proteus»**

10. У вікні **Edit Component › Program File** натиснути кнопку **OK** (рис. 2.42).

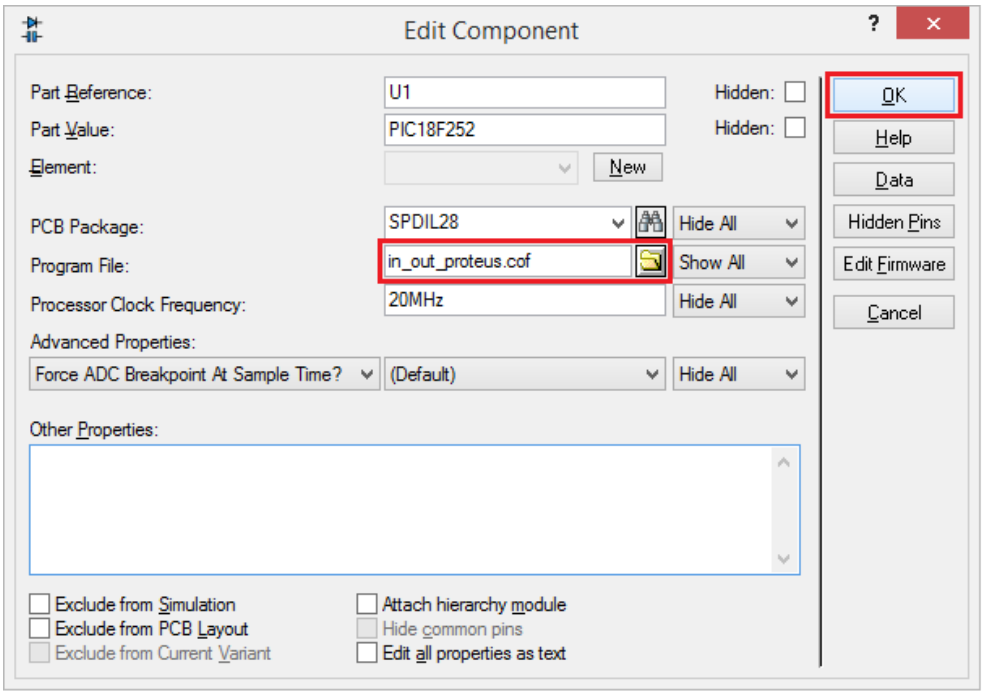

Рисунок 2.42 – Завантаження бінарного файлу з розширенням ***.HEX** або ***.COF** у середовищі моделювання **«Proteus»**

11. У середовищі моделювання **«Proteus»** виконати програму натиснувши кнопку **(Run the simulation).**

РОЗДІЛ 3. ЛАБОРАТОРНІ РОБОТИ

Для мікроконтролерів PIC розробка програмного забезпечення здійснюється мовою програмування **C** у інтегрованому середовищі розробки **PCWH** фірми **CCS**.

Всі лабораторні роботи можуть бути виконані і налагоджені за допомогою **апаратно-програмних комплексів (АПК)** (рис 2.1, рис 2.2) та у **програмі-симуляторі «Proteus»** (рис. 2.36).

Виконання лабораторних робіт

Шаблон розробки принципової схеми лабораторних робіт для варіанту **АПК** представлений на рис. 3.1:

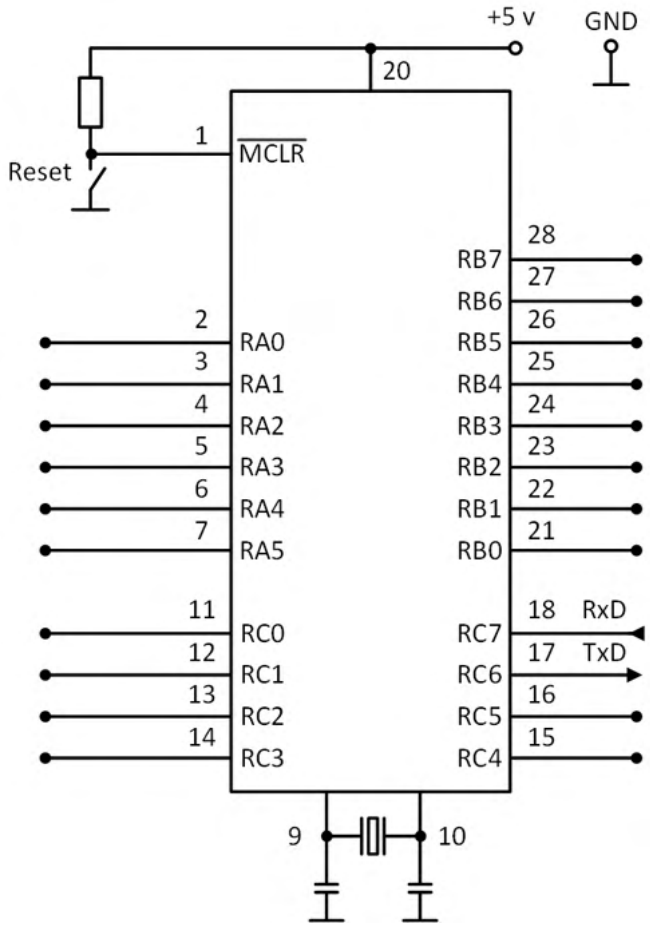

Рисунок 3.1 – Шаблон розробки принципової схеми лабораторних робіт для варіанту **АПК**

Загальна схема процесу розробки програм для варіанту **АПК** виглядає у такий спосіб (рис. 3.2):

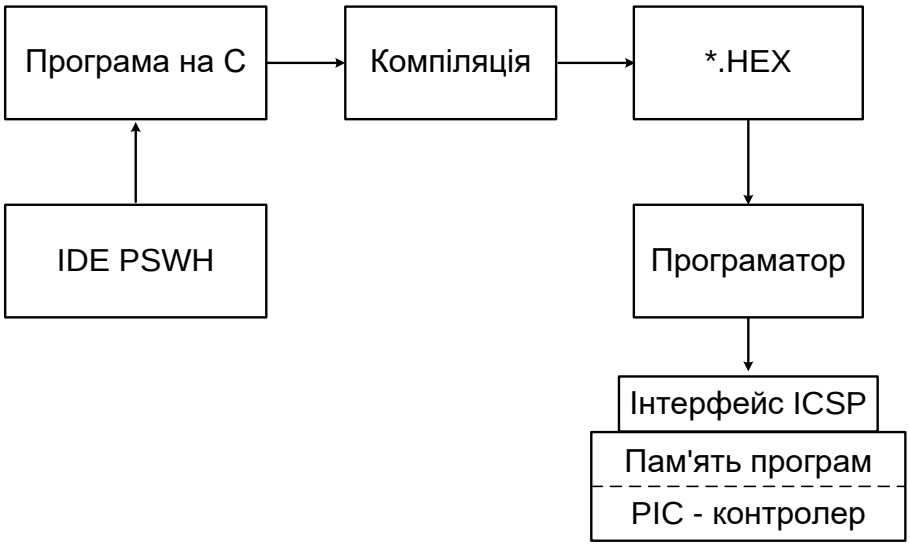

Рисунок 3.2 – Процес розробки програм для варіанту **АПК**

Загальна схема процесу розробки для **варіанту «Proteus»** виглядає наступний чином (рис. 3.3):

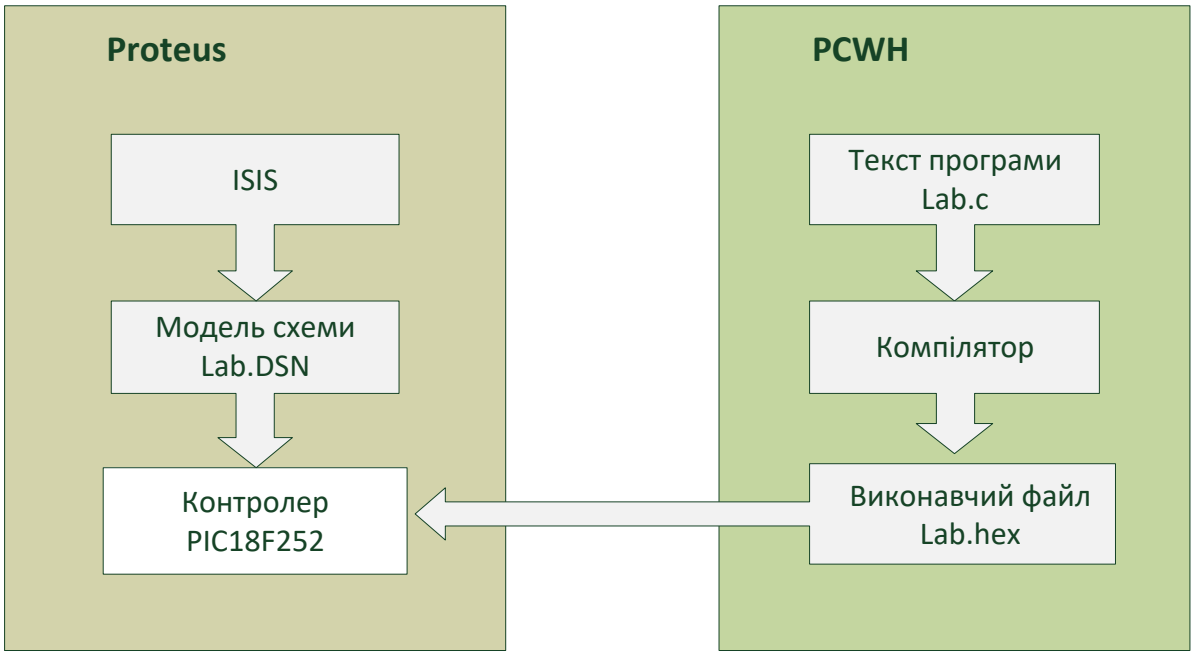

Рисунок 3.3 – Процес розробки програм для **варіанту «Proteus»**

ЛАБОРАТОРНА РОБОТА № 1 - «IN_OUT»

Тема: Введення/виведення дискретних сигналів

Ціль роботи:

Отримання навичок роботи з **IDE PCWH**, програмування мікроконтролерів, написання і налагодження програми введення/виведення дискретних сигналів.

Завдання лабораторної роботи

- для варіанту **АПК** намалювати принципову схему підключень відповідно до варіанту для виконання лабораторної роботи (таб. 3.1.1);
- для варіанту **«Proteus»** використовувати готову схему відповідно до варіанту для виконання лабораторної роботи;
- створити алгоритм програми;
- здійснити введення дискретних сигналів із кнопок і тумблерів **АПК** відповідно до варіанту для виконання лабораторної роботи;
- здійснити виведення результатів на LCD і на світлодіоди відповідно до варіанту для виконання лабораторної роботи.

Виведення на LCD повинне містити:

- номер лабораторної роботи;
- групу студента;
- прізвище та ініціали студента;
- вивід мікроконтролера, по якому у цей момент здійснюється читання сигналу.

Варіанти для виконання лабораторної роботи «IN_OUT»

Таблиця 3.1.1 – Варіанти для виконання лабораторної роботи **«IN_OUT»**

№	Введення (кнопки)								Виведення (світлодіоди)							
	A0	A1	A2	A3	B0	B1	B2	B3	B0	B1	B2	B3	B4	B5	B6	B7
1	+	+							+	+						
2		+	+							+	+					
3			+	+							+	+				
4	+			+					+			+				
5					+	+							+	+		
6						+	+							+	+	
7							+	+							+	+
8		+		+						+		+				
9					+		+						+	+		
10						+		+						+		+
11	+			+						+						+
12	+		+								+				+	
13		+		+								+		+		
14					+			+					+		+	
15						+		+							+	+

Послідовність виконання лабораторної роботи для варіанту АПК

1) для варіанту **АПК** намалювати принципову схему підключень відповідно до варіанту для виконання лабораторної роботи;
2) створити свій директорій для файлів проекту;
3) відкрити інтегроване середовище розробки **PCWH**;
4) створити проект у інтегрованому середовищі розробки **PCWH**;
5) створити алгоритм програми;
6) написати програму мовою програмування **C**, що реалізує створений алгоритм, відповідно до варіанту для виконання лабораторної роботи;
7) скомпілювати програму, одержати бінарні файли з розширеннями ***.HEX** і ***.COF** (файли з розширеннями ***.HEX** і ***.COF** створюються підчас компіляції програми);
8) завантажити бінарний файл з розширенням ***.HEX** у пам'ять програм мікроконтролера програмою **PICkit.exe**;

9) на **АПК** зкумутувати схему підключень (кнопки, тумблери і світлодіоди) відповідно до варіанту для виконання лабораторної роботи;
10) виконати програму.

Послідовність виконання лабораторної роботи для варіанту «Proteus»

1) для середовища моделювання **«Proteus»** використовувати готову схему;
2) створити свій директорій для файлів проекту;
3) відкрити інтегроване середовище розробки **PCWH**;
4) створити проект у інтегрованому середовищі розробки **PCWH**;
5) створити алгоритм програми;
6) написати програму мовою програмування **C**, що реалізує створений алгоритм, відповідно до варіанту для виконання лабораторної роботи;
7) скомпілювати програму, одержати бінарні файли з розширеннями ***.HEX** і ***.COF** (файли з розширеннями ***.HEX** і ***.COF** створюються підчас компіляції програми);
8) відкрити середовище моделювання **«Proteus»**;
9) у папці **«Proteus_students»** вибрати папку лабораторної роботи **«IN_OUT»**;
10) відкрити файл проекту з розширенням ***.DSN**;
11) у середовищі моделювання **«Proteus»** змінити схему підключень відповідно до варіанту для виконання лабораторної роботи;
12) завантажити заздалегідь скомпільований бінарний файл з розширенням ***.COF** або ***.HEX** у мікроконтролер;
13) у середовищі моделювання **«Proteus»** виконати програму.

Послідовність виконання лабораторної роботи для варіанту «Proteus» з використанням шаблону програми

1) для середовища моделювання **«Proteus»** використовувати готову схему;
2) відкрити папку **«Proteus_students»**;
3) відкрити інтегроване середовище розробки **PCWH**;
4) у папці **«Proteus_students»** вибрати папку лабораторної роботи **«IN_OUT»**;
5) відкрити файл проекту з розширенням ***.pjt**;
6) у редакторі **IDE PCWH** відкрити шаблон файлу програми з розширенням ***.c**;
7) відкрити середовище моделювання **«Proteus»**;
8) у папці **«Proteus_students»** вибрати папку лабораторної роботи **«IN_OUT»**;
9) відкрити файл проекту з розширенням ***.DSN**;
10) у середовищі моделювання **«Proteus»** змінити схему підключень відповідно до варіанту для виконання лабораторної роботи;
11) створити алгоритм програми;
12) у інтегрованому середовищі розробки **PCWH**, використовуючи шаблон програми самостійно написати програму мовою програмування **C**, що реалізує створений алгоритм, відповідно до варіанту для виконання лабораторної роботи;
13) скомпілювати програму, одержати бінарні файли з розширеннями ***.HEX** і ***.COF** (файли з розширеннями ***.HEX** і ***.COF** створюються підчас компіляції програми);
14) у середовищі моделювання **«Proteus»** виконати програму.

Примітка: *файл демонстрації виконання лабораторної роботи* ***«IN_OUT»*** *розташований на сайті* http://pksm.kntu.kr.ua/SCME.html.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Загальні аспекти архітектури PIC - контролерів

Всі мікроконтролери PIC, мають деякі загальні особливості:

- наявність пам'яті програм і пам'яті даних;
- портів введення/виведення;
- таймерів і т.д.

Організація пам'яті

Пам'ять мікроконтролерів PIC складається з трьох основних компонентів:

- регістровий файл;
- пам'ять даних;
- пам'ять програм.

Регістровий файл розділений на дві частини і містить всі регістри, доступні у мікроконтролері PIC:

- регістри спеціального призначення (SFR - Special Function Registers);
- регістри загального призначення (GPR - General Purpose Registers).

Регістри спеціального призначення служать для управління внутрішніми процесами мікроконтролера PIC. Їх набір варіюється в залежності від конкретної моделі.

Серед важливих регістрів спеціального призначення можна назвати:

- OPTION;
- INTCON;
- регістри введення/виведення;
- таймери;
- АЦП і ін.

Пам'ять даних використовується для зберігання всіх програмних змінних. Пам'ять даних енергозалежна. Це означає, що при відключенні живлення всі

дані, що містяться в ній будуть втрачені. У мікроконтролерах PIC розрядність кожної комірки пам'яті становить вісім біт. З цієї причини такі мікроконтролери PIC називаються восьмирозрядними.

Пам'ять програм служить для зберігання інформації користувача програми. У мікроконтролерах PIC пам'ять програм реалізована за технологією FLASH, при якій стирання і повторне програмування здійснюють за допомогою програматорів. Більшість мікроконтролерів PIC може також програмуватися без видалення їх із системи. Цей процес, званий внутрішньосхемним послідовним програмуванням ISP), прискорює розробку проектів і знижує витрати на розробку.

Першим словом (ширина командного слова - 12, 14 або 16 розрядів) пам'яті програм мікроконтролерів PIC є адреса вектора скидання, по якій відбувається перехід при включенні живлення або подачі сигналу низького рівня на вхід MCLR. Таким чином, перша команда, яка виконується після скидання, розміщена за адресою "0" пам'яті програм.

Порти введення/виведення

Для управління введенням/виведенням через кожен паралельний порт служить відповідна пара регістрів:

- регістр даних - PORTx;
- регістр управління напрямком передачі - TRISx.

Кількість таких пар регістрів відповідає кількості портів. З їх допомогою через будь-який порт може бути одночасно видано або прийнято від одного до восьми біт даних.

Порти у мікроконтролерах PIC - *двонаправлені*. Це означає, що будь-який вивід порту може використовуватися і як вхід, і як вихід. Це задає відповідний керуючий регістр TRISx. Запис у визначений розряд регістра TRISx лог. "1" робить відповідний вивід порту входом, а лог. "0" - виходом.

Наприклад, для того щоб розряди "0" і "1" порту B були входами, а всі інші - виходами, в регістр TRISB необхідно записати значення 00000011 (рис. 3.1.1).

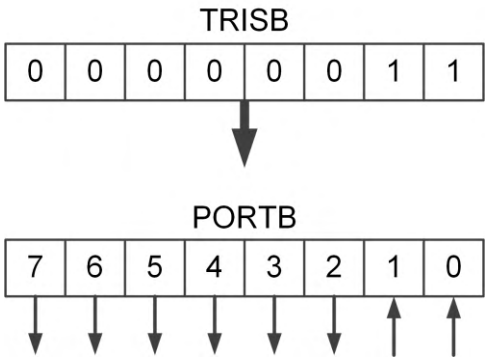

Рисунок 3.1.1 – Регістри TRISB і PORTB

Типова схема *лінії введення/виведення даних* PIC-мікроконтролерів приведена на рис. 3.1.2.

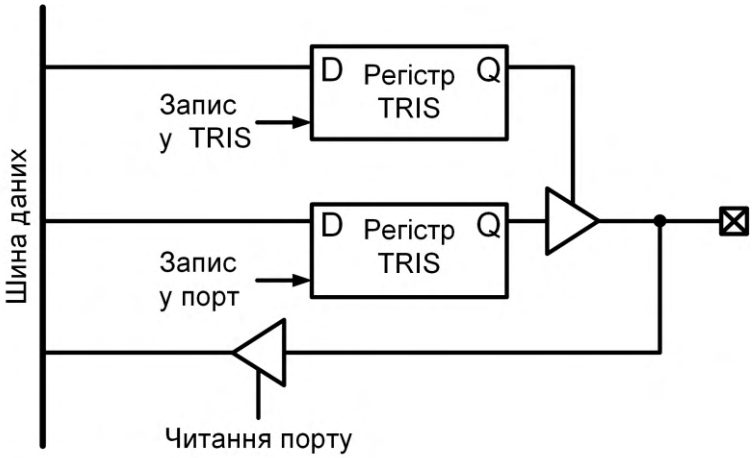

Рисунок 3.1.2 - Схема лінії введення/виведення даних PIC-мікроконтролера

Кожен порт мікроконтролера складається з певної кількості однотипних структурних компонентів, кожен з яких відповідає за визначений розряд порту введення/виведення.

Для виводів, відповідних заданому порту введення/виведення даних, використовується позначення **R%#**, де символ **%** співвіднесений з буквою порту (наприклад, порт A, порт B і т.д.), а символ **#** - з номером розряду порту.

Регістри TRIS (TRI-state buffer enable) призначені для керування напрямком передачі даних через порти.

При запису "1" в будь-який розряд регістра, відповідний вихідний буфер відключений і дані можуть надходити з входу в мікроконтролер (режим введення даних).

При запису "0" до відповідного біту регістра TRIS вихідний буфер активізується (переходить в режим виведення даних), а величина, записана в розряд регістра даних, передається на відповідний вивід мікроконтролера.

ПРИКЛАД СТВОРЕННЯ ПРОЕКТУ У ІНТЕГРОВАНОМУ СЕРЕДОВИЩІ РОЗРОБКИ PCWH

Створення проекту у інтегрованому середовищі розробки PCWH за допомогою майстра PIC Wizard

Процес створення проекту у інтегрованому середовищі розробки **PCWH** за допомогою майстра **PIC Wizard** розглянемо на прикладі лабораторної роботи «**IN_OUT»**.

Для запуску майстра **PIC Wizard** слід виконати відповідну команду меню **Project**, а потім вказати розміщення і ім'я головного файлу проекту.

В результаті відкриється вікно майстра, що складається з розділів з параметрами проекту (рис. 3.1.6).

Зовнішній вигляд вікна майстра може відрізнятися в залежності від версії компілятора **CCS-PICC** і обраного типу мікроконтролера.

1. Запустити **IDE**, натиснути кнопку **Projects** майстра **PIC Wizard** (рис. 3.1.3),

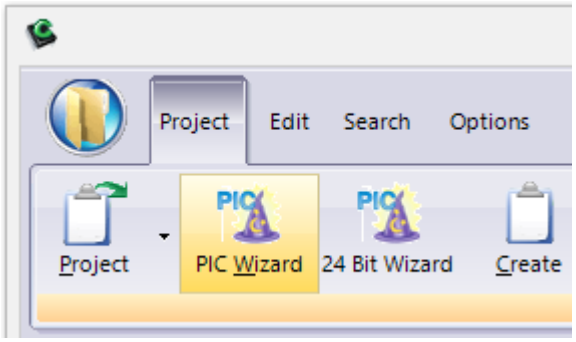

Рисунок 3.1.3 – Процес створення проекту у **IDE PCWH** за допомогою майстра **PIC Wizard**

або натиснути кнопку у вигляді відкритої папки  .

2. Вибрати: **New › Project Wizard** (рис. 3.1.4):

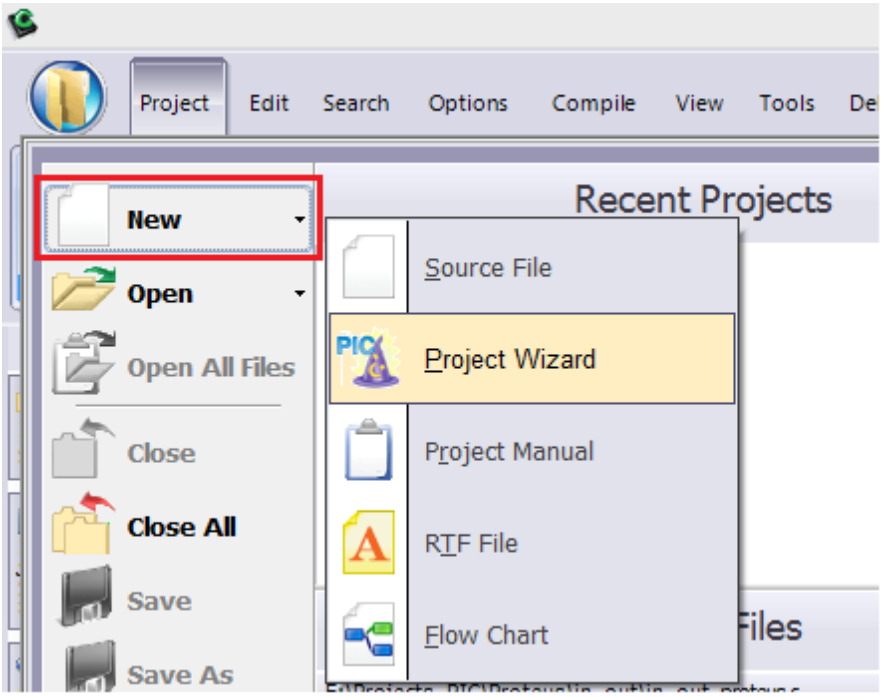

Рисунок 3.1.4 – Процес створення проекту у **IDE PCWH** за допомогою майстра **PIC Wizard**

3. Створити або вибрати свій директорій і записати у нього проект під будь-яким іменем латинським шрифтом, наприклад:

«in_out.pjt» (рис. 3.1.5):

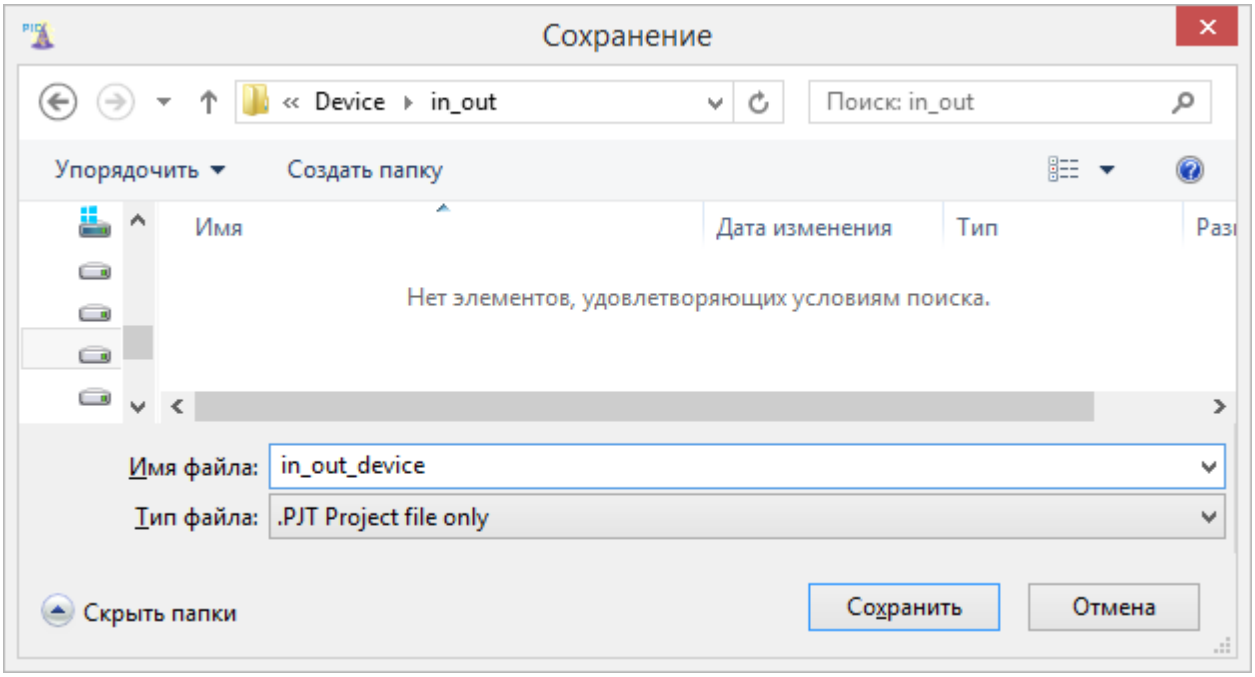

Рисунок 3.1.5 – Процес створення проекту у **IDE PCWH** за допомогою майстра **PIC Wizard**

У розділі **General** (рис. 3.1.6) вибирається цільовий мікроконтролер (список **Device**, що розкривається), його робоча частота (поле **Oscillator Frequency**), а також настроюються загальні параметри, на зразок розрядів запобігання, типу джерела системної синхронізації, порогової напруги для скидання та ін.

Для перегляду програмного коду, який буде згенерований і доданий у вихідний файл відповідно до параметрів на поточній вкладці, слід вибрати вкладку **Code**.

4. У розділі **General › Device** вибрати тип мікроконтролера, наприклад **PIC18F252** (рис. 3.1.6, рис. 3.1.7):

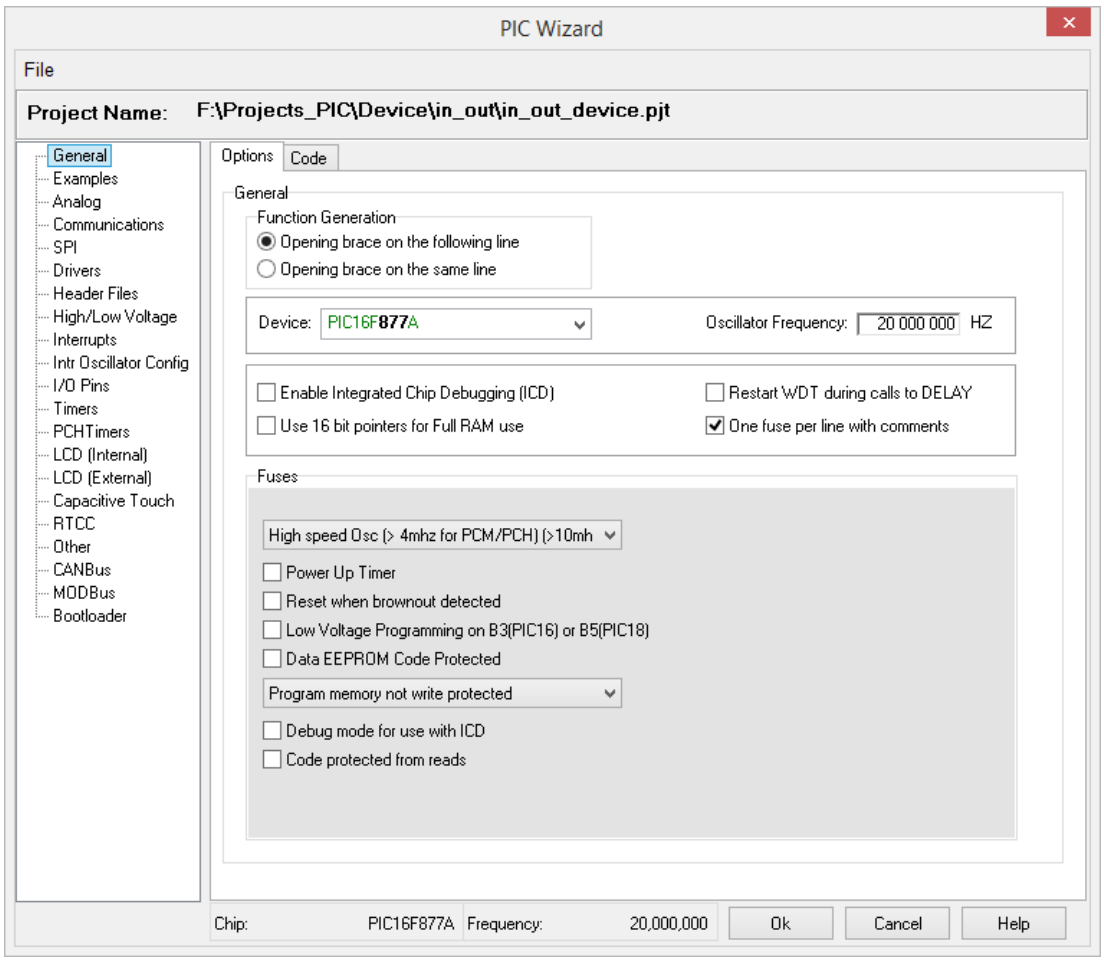

Рисунок 3.1.6 – Розділ **General** майстра **PIC Wizard**

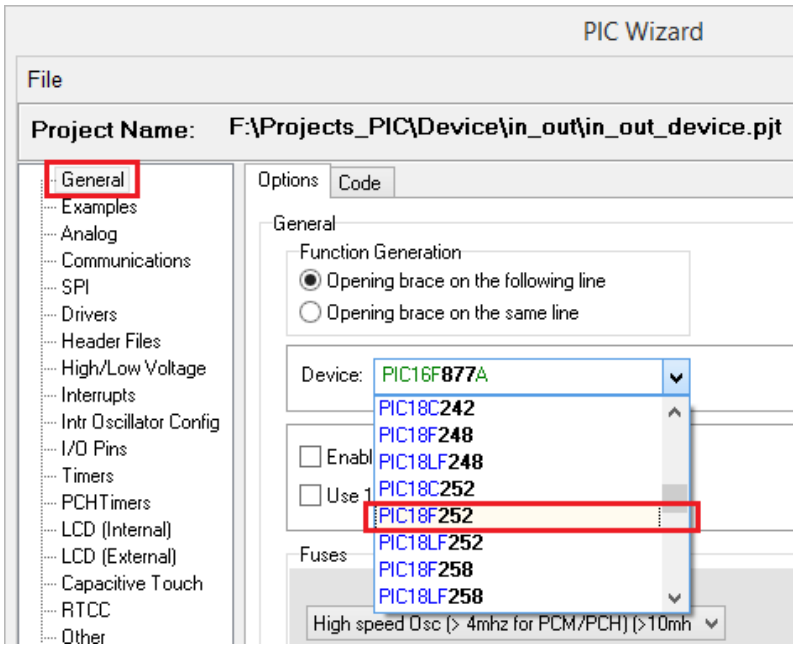

Рисунок 3.1.7 – Процес створення проекту **IDE PCWH** за допомогою майстра **PIC Wizard**

5. У розділі **General › Fuses** вибрати тип генератора **High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)** (рис. 3.1.8):

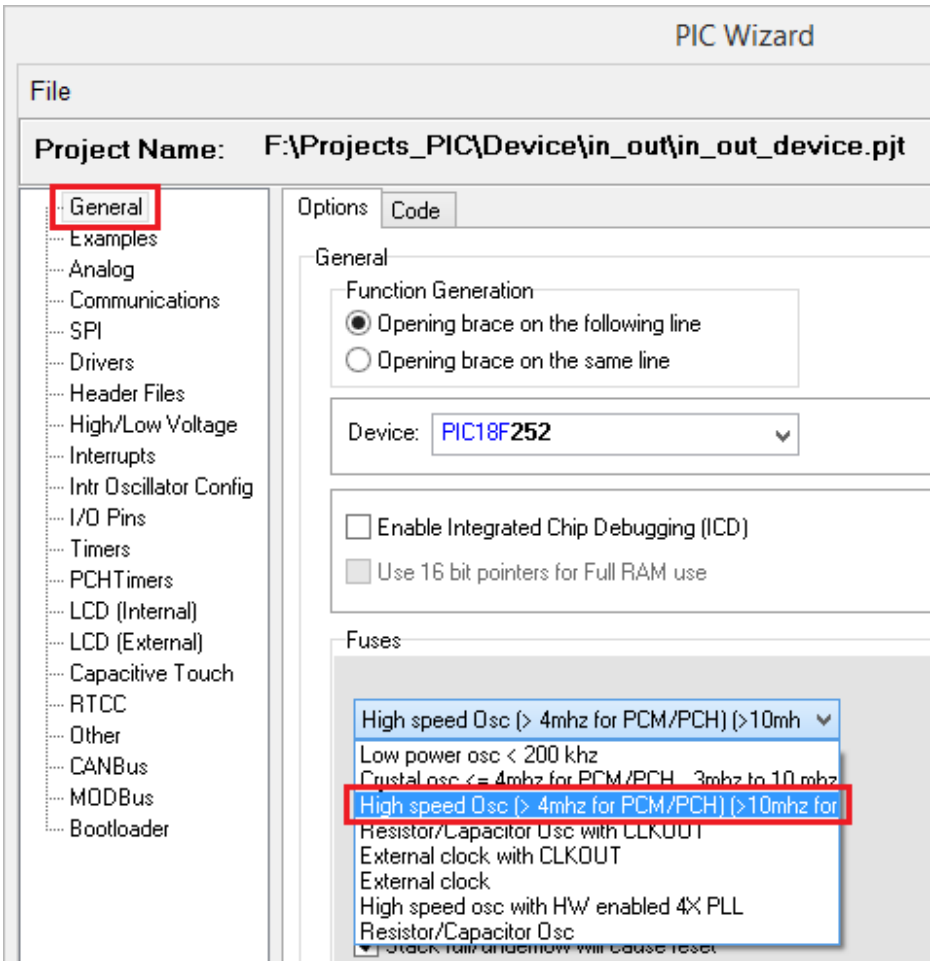

Рисунок 3.1.8 – Процес створення проекту у **IDE PCWH** за допомогою майстра **PIC Wizard**

У розділі **Communications** (рис. 3.1.9) налаштовуються параметри введення/виведення для інтерфейсів **RS-232** і **I²C** (швидкість обміну даними, відповідні лінії портів введення/виведення, перевірка помилок, режим **Master/Slave** і т.д.), а також активізується/відключається апаратний порт **PSP**.

6. У розділі **Communications** встановити мітку у полі **RS-232:**

- ☑ **Use RS-232**;

вказати:

- **Transmit:** **PIN C6** *(передача)*;
- **Receive:** **PIN C7** *(прийом)*.

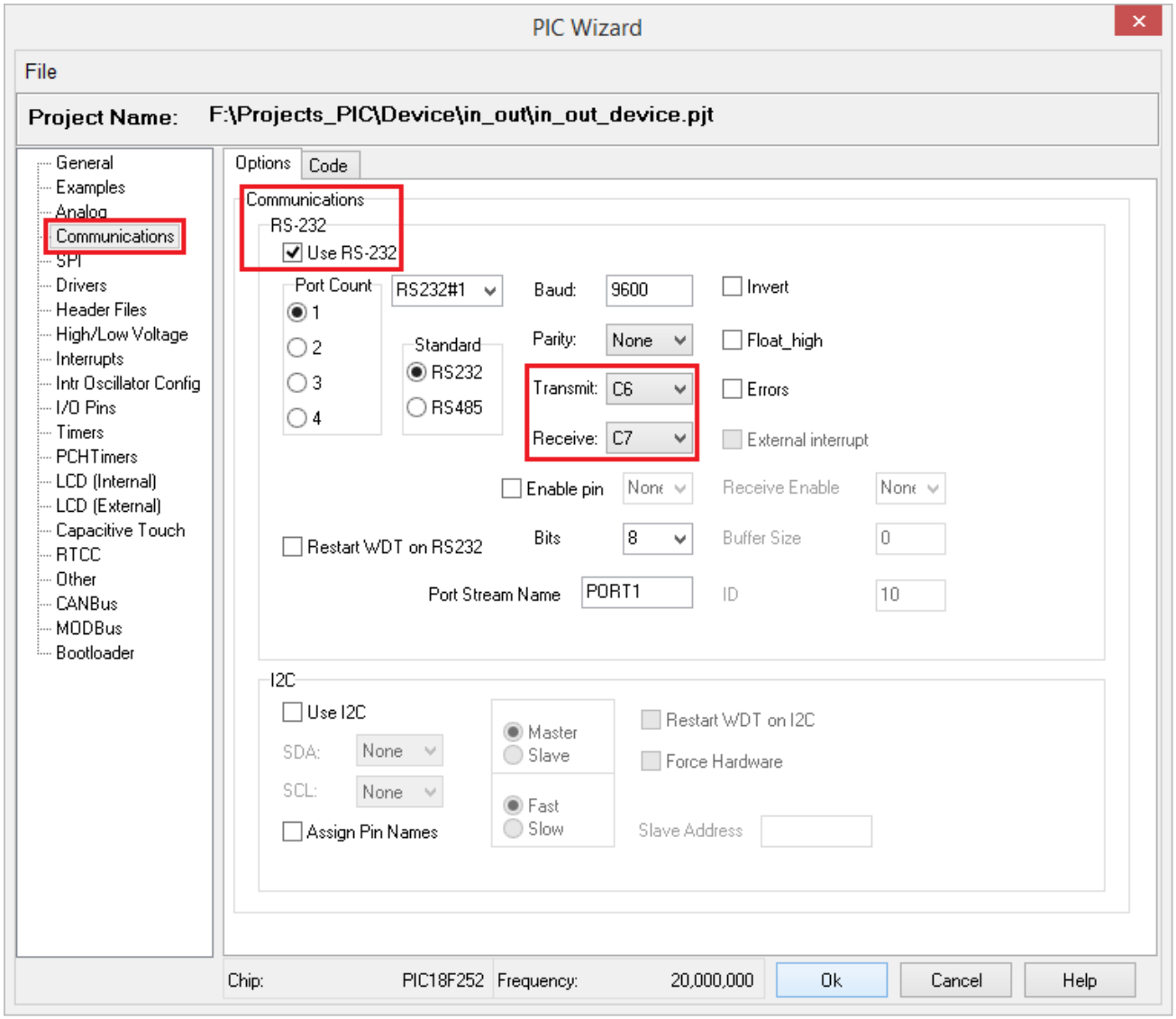

Рисунок 3.1.9 – Розділ **Communications** майстра **PIC Wizard**

Буде згенерований і доданий програмний код у вихідний файл відповідно до параметрів на поточній вкладці.

7. Для перегляду програмного коду, слід вибрати вкладку **Code** (рис. 3.1.10).

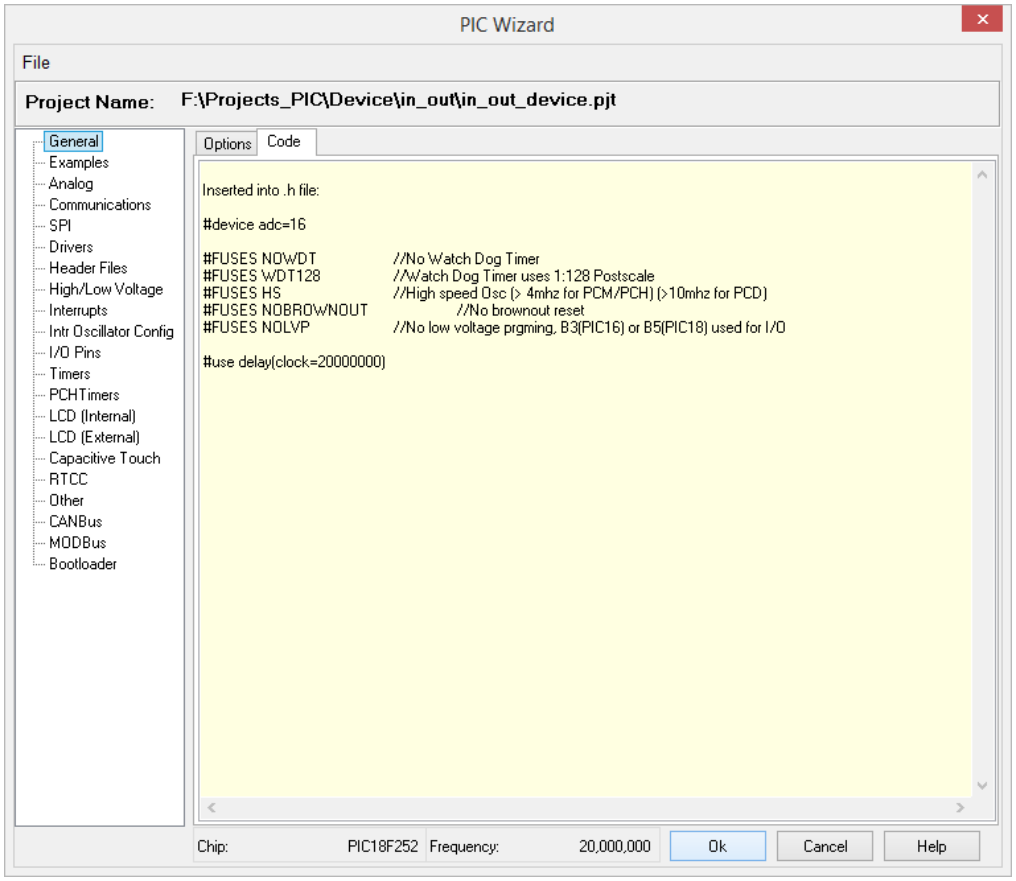

Рисунок 3.1.10 – Попередній перегляд коду, що генерується майстром **PIC Wizard** для поточного розділу **General**

8. Натиснути кнопку **OK**. Буде згенерований наступний код (рис. 3.1.11):

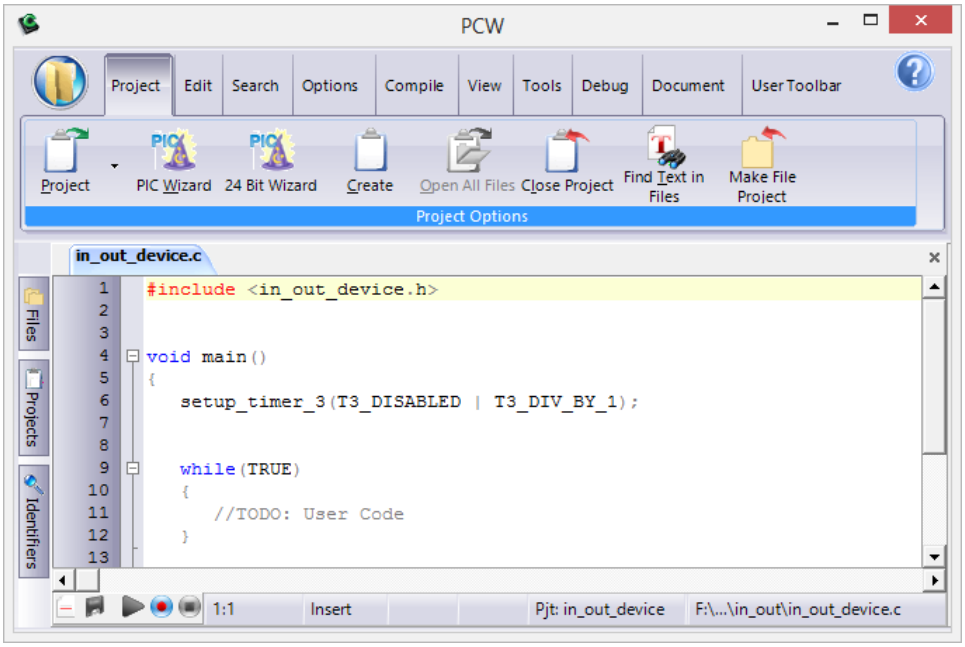

Рисунок 3.1.11 – Згенерований майстром **PIC Wizard** програмний код

Проект готовий, далі слід привести програму до наступного вигляду (лістинг 3.1.1) і використовувати як шаблон.

Можна приступати до написання програми.

Лістинг 3.1.1 – Шаблон програми

```
//#################################################################
// Найменування програми
// file: file_name.c
// Copyright (c) ПІБ / CNTU
//#################################################################

#include <in_out_device.h>
#include <swc_LCD.h>                //драйвер LCD дисплея
#CASE                               //враховувати регістр символів

//=================================================================
// Ініціалізація PIC
//=================================================================
void init()
{
setup_timer_3(T3_DISABLED | T3_DIV_BY_1);
}

//=================================================================
// Функції програми
//=================================================================
void function_N()
{
…
}
…

//=================================================================
// Main
//=================================================================
void main()
{
init();//ініціалізація мікроконтролера
lcd_init();//ініціалізація LCD дисплея

// Тут можна писати текст програми (виклики функцій)
}
```

Підключити драйвер LCD дисплея: `#include <swc_LCD.h>`

ПОЯСНЕННЯ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ «IN_OUT»

Дискретними сигналами називаються сигнали, що змінюють свій рівень стрибком, без проміжних станів.

Активний рівень сигналу із кнопок і тумблерів **АПК** – логічний "0".

Активний рівень для запалювання світлодіоду – логічна "1".

Режим введення/виведення встановлюється функцією `set_tris_x(val)`.

Функції введення/виведення даних

Мікроконтролер може зчитувати дані із зовнішнього пристрою у порти і записувати дані у зовнішній пристрій через порт **A, B** і **C**.

Напрямок передачі даних визначається функціями:

```
set_tris_x(config_word);
```

У параметрі `config_word` встановити наступні значення для кожного розряду портів **A-C**:

- `0` – передача даних;
- `1` – прийом даних.

Приклад: потрібно порт B встановити у наступний режим:

- розряди 0-3 – читання даних;
- розряди 4-7 – виведення даних.

```
set_tris_b(0b00001111);//розташування розрядів: B7 B6 B5 B4 B3 B2 B1 B0
```

або:

```
set_tris_b(0x0F);     //розташування розрядів: B7 B6 B5 B4 B3 B2 B1 B0
```

Аналогічно для інших портів.

Читання даних з порту здійснюється функціями:

```
input_x();
```

Приклад: прочитати значення сигналів на вході порту **A**:

```
int8 val;
val = input_a();
```

Аналогічно для інших портів.

Виведення даних у порт здійснюється функцією:

```
output_x();
```

Приклад: вивести у порт **B** значення 0x55:

```
int8 val;
val = 0x55;
output_b(val);
```

Аналогічно для інших портів.

Операції з бітами портів здійснюється функціями:

```
output_high(PIN);
output_low(PIN);
```

Приклад: встановити розряди 0, 3 порту **B** у стан "1", а розряди 2, 4 порту **A** у стан "0":

```
output_high(PIN_B0);
output_high(PIN_B3);
output_low(PIN_A2);
output_low(PIN_A4);
```

Примітка:

- активний рівень кнопок і тумблерів **АПК** – логічний "0";
- активний рівень для роботи світлодіодів – логічна "1".

Виведення даних на дисплей

У процесі роботи програми необхідне виведення на дисплей налагоджувальної інформації і результату виконання програми. Інформація може бути виведена у вікно термінала на PC і на LCD дисплей **АПК** (4 рядка по 20 символів).

Команди виведення у вікно термінала стандартні:

```
putc();
puts();
printf();
```

Команди читання даних стандартні:

```
getc();
gets();
```

Команди для роботи з LCD дисплеєм:

```
void lcd_putc(int8 ch);                    //вивести символ
void printf();                             //вивести форматований рядок
void lcd_goto(int8 line,int8 pos);         //перейти до рядка і
//знакомісця
void lcd_putc(int8 line, int8 pos, int8 ch); //вивести символ у
//координатах
void lcd_clear_line(int8 line);            //почистити рядок
void lcd_clear();                          //почистити всі рядки
void lcd_cursor_on();                      //включити курсор
void lcd_cursor_off();                     //виключити курсор
void lcd_bs();                             //backspace
void lcd_cursor_left();                    //курсор вліво
void lcd_cursor_right();                   //курсор вправо
void lcd_cursor_up();                      //курсор нагору
void lcd_cursor_down();                    //курсор вниз
void lcd_running_string_start(); //запустити виведення рядка, що біжить
void lcd_running_string_stop();  //зупинити виведення рядка, що біжить
```

Приклади:

Виведення рядка, що біжить:

```
lcd_clear();
lcd_running_string_begin(4);        //початок повідомлення у рядку 4
printf("Програмне забезпечення управляючих мікро-ЕОМ");
lcd_running_string_end();           //кінець повідомлення
lcd_running_string_start();         //запустити рядок, що біжить
```

Виведення символу "Q" у другому рядку у п'ятому знакомісці:

```
lcd_putc(2,5,'Q');
```

Виведення повідомлення у другому рядку із третього знакомісця:

```
lcd_goto(2,3);
printf("Hello! How do You do?");
```

Виведення повідомлення на кілька рядків:

```
printf("1234567890\n 1234567890\n");
```

Повна принципова схема з'єднань для виконання всіх лабораторних робіт представлена на рис. 3.1.12.

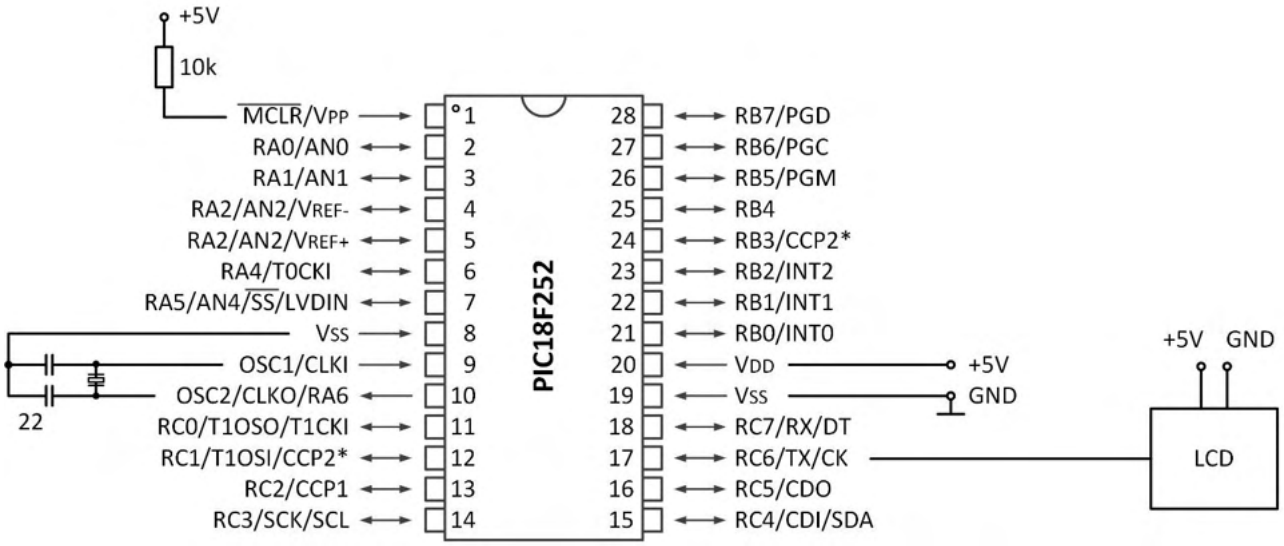

Рисунок 3.1.12 – Повна принципова схема з'єднань для виконання всіх лабораторних робіт **«IN_OUT»**

Приклад комутації з'єднань **апаратно-програмного комплексу** представлений на рис. 3.1.13:

Рисунок 3.1.13 – Комутації з'єднань **АПК** для лабораторної роботи **«IN_OUT»**

Шаблон побудови та оформлення програми

Лістинг 3.1.2 – Шаблон побудови та оформлення програми

```
//#################################################################
// Найменування програми
// file: file_name.c
// Copyright (c) ПІБ студента / CNTU
//#################################################################

#include <in_out_device.h>
#include <swc_LCD.h>                //драйвер LCD дисплея
#CASE                               //враховувати регістр символів
//=================================================================
// Назва функції (Функція, що викликається з функції main())
//=================================================================
void init()
{
…
}


//=================================================================
// Назва функції (Функція, що викликається з функції function_1())
//=================================================================
void function_1_1()
{
…
}
//=================================================================
// Назва функції (Функція, що викликається з функції main())
//=================================================================
void function_1()
{
…
function_1_1();
…
}
//=================================================================
// Назва функції (Функція, що викликається з функції main())
//=================================================================
void function_2()
{
…
}
//=================================================================
// Main
//=================================================================
void main()
{
init();                             //ініціалізація
…
for(;;){
function_1();                       //виклик функції
function_2();                       //виклик функції
}
}
```

Приклад виконання лабораторної роботи «IN_OUT» відповідно до варіанту завдання для лабораторної роботи «IN_OUT»

Алгоритм програми для лабораторної роботи **«IN_OUT»** має вигляд (рис. 3.1.14):

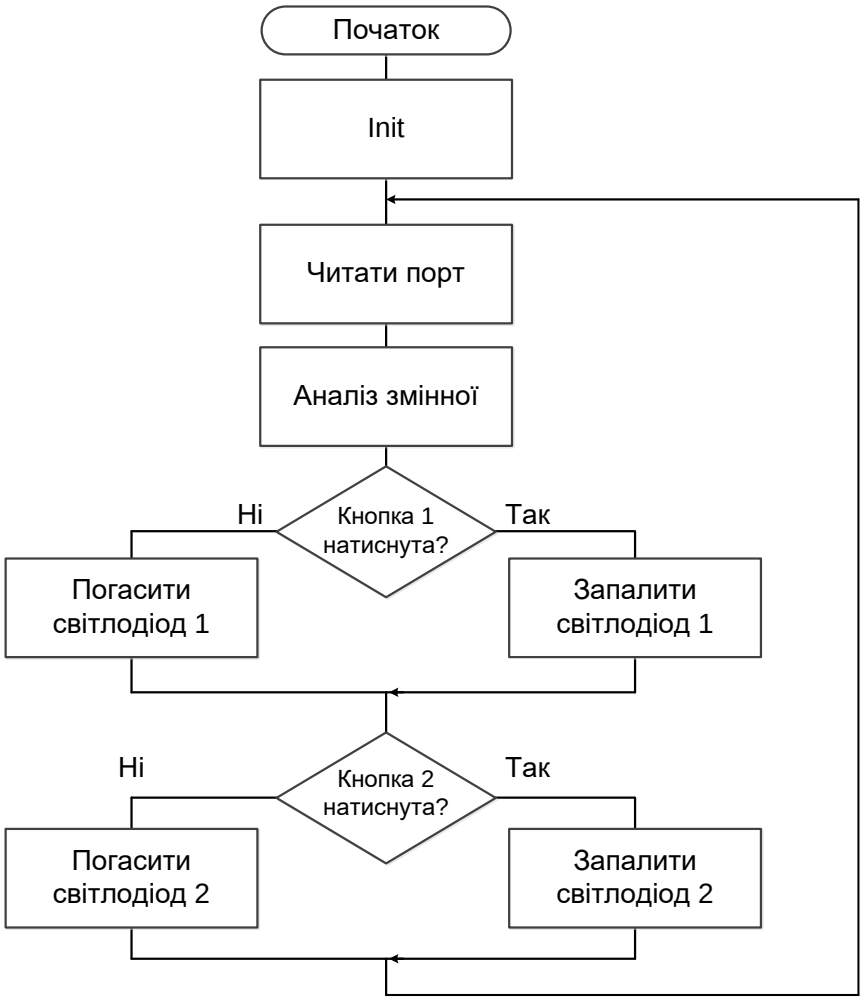

Рисунок 3.1.14 – Алгоритм програми лабораторної роботи **«IN_OUT»**

Програма має вигляд і структуру відповідно до лістингів 3.1.3, 3.1.4:

- лістинг 3.1.3 – для варіанту **АПК**;
- лістинг 3.1.4 – для варіанту **«Proteus»**.

Варіант завдання для виконання лабораторної роботи **«IN_OUT»** представлений у таблиці 3.1.2.

Таблиця 3.1.2 – Варіант завдання для виконання лабораторної роботи **«IN_OUT»**

Введення (кнопки)								**Виведення (світлодіоди)**							
A0	A1	A2	A3	B0	B1	B2	B3	B0	B1	B2	B3	B4	B5	B6	B7
	+	+												+	+

Принципова схема підключень для варіанту АПК

Принципова схема підключень для варіанту **АПК**, для виконання лабораторної роботи **«IN_OUT»** виглядає у такий спосіб (рис. 3.1.15):

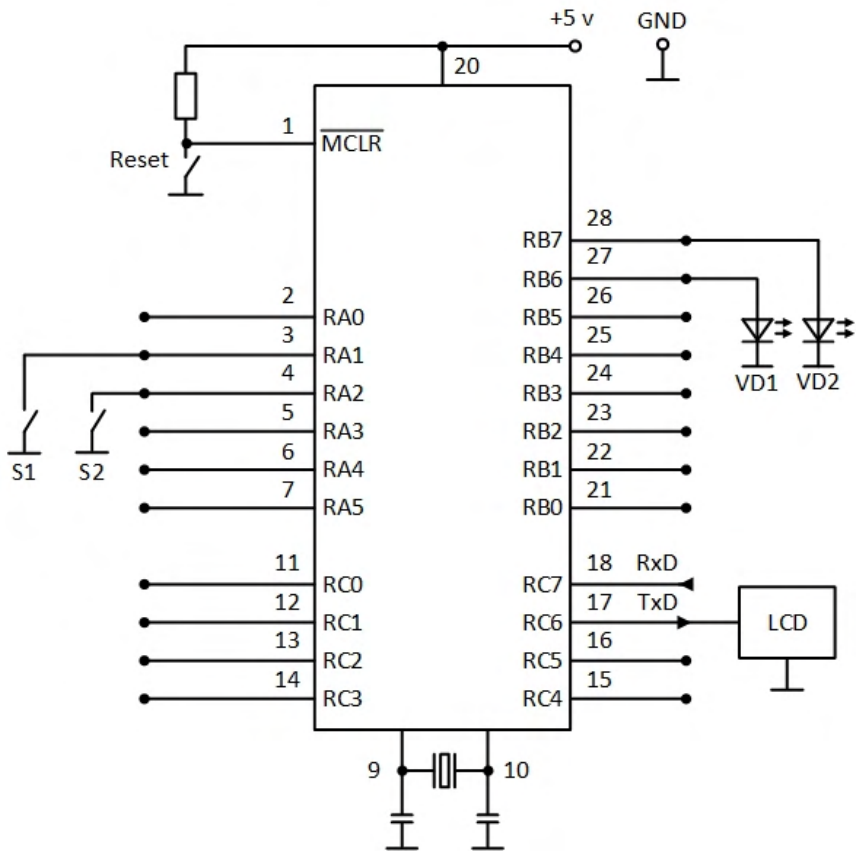

Рисунок 3.1.15 – Принципова схема підключень для лабораторної роботи **«IN_OUT»** для варіанту **АПК**

Принципова схема підключень для варіанту «Proteus»

Принципова схема для варіанту **«Proteus»**, для виконання лабораторної роботи **«IN_OUT»** виглядає у такий спосіб (рис. 3.1.16):

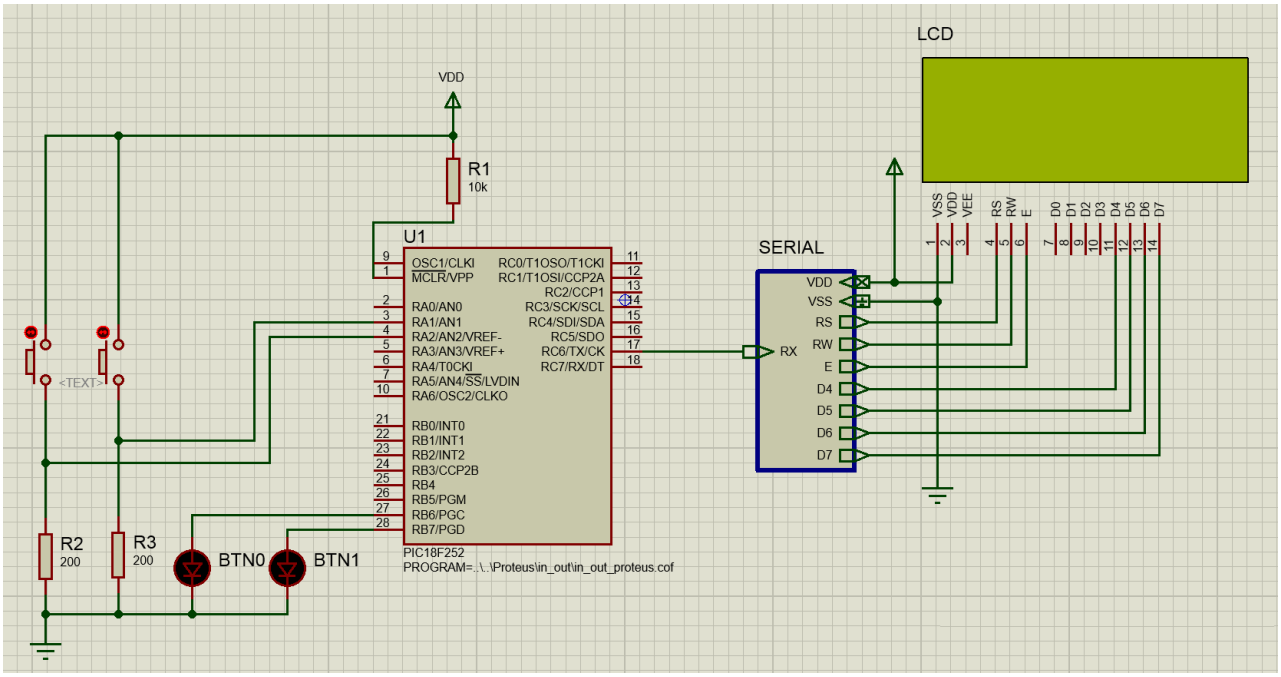

Рисунок 3.1.16 – Принципова схема підключень для лабораторної роботи **«IN_OUT»** для варіанту **«Proteus»**

Результат виконання лабораторної роботи «IN_OUT» для варіанту АПК

Лістинг 3.1.3 – Програма реалізації алгоритму (рис. 3.1.14) лабораторної роботи **«IN_OUT»** для варіанту **АПК**

```
//###################################################################
// Лабораторна робота «IN_OUT»
// file: in_out_device.c
// Copyright (c) Docent Smirnov V.V., Docent Smirnova N.V. / CNTU
//###################################################################

#include <in_out_device.h>
#include <swc_LCD.h>          //драйвер LCD дисплея
#CASE                         //враховувати регістр символів
//===================================================================
// Ініціалізація PIC
//===================================================================
void init()
{
     setup_timer_3(T3_DISABLED | T3_DIV_BY_1);
}

//===================================================================
// Запалити світлодіод 1
//===================================================================
void blink1()
{
     output_high(PIN_B6);
     delay_ms(50);
```

```
     output_low(PIN_B6);
     delay_ms(50);
}
//=============================================================================
// Запалити світлодіод 2
//=============================================================================
void blink2()
{
     output_high(PIN_B7);
     delay_ms(100);
     output_low(PIN_B7);
     delay_ms(100);
}
//=============================================================================
// Виведення заголовків
//=============================================================================
void titles()
{
     //====== виведення заголовка на дисплей
     lcd_goto(2,1);
     printf("Кафедра ПКСМ");
     lcd_goto(3,1);
     printf("Смірнов В.В.");
     lcd_goto(4,1);
     printf("Натиснута кнопка: ");

     //====== Виведення на дисплей рядка, що біжить
     lcd_running_string_begin(1);   // початок повідомлення у рядку 4
     printf("Лабораторна робота IN_OUT");
     lcd_running_string_end();      // кінець повідомлення
     lcd_running_string_start();    // запустити біжучий рядок
}

//=============================================================================
// Введення/виведення сигналів
//=============================================================================
void in_out()
{
      int8 p;

      //=== читати порт у змінну
      p = input_a();
      //=== перевірити біт 1 на натискання клавіші (1)
      // варіант 1
      //test = bit_test(p,1);
      //if(test == 0)
      // варіант 2
      //if(!bit_test(p,1))

      // варіант 3
      if(bit_test(p,1) == 0)
      {
            //=== якщо 0 запалити світлодіод 1
            blink1();
            //=== виведення на дисплей
            lcd_goto(4,19);
            printf("A1");
      }else{
            //=== погасити світлодіод 1
            output_low(PIN_B6);
      }
```

```
      //=== перевірити біт 2 на натискання клавіші (0)
      // варіант 1
      //test = bit_test(p,2);
      // варіант 2
      //if(test == 0)

      // варіант 3
      if(!bit_test(p,2))
      {
            //=== якщо 0 запалити світлодіод 2
            blink2();
            //=== виведення на дисплей
            lcd_goto(4,19);
            printf("A2");
      }else{
            //=== погасити світлодіод 2
            output_low(PIN_B7);
      }
}
//=====================================================================
// Main
//=====================================================================
void main()
{
      init();                        // ініціалізація мікроконтролера
      //====== конфігурація портів
      set_tris_a(0b11111111);
      set_tris_b(0b00111111);
      //====== затримка для lcd дисплея
      delay_ms(1000);

      //====== вивести заголовки
      titles();
      //====== головний цикл
      for(;;){
            in_out();
      }
}
```

Результат виконання лабораторної роботи «IN_OUT» для варіанту «Proteus»

Лістинг 3.1.4 – Програма реалізації алгоритму (рис. 3.1.14) лабораторної роботи **«IN_OUT»** для варіанту **«Proteus»**

```
//#################################################################
// Лабораторна робота «IN_OUT»
// file: in_out_proteus.c
// Copyright (c) Docent Smirnov V.V., Docent Smirnova N.V. / CNTU
//#################################################################

#include <in_out_proteus.h>
#include <swc_LCD.h>               //драйвер LCD дисплея
#CASE                              //враховувати регістр символів
```

```
//=============================================================================
// Ініціалізація PIC
//=============================================================================
void init()
{
setup_timer_3(T3_DISABLED | T3_DIV_BY_1);
}
//=============================================================================
// Запалити світлодіод 1
//=============================================================================
void blink1()
{
int i;
output_high(PIN_B6);
output_low(PIN_B7);
delay_ms(300);
}

//=============================================================================
// Запалити світлодіод 2
//=============================================================================
void blink2()
{
int i;
output_high(PIN_B7);
output_low(PIN_B6);
delay_ms(300);
}
//=============================================================================
// Main
//=============================================================================
void main()
{
int8 p;
init();
//====== конфігурація портів
set_tris_b(0b00111111);
output_low(PIN_B6);
output_low(PIN_B7);
//====== головний цикл
for(;;){
lcd_goto(1,1);
printf("IN_OUT");
lcd_goto(2,1);
printf("CS&NP Department");
lcd_goto(3,1);
printf("Docent Smirnov V.V.");

//=== читати порт у змінну
p = input_b();
//=== перевірити біт 0 на натискання клавіші (0)
if(bit_test(p,0))
{
//=== виведення на дисплей
lcd_goto(4,1);
printf("Button: PIN B0");
//=== якщо 0 запалити світлодіод 1
blink1();
}
//=== перевірити біт 1 на натискання клавіші (0)
if(bit_test(p,1))
{
//=== виведення на дисплей
lcd_goto(4,1);
```

```
printf("Button: PIN B1");
//=== якщо 0 запалити світлодіод 2
blink2();
}
delay_ms(200);
}}
```

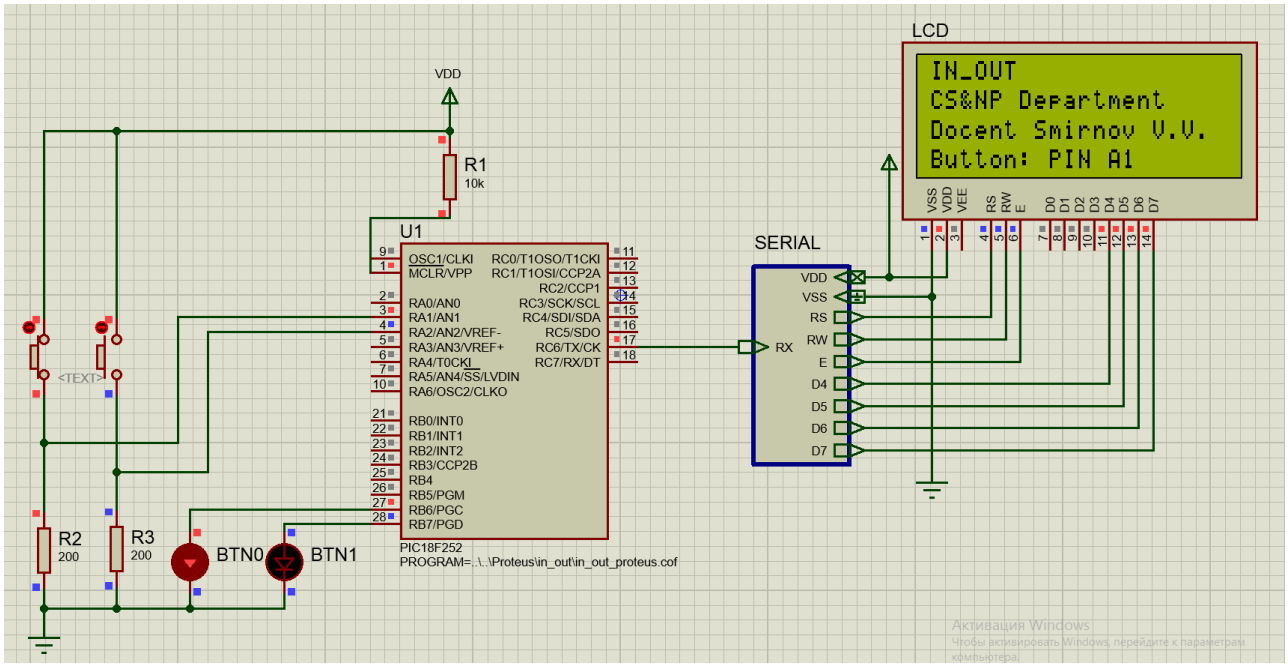

Рисунок 3.1.17 – Результат виконання лабораторної роботи **«IN_OUT»** для варіанту **«Proteus»**

Контрольні питання

1) дати коротку характеристику мікроконтролера PIC18F252;
2) як управляти напрямком введення/виведення даних у мікроконтролері?
3) які команди здійснюють виведення інформації на LCD?
4) у яку пам'ять завантажується код програми?

ЛАБОРАТОРНА РОБОТА № 2 - «INTERRUPTS»

Тема: Робота з перериваннями мікроконтролера

Ціль роботи:

Отримання навичок роботи з перериваннями мікроконтролера, програмування обробників переривань мікроконтролерів, написання та налагодження програми обробників переривань порту RS-232, таймера, зовнішніх переривань.

Завдання лабораторної роботи

- для варіанту **АПК** намалювати принципову схему підключень відповідно до варіанту для виконання лабораторної роботи (таб. 3.2.1);
- для варіанту **«Proteus»** використовувати готову схему відповідно до варіанту для виконання лабораторної роботи;
- створити алгоритм програми з обробниками переривань;
- здійснити виклик переривань із кнопок і тумблерів **АПК** та обробити їх відповідно до варіанту для виконання лабораторної роботи;
- написати програму секундоміра в обробнику переривання таймера;
- здійснити приймання рядка символів з термінальної програми на PC по інтерфейсу RS-232, що містить ПІБ студента;
- здійснити виведення результатів на LCD і на світлодіоди з обробників переривань відповідно до варіанту для виконання лабораторної роботи.

Виведення на LCD повинне містити:

- номер лабораторної роботи;
- групу студента;
- прізвище та ініціали студента;
- секунди і хвилини в обробнику переривання таймера;

- номер зовнішнього переривання в обробнику зовнішнього переривання відповідно до варіанту для виконання лабораторної роботи;
- рядок символів в обробнику переривання порту RS-232.

Варіанти для виконання лабораторної роботи «INTERRUPTS»

Таблиця 3.2.1 – Варіанти для виконання лабораторної роботи **«INTERRUPTS»**

№	Переривання				
	RTCC	RS-232	INT0	INT1	INT2
1	+	+	+	+	
2	+	+	+		+
3	+	+		+	+
4	+	+	+		+
5	+	+	+	+	
6	+	+	+		+
7	+	+		+	+
8	+	+	+		+
9	+	+	+	+	
10	+	+	+		+
11	+	+		+	+
12	+	+	+		+
13	+	+	+	+	
14	+	+	+		+
15	+	+		+	+

Послідовність виконання лабораторної роботи для варіанту АПК

1) для варіанту **АПК** намалювати принципову схему підключень відповідно до варіанту для виконання лабораторної роботи;
2) створити свій директорій для файлів проекту;
3) відкрити інтегроване середовище розробки **PCWH**;
4) створити проект у інтегрованому середовищі розробки **PCWH**;
5) створити алгоритм програми з обробниками переривань;

6) написати програму мовою програмування **C**, що реалізує створений алгоритм в обробнику переривань, відповідно до варіанту для виконання лабораторної роботи;
7) скомпілювати програму, одержати бінарні файли з розширеннями ***.HEX** і ***.COF** (файли з розширеннями ***.HEX** і ***.COF** створюються підчас компіляції програми);
8) завантажити бінарний файл з розширенням ***.HEX** у пам'ять програм мікроконтролера програмою **PICkit.exe**;
9) на **АПК** зкумутувати схему підключень (кнопки і світлодіоди) відповідно до варіанту для виконання лабораторної роботи.

Примітка: *Кнопками здійснювати виклик переривань INT0, INT1, INT2. Переривання порту RS-232 здійснювати шляхом посилки рядка символів з терміналу SIOW.exe або Terminal.exe;*

10) виконати програму.

Послідовність виконання лабораторної роботи для варіанту «Proteus»

1) для середовища моделювання **«Proteus»** використовувати готову схему;
2) створити свій директорій для файлів проекту;
3) відкрити інтегроване середовище розробки **PCWH**;
4) створити проект у інтегрованому середовищі розробки **PCWH**;
5) створити алгоритм програми з обробниками переривань;
6) написати програму мовою програмування C, що реалізує створений алгоритм в обробнику переривань, відповідно до варіанту для виконання лабораторної роботи;
7) скомпілювати програму, одержати бінарні файли з розширеннями ***.HEX** і ***.COF** (файли з розширеннями ***.HEX** і ***.COF** створюються підчас компіляції програми);

8) відкрити середовище моделювання **«Proteus»**;
9) у папці **«Proteus_students»** вибрати папку лабораторної роботи **«INTERRUPTS»**;
10) відкрити файл проекту з розширенням ***.DSN**;
11) у середовищі моделювання **«Proteus»** змінити схему підключень відповідно до варіанту для виконання лабораторної роботи;
12) завантажити заздалегідь скомпільований бінарний файл з розширеннями ***. COF** або ***. HEX** у мікроконтролер;
13) у середовищі моделювання **«Proteus»** виконати програму.

Послідовність виконання лабораторної роботи для варіанту «Proteus» з використанням шаблону програми

1) для середовища моделювання **«Proteus»** використовувати готову схему;
2) відкрити папку **«Proteus_students»**;
3) відкрити інтегроване середовище розробки **PCWH**;
4) у папці **«Proteus_students»** вибрати папку лабораторної роботи **«INTERRUPTS»**;
5) відкрити файл проекту з розширенням ***.pjt**;
6) у редакторі **IDE PCWH** відкрити шаблон файлу програми з розширенням ***.c**;
7) відкрити середовище моделювання **«Proteus»**;
8) у папці **«Proteus_students»** вибрати папку лабораторної роботи **«INTERRUPTS»**;
9) відкрити файл проекту з розширенням ***.DSN**;
10) у середовищі моделювання **«Proteus»** змінити схему підключень відповідно до варіанту для виконання лабораторної роботи;
11) створити алгоритм програми з обробниками переривань;
12) у інтегрованому середовищі розробки **PCWH**, використовуючи шаблон програми самостійно написати програму мовою

програмування **C**, що реалізує створений алгоритм в обробнику переривань, відповідно до варіанту для виконання лабораторної роботи;

13) скомпілювати програму, одержати бінарні файли з розширеннями ***.HEX** і ***.COF** (файли з розширеннями ***.HEX** і ***.COF** створюються підчас компіляції програми);
14) у середовищі моделювання **«Proteus»** виконати програму.

Примітка: *файл демонстрації виконання лабораторної роботи* ***«INTERRUPTS»*** *розташований на сайті* http://pksm.kntu.kr.ua/SCME.html.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Переривання

Переривання - це реакція мікроконтролера на внутрішні або зовнішні події. В результаті переривання виконання програми зупиняється і відбувається перехід до відповідної підпрограми обробки переривання (функції).

Переривання бувають *внутрішніми, зовнішніми, апаратними* та *програмними*.

Джерелами *внутрішнього* переривання є вбудовані модулі мікроконтролера (наприклад, таймер/лічильник або сторожовий таймер), або події програми.

Зовнішні переривання викликаються скиданням (сигнал на виводі RESET) або сигналами встановленого рівня на виводах INT та інших.

У момент виникнення переривання у стек поміщається адреса повернення - адреса команди, яка повинна бути виконана першою після виходу з підпрограми обробки переривань.

Мікроконтролери **PIC18FXX2** мають кілька джерел переривань і функцію пріоритетної системи переривань, яка дозволяє для кожного джерела переривань призначити високий або низький пріоритет.

При виникненні переривання з високим пріоритетом відбувається перехід по вектору 000008h, а при виникненні переривання з низьким пріоритетом - 000018h. Переривання з високим пріоритетом припиняють обробку переривань з низьким пріоритетом.

У мікроконтролерах **PIC18F252** передбачено 10 регістрів спеціального призначення для управління переривань:

- `RCON;`
- `INTCON;`
- `INTCON2;`
- `INTCON3;`
- `PIR1, PIR2` - містить прапори запитів на переривання;
- `PIE1, PIE2` - містить прапори дозволу переривань від периферійних пристроїв;
- `IPR1, IPR2` - задають пріоритети переривань від периферійних пристроїв.

Кожному джерелу переривань відповідає три керуючих біта:

- прапор переривань, вказує на те, що виконана умова виникнення переривання;
- біт дозволу переривання, дозволяє перехід по вектору переривання при встановленні відповідного прапору;
- біт пріоритету, вибір низького або високого пріоритету переривання.

Послідовний інтерфейс RS-232

Послідовний інтерфейс RS-232 - це промисловий стандарт EIA RS-232-C, CCITT V.24 для послідовної двобічної асинхронної передачі даних.

У послідовному інтерфейсі RS-232 дані передаються послідовно, тобто по лінії зв'язку кожен біт посилається по черзі після надсилання попереднього біту.

Для передачі між персональним комп'ютером і мікроконтролером потрібні тільки 3 лінії (рис 3.2.1):

- для передачі **T_X** (Transmit);
- для прийому **R_X** (Receive);
- для загальної шини **GND** (Ground).

Просте підключення через послідовний інтерфейс RS-232/UART

У персональному комп'ютері є роз'єм USB, до якого через перехідник USB-UART може приєднуватися кабель для послідовної передачі даних.

Схема підключення мікроконтролера до персонального комп'ютера через послідовний інтерфейс показана на рис. 3.2.1.

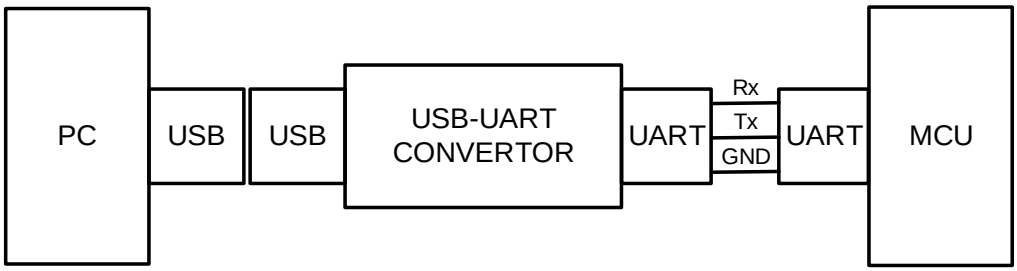

Рисунок 3.2.1 - Підключення мікроконтролера до PC
за допомогою послідовного інтерфейсу

У багатьох випадках вибирають саме цей варіант підключення, який дозволяє швидко зібрати необхідну електричну схему підключення апаратних засобів і досить просто розмістити керуючу програму у FLASH-пам'яті мікроконтролера.

Формат передачі даних RS-232/UART

Щоб дані могли послідовно передаватися по лінії, для них повинен бути визначений момент початку і кінця слова. Є різні види протоколів передачі по інтерфейсу. Проте в будь-якому випадку передача починається зі стартового біту і закінчується стоповим бітом. Формат даних і швидкість передачі, що використовуються у передавачі, повинні бути встановлені точно такими ж, як і

у приймачі, оскільки інакше не може бути забезпечена надійна передача. Тому при проблемах передачі спочатку потрібно перевіряти всі налаштування. При найпростішому способі передачі застосовується 1 стартовий біт, 8 біт даних і 1 стоповий біт.

Якщо ж потрібна найбільш достовірна передача, то може бути доданий ще біт парності, який передається після слова даних. Біт парності призначений для захисту від помилок, який визначає окремі помилки при передачі даних. Якщо при передачі буде пошкоджений один біт, і відповідно невірно визначений, то ця помилка може бути визначена. Якщо ж будуть пошкоджені два або більша кількість бітів, то за допомогою біту парності визначити такі помилки вже не представляється можливим. У випадку використання біту парності можна вибирати між перевіркою на парність і непарність. При перевірці на парність (EvenParity) у біті парності буде логічний "0", якщо сума одиниць у слові даних парна, або буде логічна "1", якщо сума непарна. При перевірці на непарність - все в точності навпаки.

У багатьох випадках від передачі біту парності відмовляються, т.я. у цьому випадку потрібні додаткові витрати на перевірку або установку біту відповідним чином.

Після того як визначено формат для передачі даних, потрібно встановлювати швидкість, з якою повинні передаватися дані. Оскільки загальна лінія з тактовим сигналом відсутня, то перед початком обміну даними потрібно повідомляти передавача і приймача, з якою швидкістю буде здійснюватися передача. Тому таку передачу називають асинхронною. Швидкість тактування або швидкість передачі вказується у бодах. Не можна плутати цю швидкість зі швидкістю передачі даних, т. я. при цьому мова йде про кількість корисних даних.

При швидкості передачі 19 200 бод або 19,2 кбод рівень сигналу змінюється 19 200 раз в секунду. Так як для передачі 1 байта ще потрібні стартовий і стоповий біт і швидкість передачі даних становить максимум 8/10 швидкості передачі. З цього випливає, що в цьому прикладі можуть передаватися

максимум 15 360 біт в секунду або 1 920 байт в секунду. Оскільки в цьому випадку мається на увазі асинхронна передача даних, то стартовий біт може надсилатися в будь-який час. Приймач повинен тоді під час обміну зчитувати дані з тією ж самою швидкістю, з якою вони посилаються передавачем.

В інтерфейсі RS-232 дані, що передаються, повинні мати більш високі рівні напруги, ніж рівні ТТЛ (транзисторно-транзисторної логіки) в UART. Для передачі окремих бітів використовуються позитивні і негативні напруги. Причому рівень між +3 В і +15 В інтерпретується як низький логічний рівень (0, Low, Space), а напруги між -3 В і -15 В як високий (1, High, Mark). Як правило, для передачі використовуються напруги +12 В і -12 В.

На рис. 3.2.2 представлена процедура передачі даних по інтерфейсу UART / RS-232. Передача здійснюється молодшим розрядами вперед.

Параметри: 8 біт, біт парності - немає, стоп-біт - "1". У прикладі передається латинська буква "M" в ASCII-коді ("M" = 0x4D або 0b01001101).

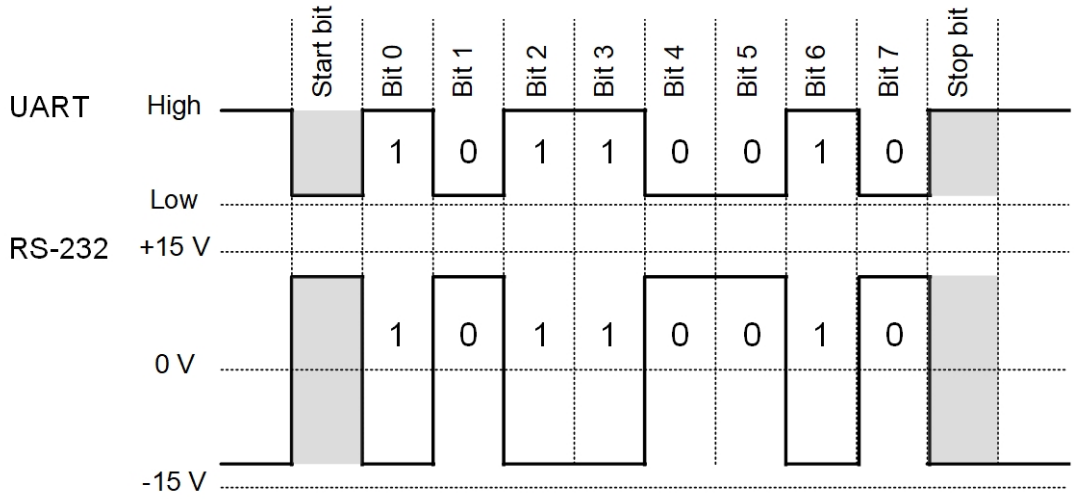

Рисунок 3.2.2 - Передача даних по інтерфейсу UART / RS-232

Процедура обміну даними між пристроями по інтерфейсу RS-232

Процедура обміну даними між пристроями по інтерфейсу RS-232 передбачає використання ліній протокольної взаємодії.

Таблиця 3.2.2 - Відповідність сигналів контактам для 9-ти і 25-ти контактних роз'ємів

Найменування	**Напрямок**	**Опис**	**Контакт (25-контактний роз'єм)**	**Контакт (9-контактний роз'єм)**
DCD	IN	Carrie Detect (Визначення несучої)	8	1
RXD	IN	Receive Data (Дані, що приймаються)	3	2
TXD	OUT	Transmit Data (Передані дані)	2	3
DTR	OUT	Data Terminal Ready (Готовність терміналу)	20	4
GND	-	System Ground (Корпус пристрою)	7	5
DSR	IN	Data Set Ready (Готовність даних)	6	6
RTS	OUT	Request to Send (Запит на відправку)	4	7
CTS	IN	Clear to Send (Готовність прийому)	5	8
RI	IN	Ring Indicator (Індикатор)	22	9

Лінії *«запит на пересилку»* (RTS) і *«ініціювання пересилання»* (CTS) використовуються зазвичай для управління *потоками даних*, що передаються між *комп'ютерами* (DCE) і *модемами* (DTE). Після підготовки комп'ютера до передачі даних він активізує лінію RTS. Якщо *кінцевий пристрій* (DTE) готовий до прийому даних, він формує сигнал CTS. Якщо комп'ютер не в змозі прийняти дані (наприклад, через те, що його буфер заповнений або виконуються будь-які операції по обробці даних), сигнал на лінії RTS видаватися не буде, тим самим відповідний пристрій повідомляється про неможливість прийому комп'ютером додаткової інформації.

Лінії *«готовність терміналу»* (DTR) і *«готовність модему»* (DSR) зазвичай застосовуються для підготовки *сеансу передачі даних*. У випадку готовності до взаємодії з кінцевим пристроєм (DTE) комп'ютер видає в лінію DTR відповідний сигнал (повідомлення). Якщо кінцевий пристрій може прийняти дані, він формує сигнал у лінії DSR для повідомлення комп'ютера про готовність до сеансу передачі даних. При виникненні будь-яких помилок, пов'язаних з апаратними засобами, цей пристрій скасовує повідомлення у лінії DSR для повідомлення комп'ютера про виниклі проблеми. Аналогічним чином при зникненні *сигналу несучої* модеми скасують повідомлення DSR.

У лінії *«виявлення сигналу несучої»* (DCD) повідомлення формується, коли модемом встановлено зв'язок з іншими пристроями (модемом). За допомогою лінії «індикація сигналу виклику» (RI) комп'ютер інформується про генерацію *сигналів виклику*.

Ці лінії, поряд з лініями *«рукостискання»*, досить рідко використовуються у додатках з PIC-мікроконтролерами.

Пристрій передачі даних (DCE) і *кінцевий пристрій* (DTE) завжди пов'язані спільним («земляним») проводом. Ця лінія виявляється досить критичною для інтерфейсу RS-232, від неї залежить робота вхідних *перетворювачів рівнів*, за допомогою яких визначаються реальні логічні рівні вхідних напруг ліній.

ПРИКЛАД СТВОРЕННЯ ПРОЕКТУ У ІНТЕГРОВАНОМУ СЕРЕДОВИЩІ РОЗРОБКИ PCWH

Для виконання лабораторної роботи **«INTERRUPTS»**, можна згенерувати новий проект автоматично за допомогою майстра **PIC Wizard**, який викликається по команді меню **Project › PIC Wizard**, або використовувати шаблон програми, що знаходиться у папці **«Proteus_students»** в якій вибрати папку лабораторної роботи **«INTERRUPTS»**.

Створення проекту у інтегрованому середовищі розробки PCWH за допомогою майстра PIC Wizard

Для запуску майстра **PIC Wizard** слід виконати відповідну команду меню **Project**, а потім вказати розміщення і ім'я головного файлу проекту.

В результаті відкриється вікно майстра, що складається з розділів з параметрами проекту (рис. 3.2.7).

Зовнішній вигляд вікна майстра може відрізнятися в залежності від версії компілятора **CCS-PICC** і обраного типу мікроконтролера.

1. Запустити **IDE PCWH**, натиснути кнопку **Projects** майстра **PIC Wizard** (рис. 3.2.4):

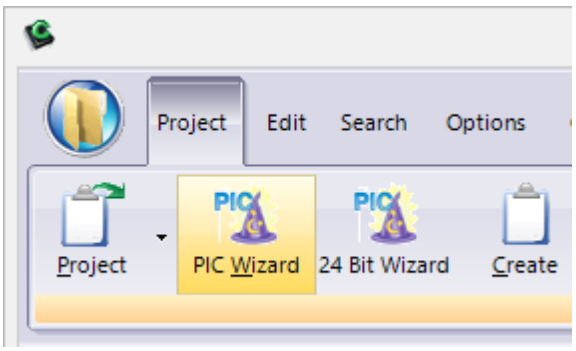

Рисунок 3.2.4 – Процес створення проекту у **IDE PCWH** за допомогою майстра **PIC Wizard**

або натиснути кнопку у вигляді відкритої папки 

2. Вибрати: **New › Project Wizard** (рис. 3.2.5):

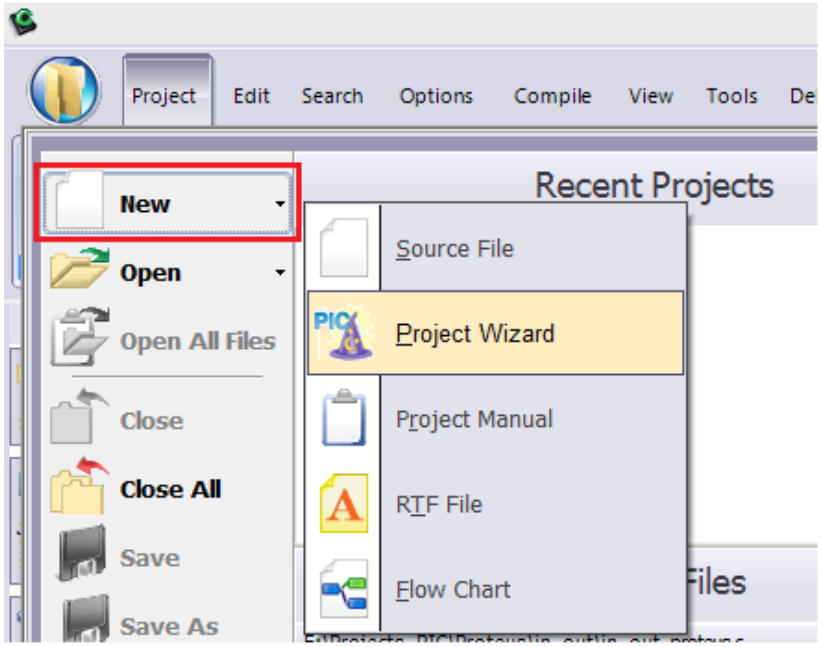

Рисунок 3.2.5 – Процес створення проекту у **IDE PCWH** за допомогою майстра **PIC Wizard**

3. Створити або вибрати свій директорій і записати у нього проект під будь-яким іменем латинським шрифтом, наприклад: «**interrupts.pjt**» (рис. 3.2.6):

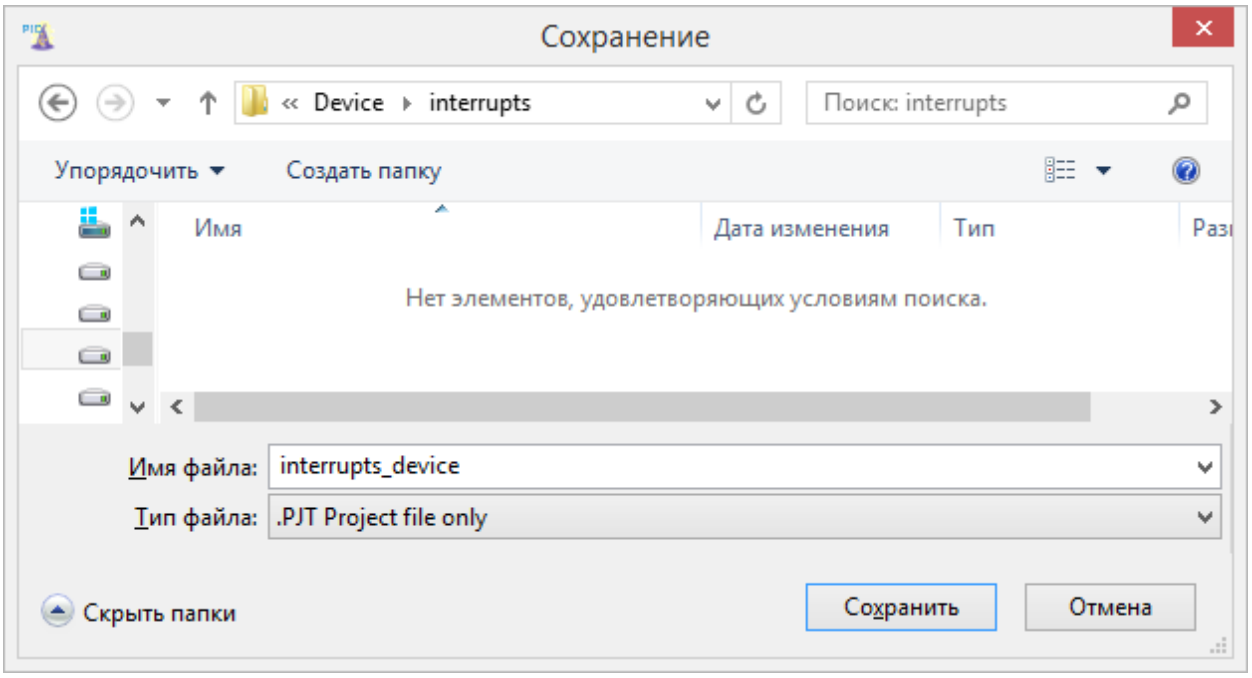

Рисунок 3.2.6 – Процес створення проекту у **IDE PCWH** за допомогою майстра **PIC Wizard**

У розділі **General** (рис. 3.2.7) вибирається цільовий мікроконтролер (список **Device**, що розкривається), його робоча частота (поле **Oscillator Frequency**), а також настроюються загальні параметри, на зразок розрядів запобігання, типу джерела системної синхронізації, порогової напруги для скидання та ін.

Для перегляду програмного коду, який буде згенерований і доданий у вихідний файл відповідно до параметрів на поточній вкладці, слід вибрати вкладку **Code**.

4. У розділі **General › Device** вибрати тип мікроконтролера, наприклад **PIC18F252**;

5. У розділі **General › Fuses** вибрати тип генератора **High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)** (рис. 3.2.7):

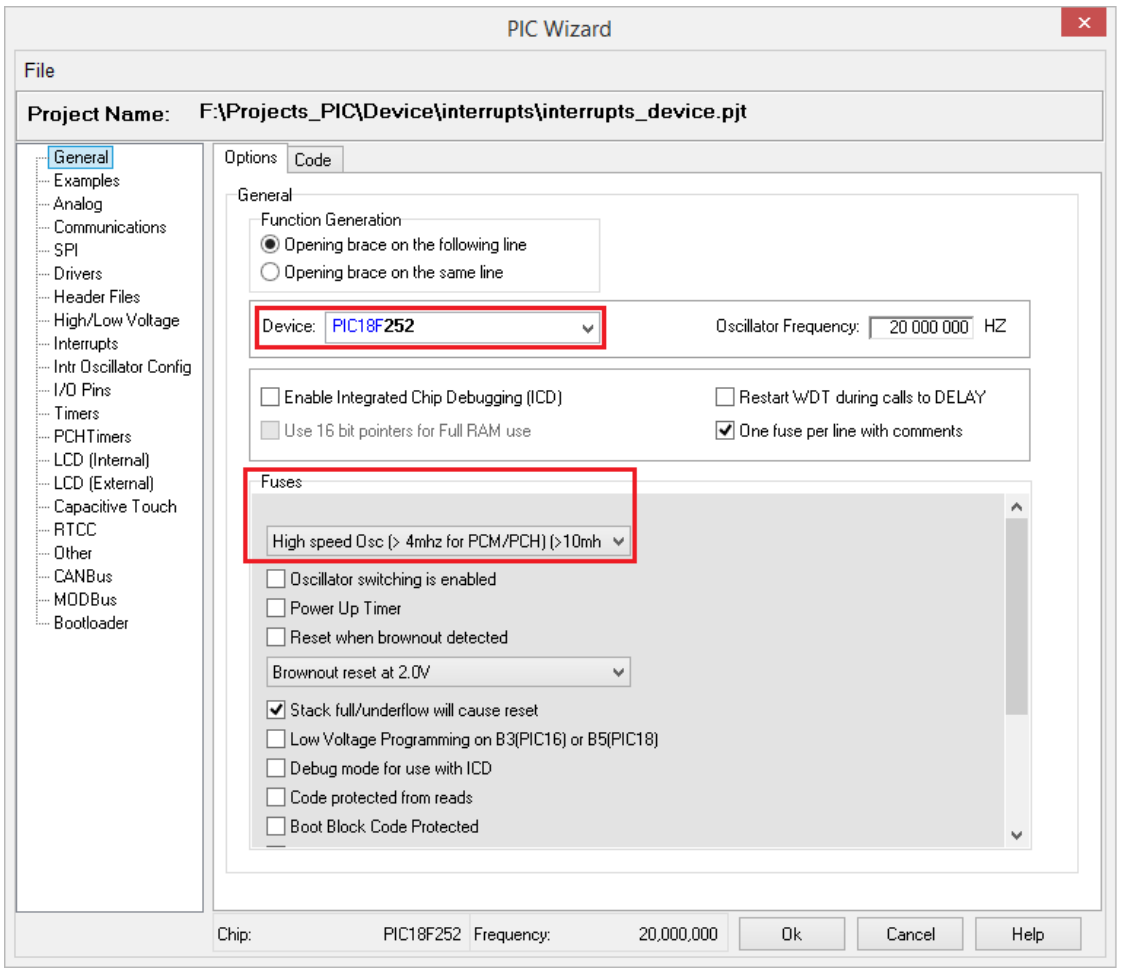

Рисунок 3.2.7 – Розділ **General** майстра **PIC Wizard**

У розділі **Communications** (рис. 3.2.8) налаштовуються параметри введення/виведення для інтерфейсів **RS-232** і **I²C** (швидкість обміну даними, відповідні лінії портів введення/виведення, перевірка помилок, режим **Master/Slave** і т.д.), а також активізується/відключається апаратний порт **PSP**.

6. У розділі **Communications** встановити мітку у полі **RS-232**:

- ☑ **Use RS-232**;

вказати:

- **Transmit:** **PIN C6** (передача);
- **Receive:** **PIN C7** (прийом).

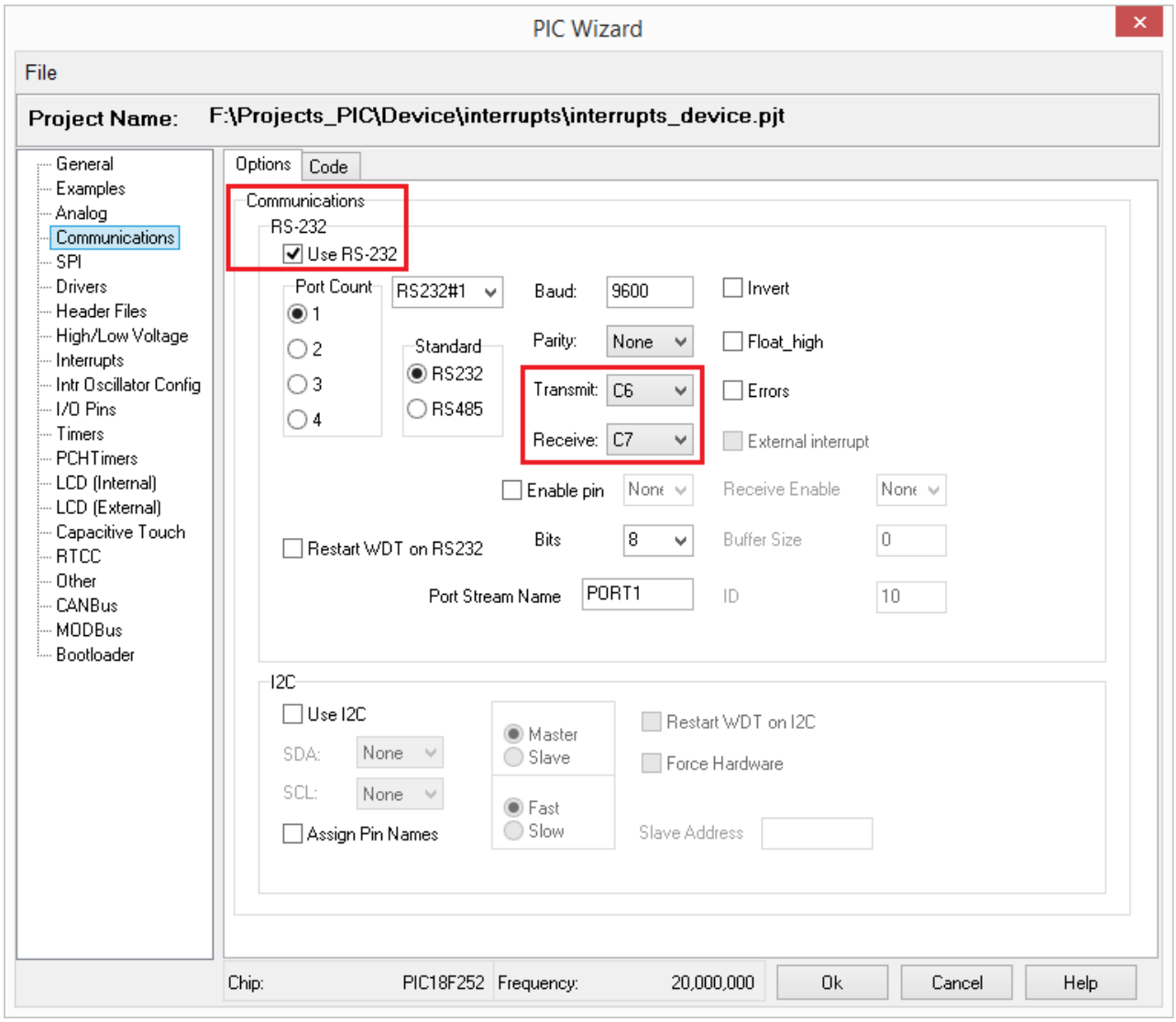

Рисунок 3.2.8 – Розділ **Communications** майстра **PIC Wizard**

Встановлення переривання мікроконтролера з IDE PCWH

Розділ **Interrupts** (рис. 3.2.9) дозволяє за допомогою набору прапорців дозволити або заборонити те чи інше переривання, доступне для вибраного пристрою.

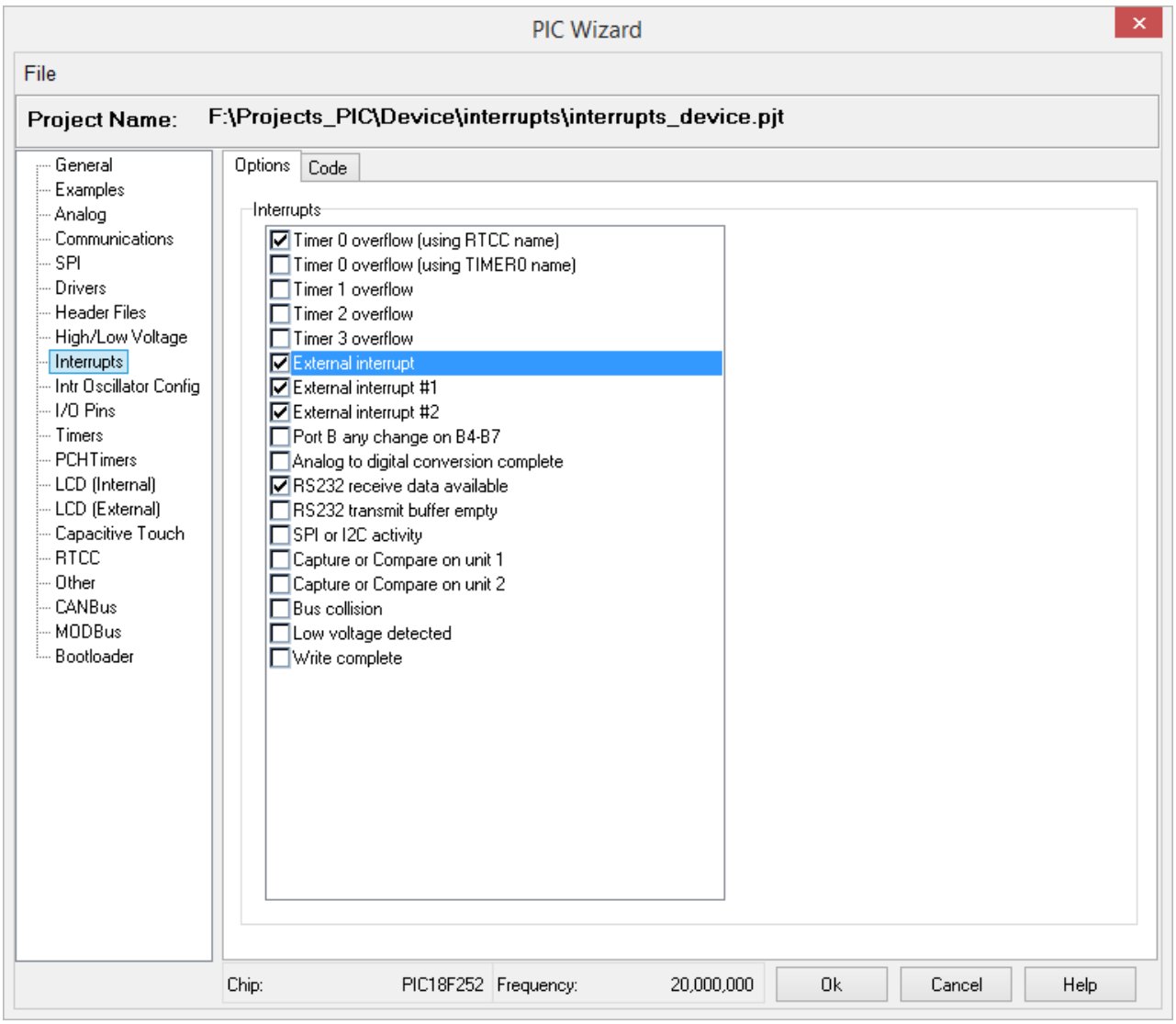

Рисунок 3.2.9 – Розділ **Interrupts** майстра **PIC Wizard**.

Для кожного обраного переривання у головну процедуру програми `main()` додається виклик відповідної підпрограми обробки переривання `enable_interrupts();`, а тіло самої підпрограми розміщується вище `main()` (лістінг 3.2.1).

Для роботи з перериваннями необхідно у інтегрованому середовищі розробки **PCWH** вказати компілятору, які переривання будуть використані у програмі (рис. 3.2.9).

7. У розділі **Interrupts › Options** вибрати переривання:
    1) Timer 0 overflow (using RTCC name);
    2) External interrupt;
    3) External interrupt 1;
    4) External interrupt 2;
    5) RS-232 receive data available.

Встановлення переривань мікроконтролера з **IDE PCWH** представлене на рис. 3.2.9.

Після встановлення переривань мікроконтролера у розділі **Interrupts › Options** буде згенерований і доданий програмний код у вихідний файл відповідно до параметрів на поточній вкладці.

Для перегляду програмного коду, слід вибрати вкладку **Code** (рис. 3.2.10).

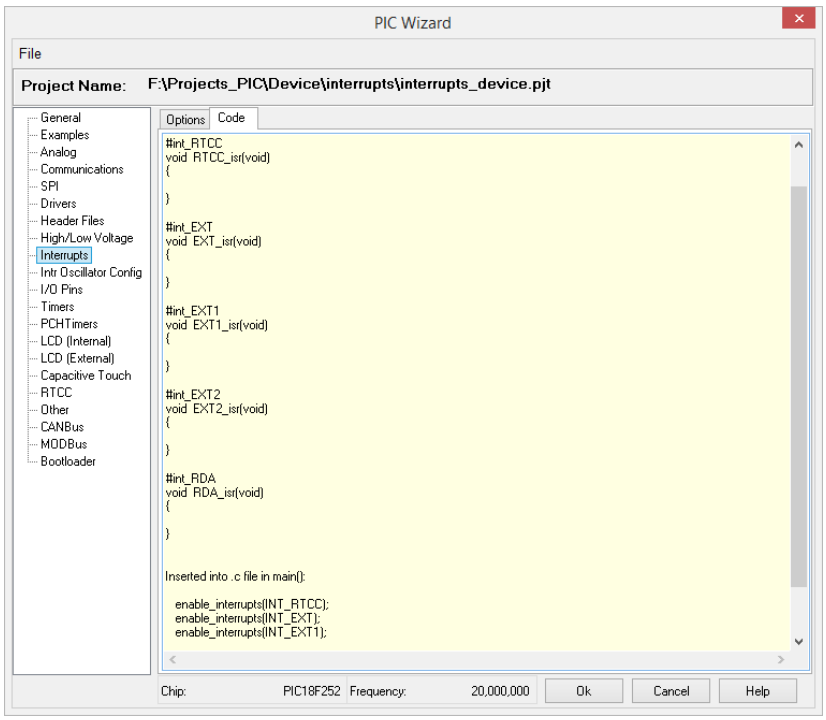

Рисунок 3.2.10 – Попередній перегляд коду, що генерується майстром **PIC Wizard** для розділу **Interrupts**

Встановлення таймера RTCC з IDE PCWH

8. У розділі **Timers** (рис. 3.2.11) налаштовуються параметри таймерів, включаючи сторожовий таймер.

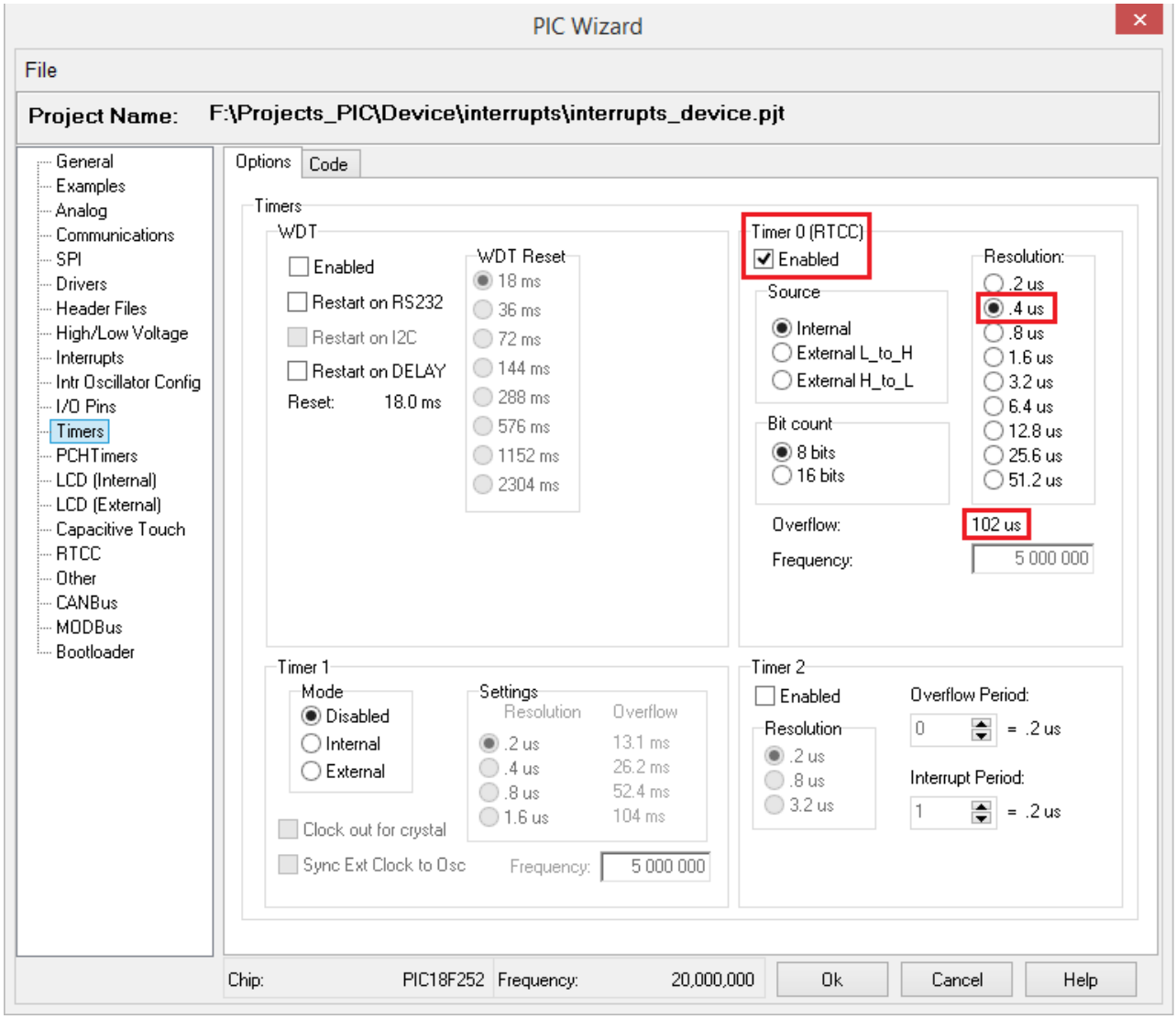

Рисунок 3.2.11 – Розділ **Timers** майстра **PIC Wizard**.

Значення деяких параметрів:

- **WDT Reset** – період між сигналами скидання від сторожового таймера;
- **Source** – вибір джерела тактування таймера TMR0: внутрішній або зовнішній (по наростаючому чи спадаючому фронті сигналу);
- **Frequency** – встановлення частоти у разі, коли вибране зовнішнє тактування таймера TMR0;

- **Resolution** – дозвіл таймера, що впливає на період між переповненнями лічільного регістру (для таймерів TMR0 і TMR1 обчислюється автоматично і відображається в полі **Overflow**);
- **Interrupt Period** – період між запитами на переривання від таймера TMR2.

Якщо обраний мікроконтролер надає додаткові таймери, їх параметри налаштовуються у розділі **PCH Timers** майстра **PIC Wizard**.

Встановлення переривань таймера **RTCC** з **IDE PCWH** представлене на рис. 3.2.11.

9. Натиснути кнопку **OK**.

Буде згенерований наступний код (рис. 3.2.12):

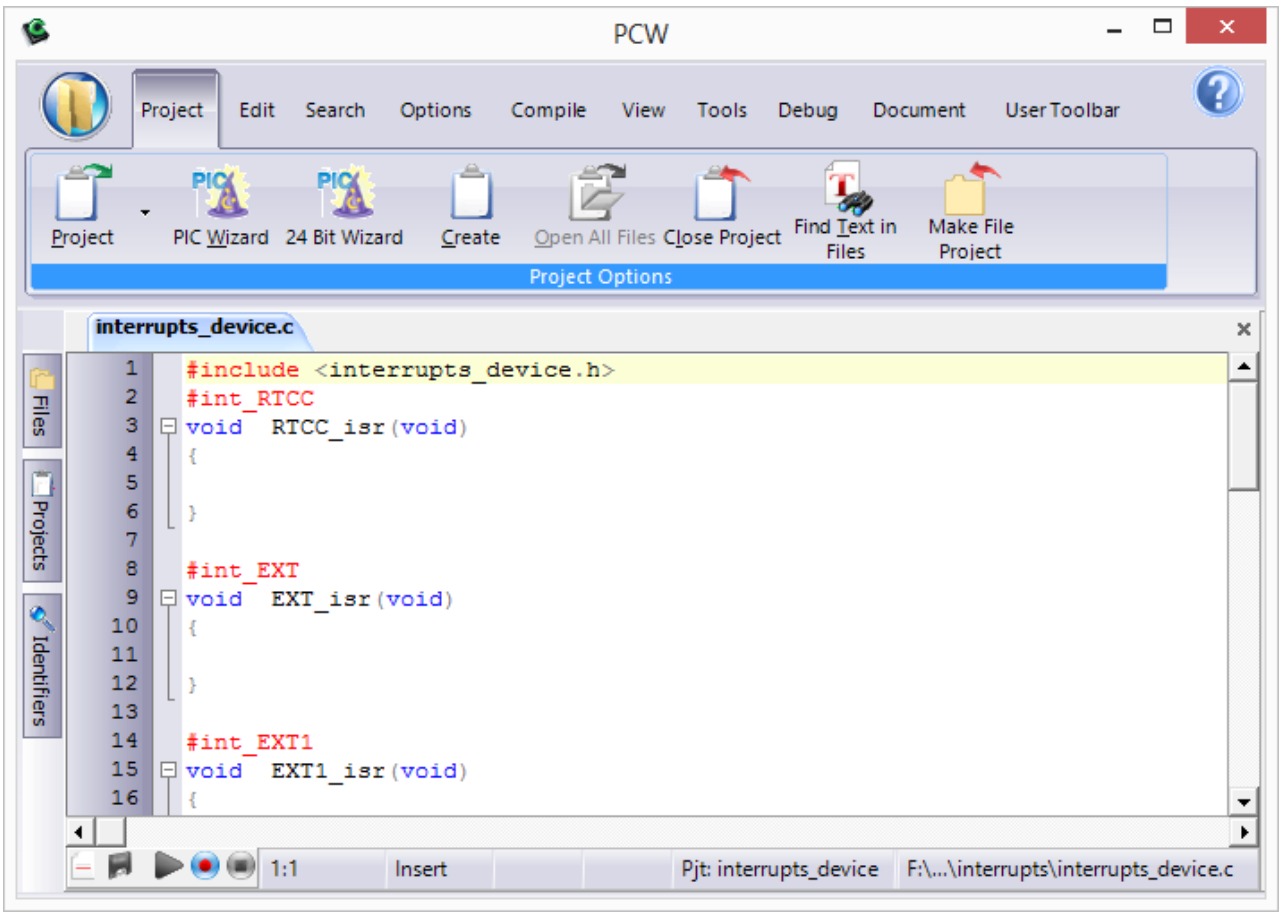

Рисунок 3.2.12 – Згенерований майстром **PIC Wizard** програмний код

Лістинг 3.2.1 – Приклад програмного коду, згенерованого майстром PIC Wizard

```
#include <interrupts_device.h>
#int_RTCC
void RTCC_isr(void)
{

}
#int_EXT
void EXT_isr(void)
{

}
#int_EXT1
void EXT1_isr(void)
{

}


#int_EXT2
void EXT2_isr(void)
{

}

#int_RDA
void RDA_isr(void)
{

}
void main()
{
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_2|RTCC_8_bit); //102 us overflow
setup_timer_3(T3_DISABLED | T3_DIV_BY_1);
enable_interrupts(INT_RTCC);
enable_interrupts(INT_EXT);
enable_interrupts(INT_EXT1);
enable_interrupts(INT_EXT2);
enable_interrupts(INT_RDA);
enable_interrupts(GLOBAL);
while(TRUE)
{
//TODO: User Code
}
}
```

Проект готовий, можна приступати до написання програми.

Створення проекту у інтегрованому середовищі розробки PCWH за допомогою команди меню Project/Create

Новий проект можна створити вручну за допомогою команди меню **Project/Create** (рис. 3.2.12).

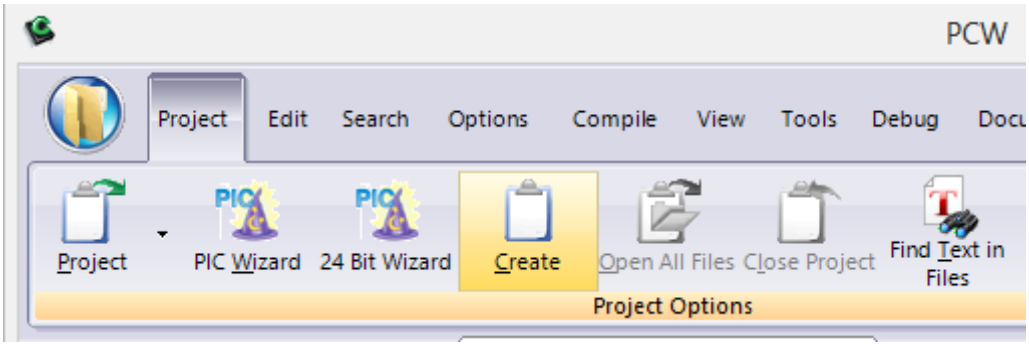

Рисунок 3.2.12 – Створення проекту у ручному режимі
за допомогою команди меню **Project / Create**

Якщо новий проект створений у ручному режимі, для роботи з перериванням таймера **RTCC** необхідно здійснити його ініціалізацію:

```
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_8);
```

Заборонити використання інших таймерів можна директивами:

```
setup_timer_1(T1_DISABLED);
setup_timer_2(T2_DISABLED,0,1);
setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
```

Для роботи з перериваннями їх необхідно дозволити:

```
enable_interrupts(INT_RTCC);        //дозволити переривання таймера RTCC
enable_interrupts(INT_EXT);         //дозволити зовнішнє переривання 0
enable_interrupts(INT_EXT1);        //дозволити зовнішнє переривання 1
enable_interrupts(INT_EXT2);        //дозволити зовнішнє переривання 2
enable_interrupts(INT_RDA);         //дозволити переривання порту RS-232
```

Далі необхідно дозволити всі зазначені переривання:

```
enable_interrupts(GLOBAL);          //дозволити всі зазначені переривання
```

Можна у довільному порядку заборонити переривання:

```
disable_interrupts(INT_RTCC);       //заборонити переривання таймера RTCC
disable_interrupts(INT_EXT);        //заборонити зовнішнє переривання 0
disable_interrupts(INT_EXT1);       //заборонити зовнішнє переривання 1
disable_interrupts(INT_EXT2);       //заборонити зовнішнє переривання 2
disable_interrupts(INT_RDA);        //заборонити переривання порту RS-232
disable_interrupts(GLOBAL);         //заборонити всі зазначені
//переривання
```

Примітка: При роботі із зовнішніми перериваннями порту **B** мікроконтролера необхідно виводи порту **B** підключити до джерела живлення через резистори, тим самим забезпечивши наявність логічної "1" на виводах порту:

```
port_b_pullups(TRUE);          //входи порту B через резистори підключити
//до джерела 5 В
```

Розташування виводів зовнішніх переривань мікроконтролера представлене на рис. 3.2.13.

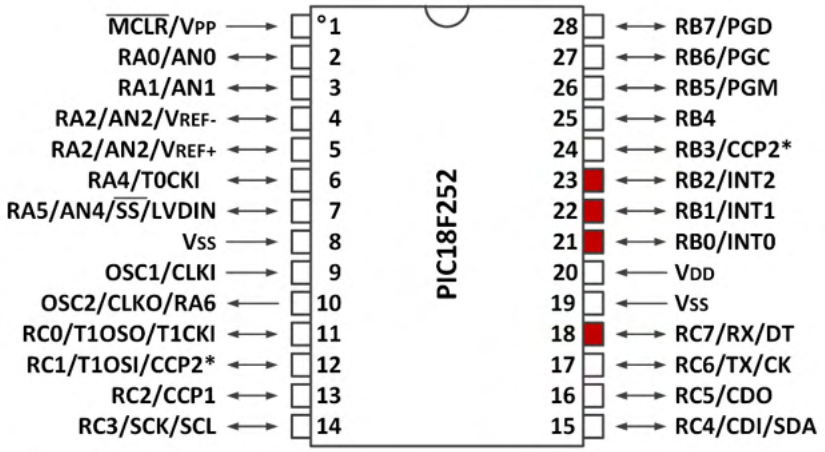

Рисунок 3.2.13 – Розташування виводів зовнішніх переривань мікроконтролера

Доступні переривання мікроконтролера **PIC18F252**:

- натиснути кнопку у меню **View › Valid Interrupts** (рис. 3.2.14 – 3.2.15):

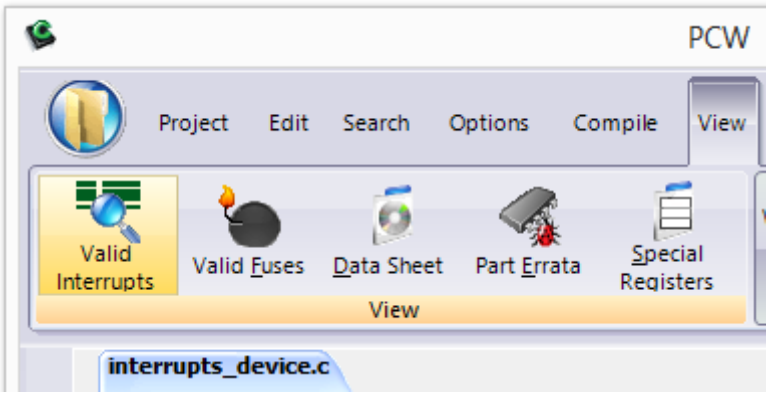

Рисунок 3.2.14 – Доступні переривання мікроконтролера **PIC18F252**

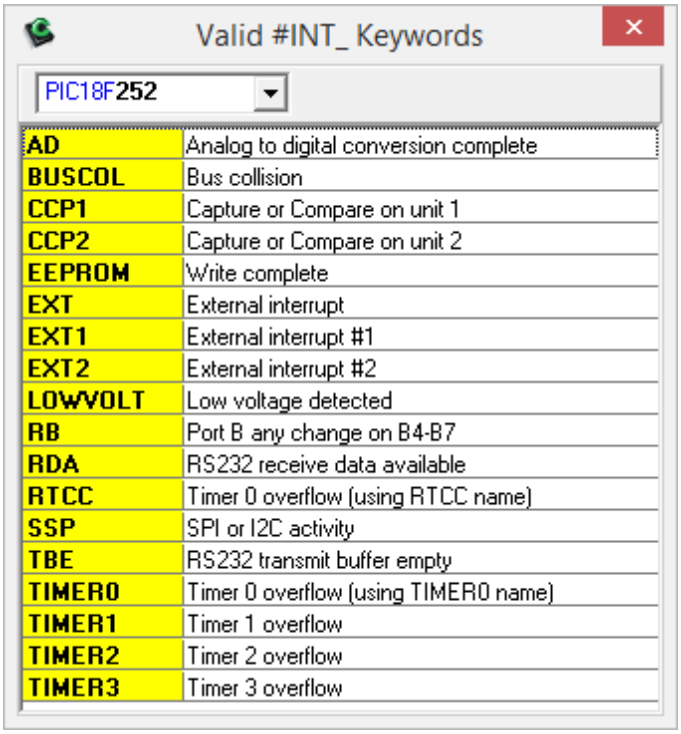

Рисунок 3.2.15 – Доступні переривання мікроконтролера **PIC18F252**

ПОЯСНЕННЯ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ «INTERRUPTS»

Програмне забезпечення для передачі даних за допомогою послідовного інтерфейсу RS-232

Іноді потрібно для обміну даними підключати мікроконтролер до персонального комп'ютера (ПК). У такому випадку на персональному комп'ютері програмується графічний інтерфейс, за допомогою якого керують зовнішнім пристроєм, а підключають апаратні засоби до персонального комп'ютера за допомогою послідовного інтерфейсу.

Щоб передавати дані у мікроконтролер або приймати їх від нього за допомогою послідовного інтерфейсу RS-232, потрібне спеціальне програмне забезпечення на персональному комп'ютері, яке може відображати ці дані.

Після запуску програми потрібно вказувати кілька установок для нового підключення.

Всі зроблені установки можна змінювати також знову в більш пізній момент у відповідному меню.

Програма – термінал Terminal 1.9b

Програма – термінал **Terminal 1.9b** (рис. 3.2.16) є монітором COM порту персонального комп'ютера.

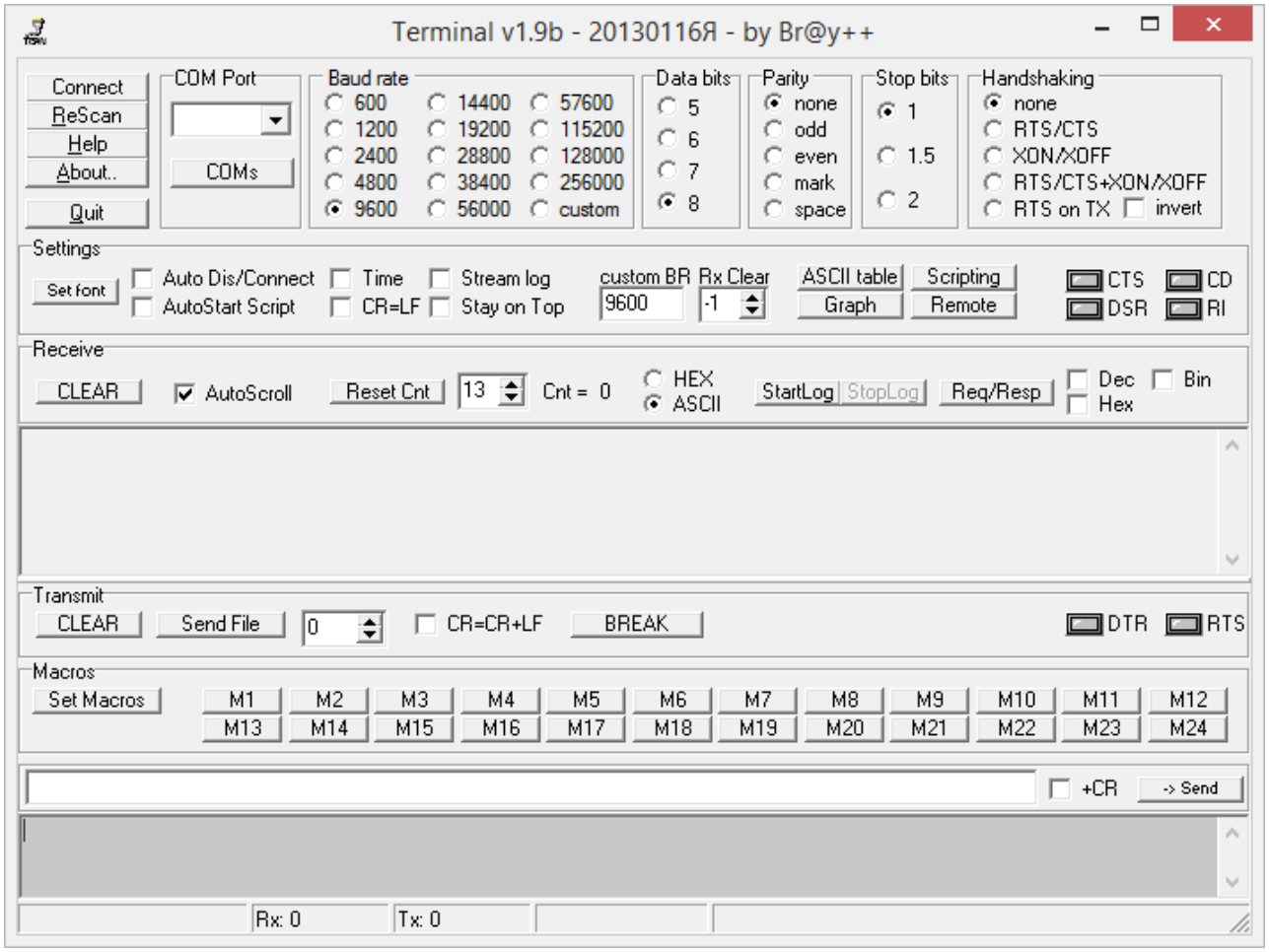

Рисунок 3.2.16 – Програма – термінал **Terminal 1.9b**

За допомогою програми можна легко відправляти і приймати дані через COM порт комп'ютера по протоколу RS-232. Серед переваг Terminal – гнучке налаштування програми під різні режими роботи. Інтерфейс програми простий і зрозумілий).

Основні можливості Terminal 1.9b:

- працює без установки. Вся програма – один exe-файл розміром близько 300Кб;
- є лічильник переданих і прийнятих байтів;
- можливість відправляти файли;
- крім стандартних швидкостей (baudrate) є можливість встановити свою нестандартну швидкість;
- підтримує до 64 COM-портів;
- можна весь лог роботи записувати у файл;
- можна призначити до 24 макросів;
- реалізовані Pascale-подібні скрипти.

Для роботи із програмою необхідно в першу чергу підключити пристрій, з яким збираєтеся працювати до комп'ютера через COM-порт. Підключіть живлення. Тепер запустіть програму - термінал **Terminal v1.9b**.

Інтерфейс і основні настройки підключення по порту

У верхньому полі знаходяться параметри підключення (рис. 3.2.17):

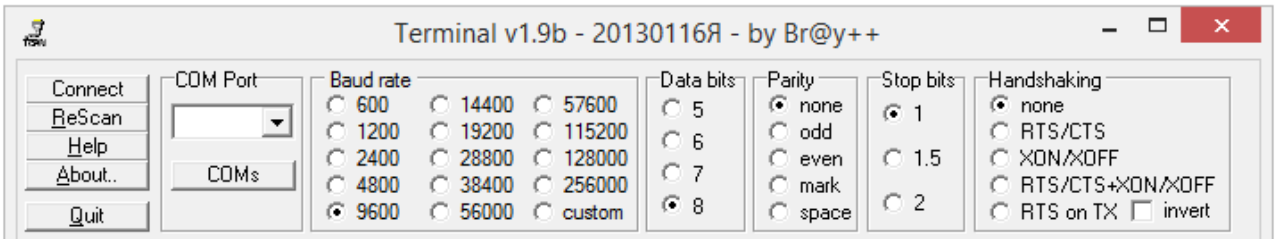

Рисунок 3.2.17 – Параметри підключення програми - терміналу **Terminal 1.9b**

Кнопка **Connect** – кнопка для відкриття COM-порту.

Кнопка **Rescan** – пересканувати список COM-портів.

Кнопка **Help** – довідка.

Кнопка **About ..** – про програму.

Кнопка **Quit** – вихід з програми.

Розділ **COM Port** – поле вибору номера COM-порту для підключення.

Розділ **Baud rate** – вибір швидкості COM-порту.

Розділ **Data bits** – вибір кількості біт даних.

Розділ **Parity** – вибір парності.

Розділ **Stop bits** – вибір кількості стопових біт.

Розділ **Handshaking** – вибір типу управління потоком.

У розділі **Settings** знаходяться додаткові параметри. Вони стануть в нагоді для написання скриптів, роботи з нестандартними швидкостями або для запису логу від пристрою.

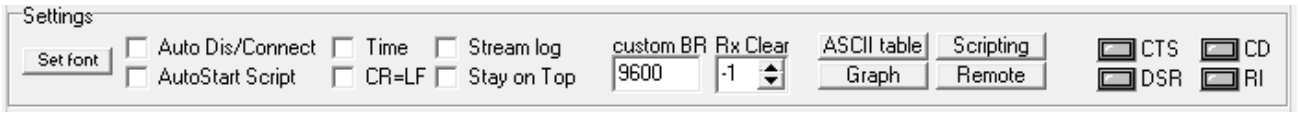

Рисунок 3.2.18 – Розділ **Setting**s

У розділі **Recieve** знаходяться параметри відображення відповіді від пристрою.

Рисунок 3.2.19 – Розділ **Recieve**

У розділі **Transmit** знаходяться параметри передачі даних на пристрій. Кнопки **DTR** і **RTS** встановлюють відповідні виводи у позитивний стан.

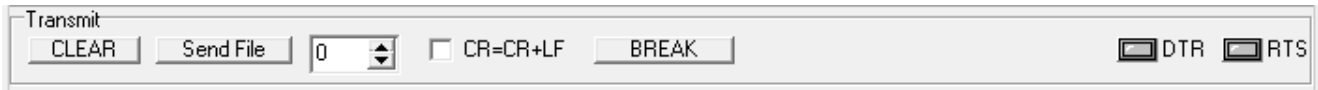

Рисунок 3.2.20 – Розділ **Transmit**

Макроси

Поле **Macros** призначене для створення користувацьких швидких клавіш.

Рисунок 3.2.21 – Поле **Macros**

Для цього потрібно натиснути на кнопку **SetMacros** і у вікні, присвоїти кожній кнопці визначену послідовність символів, яка буде відправлятися на пристрій.

Рисунок 3.2.22 – Вікно **Macro Settings**

Блок №1:

Поле для введення послідовності символів для відправки.

Для того щоб відправити спеціальні символи, необхідно скористатися ASCII таблицею і ввести код символу, попередньо екранувати його знаком "$".

Блок №2:

У лівому полі задається ім'я кнопки, а в правому відображається сама кнопка.

Блок №3:

Задається затримка при автоматичному повторенні команд.

Блок №4:

Включення автоматичного повтору команди через інтервал часу.

Кнопки **Load** і **Save** дозволяють зберегти або завантажити файл з макросами, введеними в цьому вікні.

Відправлення та прийом даних

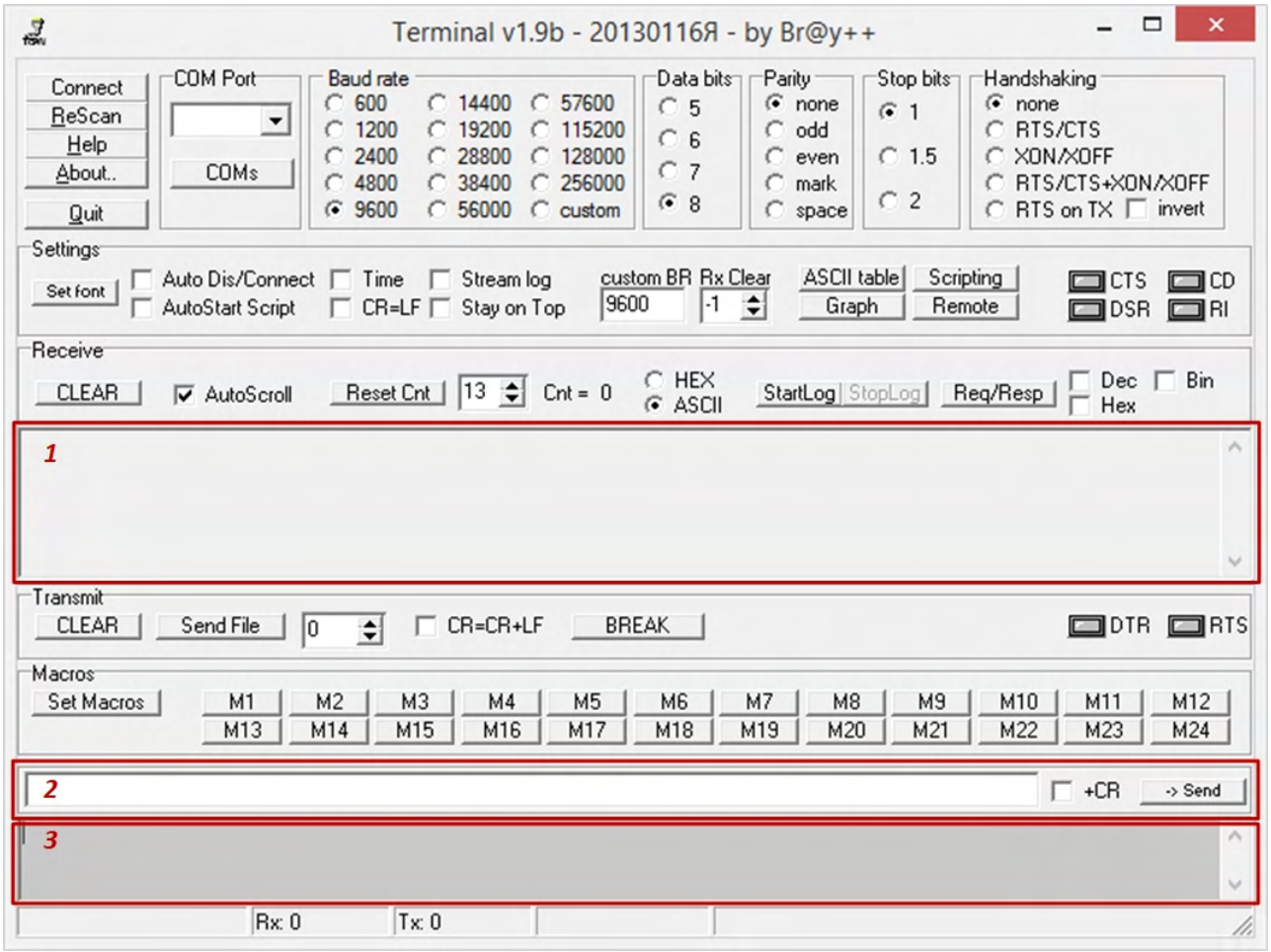

Рисунок 3.2.23 – Відправлення та прийом даних

Блок №1:

У цьому великому полі ви будете бачити відповіді від вашого пристрою.

Якщо у розділі «**Settings**» поставити галочку біля пункту «**Time**», то перед кожним рядком буде проставлена мітка часу. Це буває дуже корисно при аналізі логів з пристрою.

Блок №2:

Тут знаходиться поле для відправки тексту повідомлень.

Встановлена галочка біля пункту « **+ CR** » буде дописувати до повідомленню, яке відправляється символ повернення каретки (емулювати натискання клавіші **Enter**).

Кнопка « → **Send** » відправить ваше повідомлення на пристрій.

Блок №3:

У самому низу знаходиться поле, де можна бачити відправлені на пристрій команди.

Програма – термінал SIOW.exe

Програму – термінал **SIOW.exe** можна запустити у директорії **PICC › SIOW.exe**. Вікно Програми - терміналу **SIOW.exe** представлено на рис. 3.2.24:

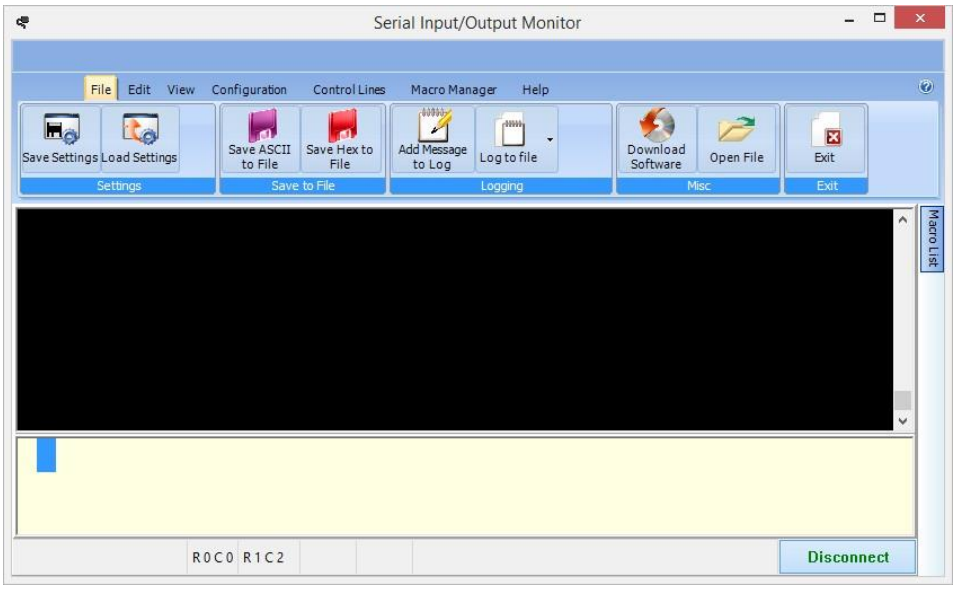

Рисунок 3.2.24 – Програма - термінал **SIOW.exe**

Для роботи із програмою необхідно вибрати порт і встановити параметри порту (рис. 3.2.25):

у меню **Configuration › Set Port Options**

вибрати: **ComPorts:** Direct to COM1

Baud rates: 9600

Натиснути кнопку **OK**

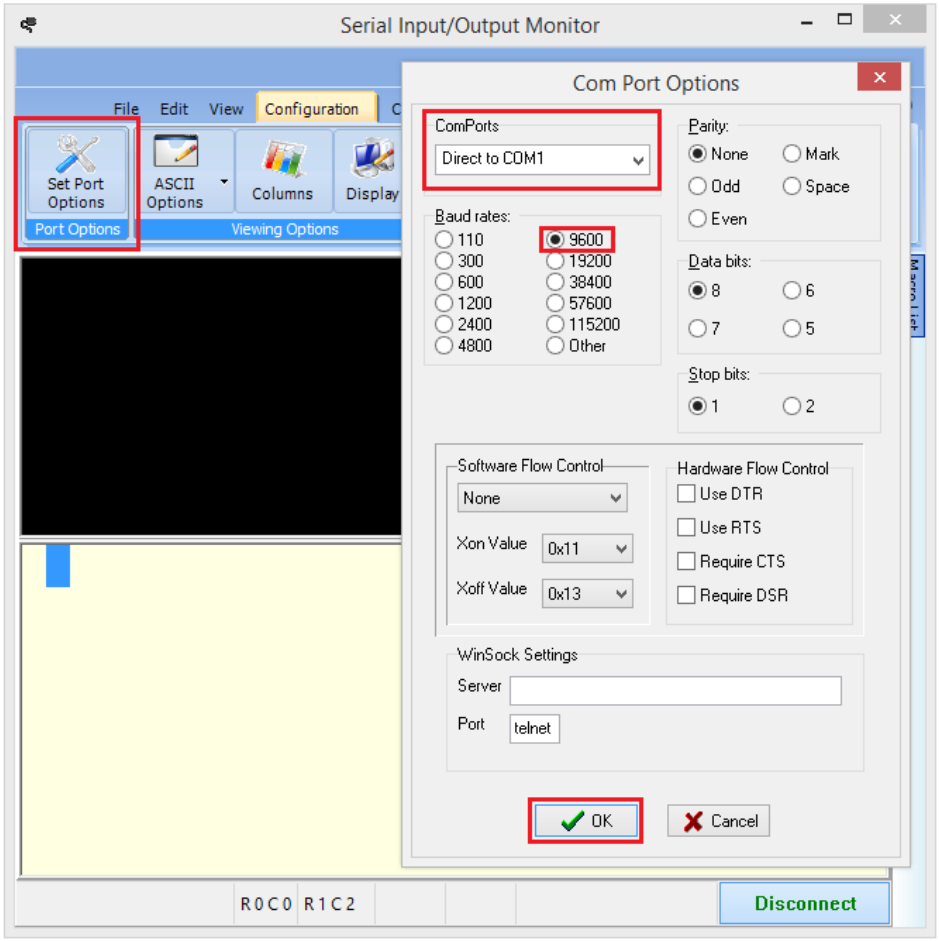

Рисунок 3.2.25 – Програма - термінал **SIOW.exe**

Потім у меню **File › Download Software** вибрати файл з розширенням ***.HEX**.

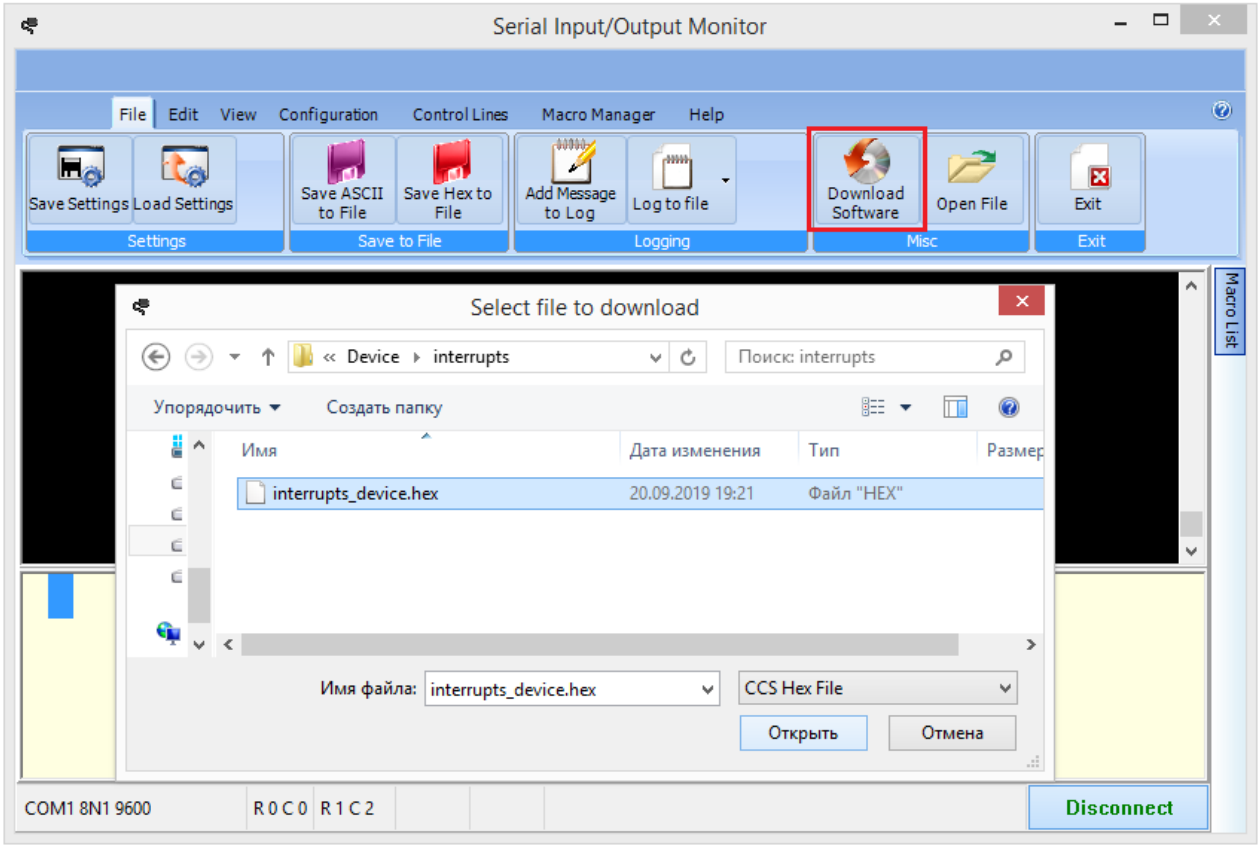

Рисунок 3.2.26 – Програма - термінал **SIOW.exe**

Примітка: у інтегрованому середовищі розробки **PCWH** в обробнику переривання порту **RS-232** необхідно очистити прийомний буфер, що б уникнути повторних переривань.

Очищення буфера здійснюється шляхом його читання функціями:

```
getc();
gets();
get_string();
```

Приклад виконання лабораторної роботи «INTERRUPTS» відповідно до варіанту завдання для лабораторної роботи «INTERRUPTS»

Варіант завдання для виконання лабораторної роботи **«INTERRUPTS»** представлений у таблиці 3.2.3.

Таблиця 3.2.3 – Варіант завдання для виконання лабораторної роботи **«INTERRUPTS».**

Переривання				Виведення (світлодіоди)		
RS-232	INT0	INT1	INT2	B5	B6	B7
+	+	+	+	+	+	+

Принципова схема підключень для варіанту АПК

Принципова схема підключень для варіанту **АПК** відповідно варіанту завдання (таб. 3.2.3), для виконання лабораторної роботи **«INTERRUPTS»** виглядає у такий спосіб (рис. 3.2.27):

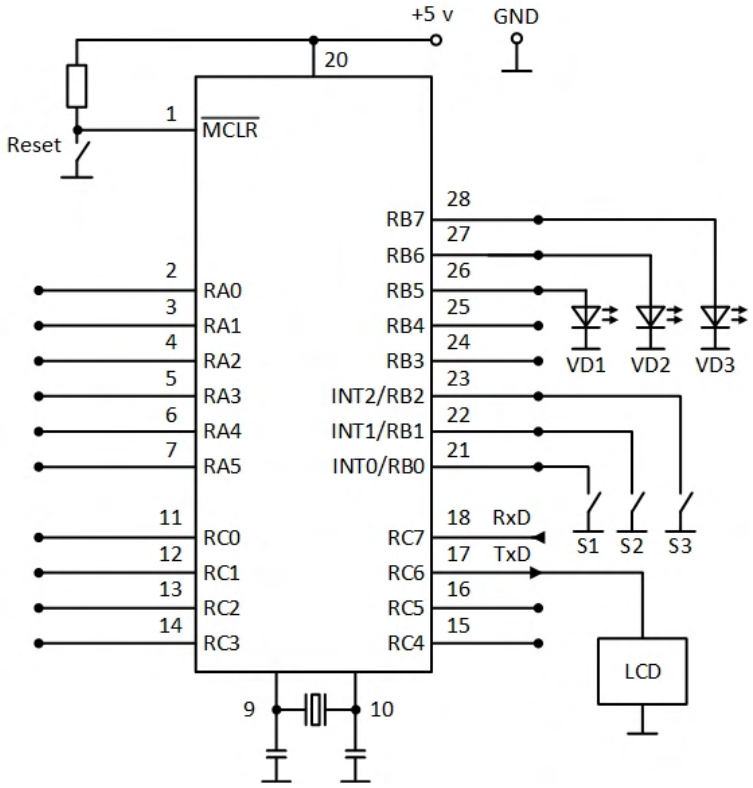

Рисунок 3.2.27 – Принципова схема підключень для лабораторної роботи **«INTERRUPTS»** для варіанту **АПК**

Принципова схема підключень для варіанту «Proteus»

Принципова схема підключень для варіанту **«Proteus»**, для виконання лабораторної роботи **«INTERRUPTS»** виглядає у такий спосіб (рис. 3.2.28):

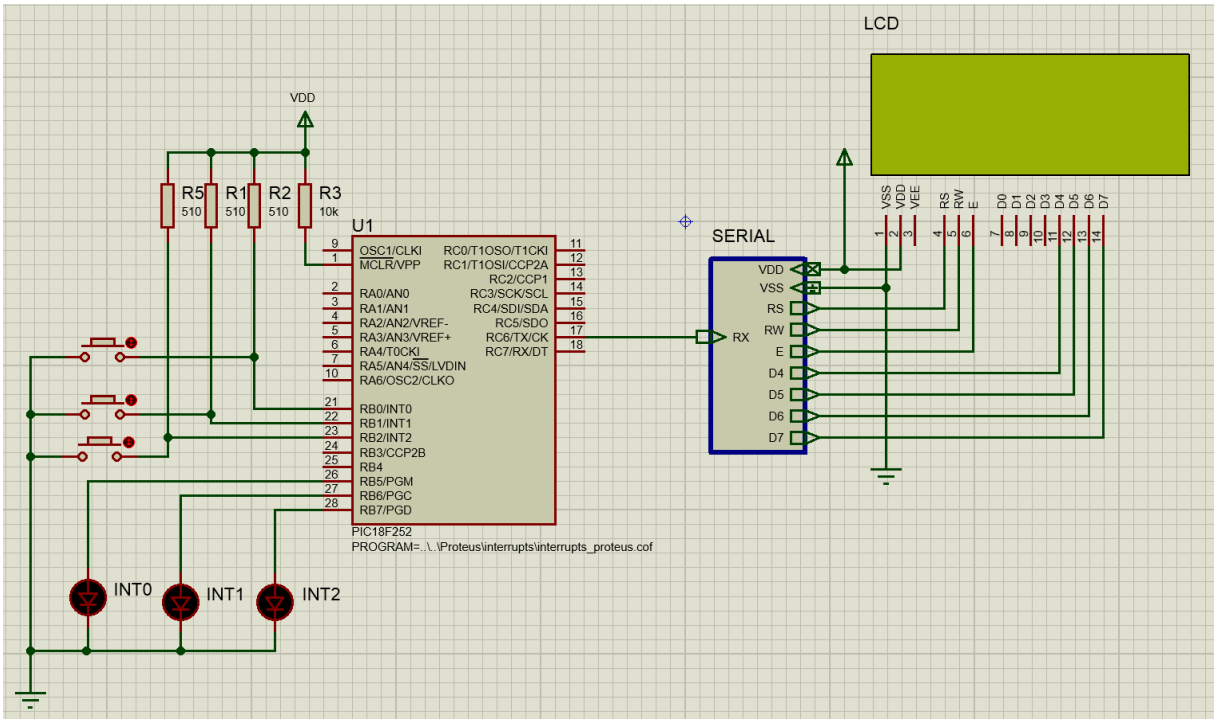

Рисунок 3.2.28 – Принципова схема підключень для лабораторної роботи **«INTERRUPTS»** для варіанту **«Proteus»**

Результат виконання лабораторної роботи «INTERRUPTS» для варіанту «Proteus»

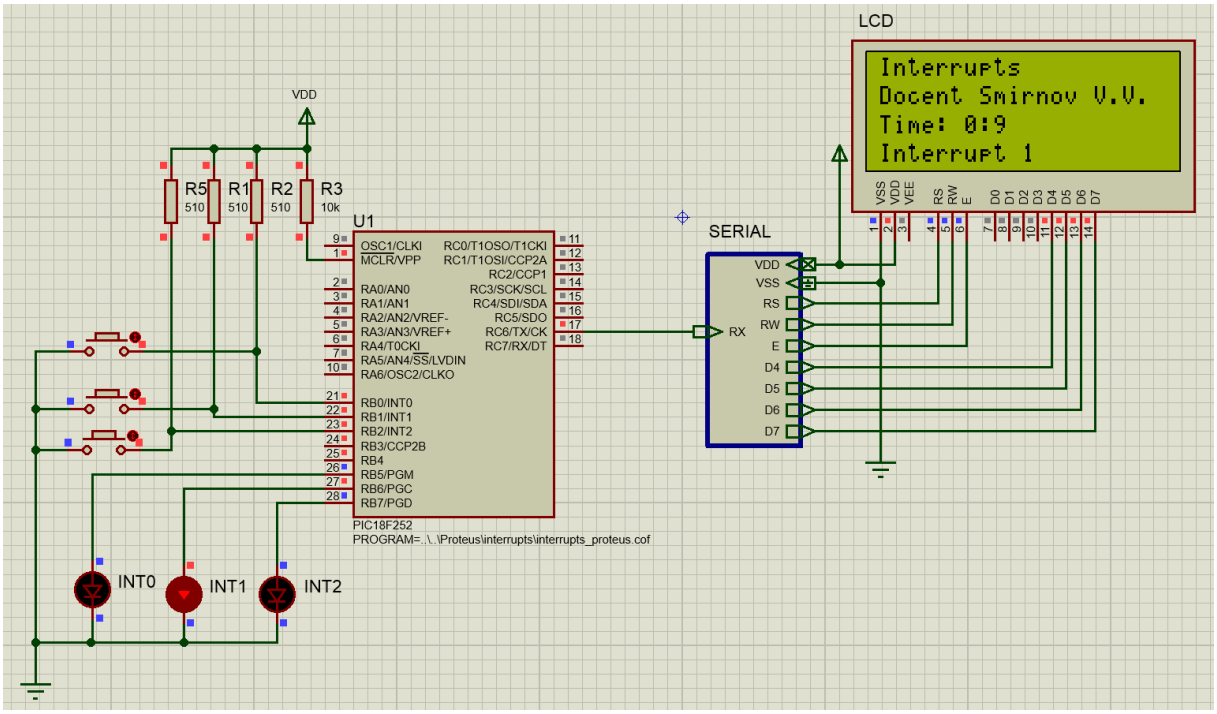

Рисунок 3.2.29 – Результат виконання лабораторної роботи **«INTERRUPTS»** для варіанту **«Proteus»**

Контрольні питання

1) які умови і причини виникнення внутрішніх і зовнішніх переривань мікроконтролера **PIC18F252** (типи переривань)?
2) як управляти (дозволяти, забороняти) перериваннями у мікроконтролері?
3) чому у перериванні порту RS-232 необхідно очистити прийомний буфер?
4) якими по відношенню до мікроконтролера бувають переривання?
5) навіщо потрібні переривання у системах керування?

ЛАБОРАТОРНА РОБОТА № 3 - «ADC»

Тема: Робота з АЦП

Ціль роботи:

Отримання навичок роботи з **АЦП** мікроконтролера і написання програм обробки оцифрованих аналогових сигналів.

Завдання лабораторної роботи

- для варіанту **АПК** намалювати принципову схему підключень відповідно до варіанту для виконання лабораторної роботи (таб. 3.3.1);
- для варіанту **«Proteus»** використовувати готову схему відповідно до варіанту для виконання лабораторної роботи;
- створити алгоритм програми роботи з АЦП для перетворення рівня напруги з потенціометра і клавіатури;
- здійснити зчитування рівня напруги з потенціометра, оцифрувати за допомогою АЦП і перетворити в десятковий формат відповідно до варіанту для виконання лабораторної роботи;
- відобразити рівень поточної напруги у вольтах на LCD. Показання напруги на LCD повинно відрізнятися від показань вольтметра не більше, ніж на 0,01 В;
- написати програму визначення натиснутої кнопки на клавіатурі;
- здійснити виведення результатів на LCD відповідно до варіанту для виконання лабораторної роботи.

Виведення на LCD повинне містити:

1) зчитані дані з АЦП:
- у HEX форматі;
- у десятковому форматі;
- у вольтах;

2) символ натиснутої клавіші;
3) прізвище та ініціали студента.

Варіанти для виконання лабораторної роботи «ADC»

Таблиця 3.3.1 – Варіанти для виконання лабораторної роботи **«ADC»**

	Канал АЦП								**Розрядність АЦП**	
	Потенціометр				**Клавіатура**					
№	**A0**	**A1**	**A2**	**A3**	**A0**	**A1**	**A2**	**A3**	**8**	**10**
1	+					+			+	
2		+					+			+
3			+					+	+	
4				+	+					+
5				+		+			+	
6			+			+				+
7	+							+	+	
8	+						+			+
9		+						+	+	
10			+		+					+
11				+	+				+	
12	+					+				+
13		+			+				+	
14			+					+		+
15				+			+		+	

Послідовність виконання лабораторної роботи для варіанту АПК

1) для варіанту **АПК** намалювати принципову схему підключень відповідно до варіанту для виконання лабораторної роботи;
2) створити свій директорій для файлів проекту;
3) відкрити інтегроване середовище розробки **PCWH**;
4) створити проект у інтегрованому середовищі розробки **PCWH**;
5) створити алгоритм програми роботи з АЦП для перетворення рівня напруги з потенціометра і клавіатури;
6) написати програму мовою програмування **C**, що реалізує створений алгоритм, відповідно до варіанту для виконання лабораторної роботи;

7) скомпілювати програму, одержати бінарні файли з розширеннями ***.HEX** і ***.COF** (файли з розширеннями ***.HEX** і ***.COF** створюються підчас компіляції програми); завантажити бінарний файл з розширенням ***.HEX** у пам'ять програм мікроконтролера програмою **PICkit.exe**;
8) на **АПК** зкумутувати схему підключень (підключити потенціометр, клавіатуру і вольтметр до АЦП відповідно до рис. 3.3.1, рис. 3.3.2) відповідно до варіанту для виконання лабораторної роботи.

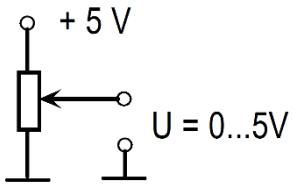

Рисунок 3.3.1 – Принципова схема потенціометра

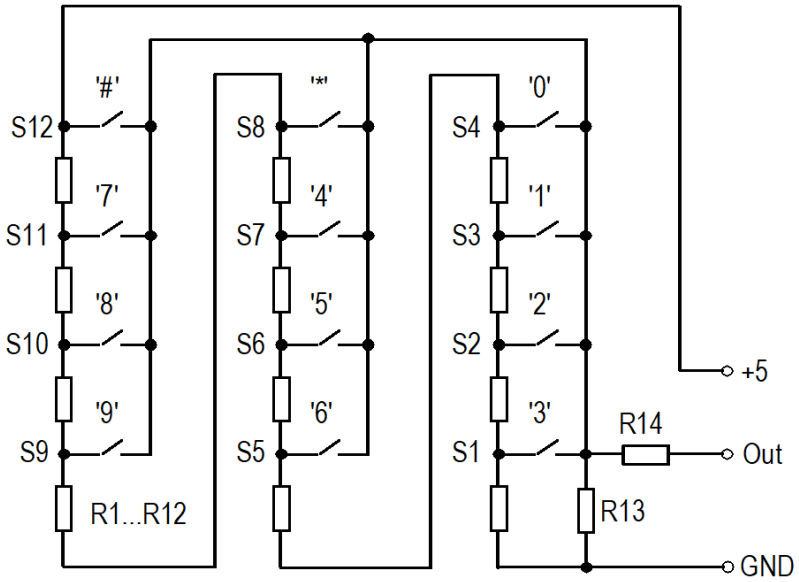

Рисунок 3.3.2 – Принципова схема клавіатури

9) виконати програму.

Послідовність виконання лабораторної роботи для варіанту «Proteus»

1) для середовища моделювання **«Proteus»** використовувати готову схему;
2) створити свій директорій для файлів проекту;
3) відкрити інтегроване середовище розробки **PCWH**;
4) створити проект у інтегрованому середовищі розробки **PCWH**;
5) створити алгоритм програми;
6) написати програму мовою програмування **C**, що реалізує створений алгоритм, відповідно до варіанту для виконання лабораторної роботи;
7) скомпілювати програму, одержати бінарні файли з розширеннями ***.HEX** і ***.COF** (файли з розширеннями ***.HEX** і ***.COF** створюються підчас компіляції програми);
8) відкрити середовище моделювання **«Proteus»**;
9) у папці **«Proteus_students»** вибрати папку лабораторної роботи **«ADC»**;
10) відкрити файл проекту з розширенням ***.DSN**;
11) у середовищі моделювання **«Proteus»** змінити схему підключень відповідно до варіанту для виконання лабораторної роботи;
12) завантажити заздалегідь скомпільований бінарний файл з розширенням ***.COF** або ***.HEX** у мікроконтролер;
13) у середовищі моделювання **«Proteus»** виконати програму.

Послідовність виконанн.я лабораторної роботи для варіанту «Proteus» з використанням шаблону програми

1) для середовища моделювання **«Proteus»** використовувати готову схему;
2) відкрити папку **«Proteus_students»**;
3) відкрити інтегроване середовище розробки **PCWH**;

4) у папці **«Proteus_students»** вибрати папку лабораторної роботи **«ADC»**;
5) відкрити файл проекту з розширенням ***.pjt**;
6) у редакторі **IDE PCWH** відкрити шаблон файлу програми з розширенням ***.c**;
7) відкрити середовище моделювання **«Proteus»**;
8) у папці **«Proteus_students»** вибрати папку лабораторної роботи **«ADC»**;
9) відкрити файл проекту з розширенням ***.DSN**;
10) у середовищі моделювання **«Proteus»** змінити схему підключень відповідно до варіанту для виконання лабораторної роботи;
11) створити алгоритм програми;
12) у інтегрованому середовищі розробки **PCWH**, використовуючи шаблон програми самостійно написати програму мовою програмування **C**, що реалізує створений алгоритм, відповідно до варіанту для виконання лабораторної роботи,
13) скомпілювати програму, одержати бінарні файли з розширеннями ***.HEX** і ***.COF** (файли з розширеннями ***.HEX** і ***.COF** створюються підчас компіляції програми);
14) у середовищі моделювання **«Proteus»** виконати програму.

Примітка: *файл демонстрації виконання* лабораторної роботи ***«ADC»*** *розташований на сайті* http://pksm.kntu.kr.ua/SCME.html.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Аналого-цифровий перетворювач (АЦП)

Аналого-цифровий перетворювач (АЦП) являє собою пристрій для перетворення безупинно мінливих у часі аналогових сигналів в еквівалентні значення числових кодів. АЦП характеризується набором параметрів. АЦП виконаний або у вигляді модуля МК, або у вигляді окремого чіпу.

Характеристики АЦП

Характеристика перетворення АЦП - це залежність між напругою на його аналоговому вході і безліччю можливих значень вихідного коду, задана у вигляді таблиці, графіка або формули.

Розрізняють *номінальну* характеристику перетворення (рис. 3.3.3), встановлену у стандартах або технічних умовах на АЦП конкретного типу, і *дійсну* характеристику перетворення, знайдену експериментальним шляхом і настільки наближену до *істинної* характеристики перетворення конкретного АЦП, що для даної мети може бути використана замість неї.

Відхилення характеристики перетворення АЦП від прямої, що з'єднує точки *(0; 0)* і *(V_{REF}; 111 ... 1)*, характеризує *похибку квантування*.

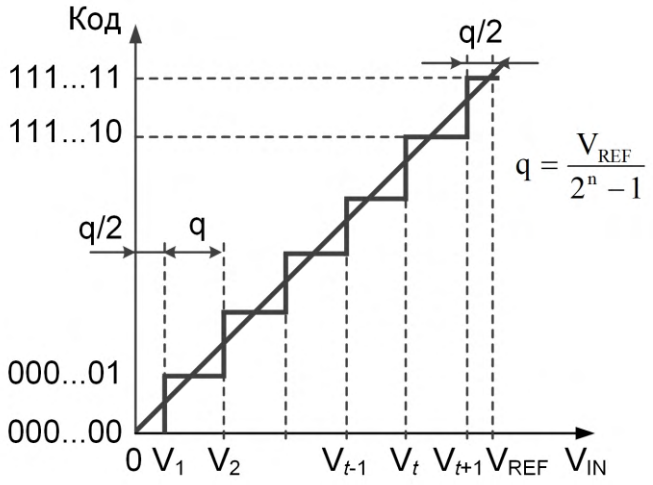

Рисунок 3.3.3 - Характеристика перетворення АЦП

Як видно з рис. 3.3.3, для *ідеального* АЦП, що має номінальну характеристику перетворення, максимальна величина цього відхилення становить *q/2* і залежить тільки від кількості розрядів АЦП.

Розрядність. Це основний параметр АЦП, він визначає максимальне двійкове число на виході.

Роздільна здатність (resolution) - це величина, зворотна максимальному числу кодових комбінацій на виході АЦП.

Роздільна здатність виражається у відсотках, розрядах або децибелах і характеризує потенційні можливості АЦП з точки зору досяжної точності.

Наприклад, 12-розрядний АЦП має роздільну здатність 1/4096, або 0.0245% від повної шкали, або -72.2 дБ.

У табл. 3.3.2 наведений ряд числових співвідношень між кількістю розрядів, роздільною здатністю, а також залежність значення молодшого значущого розряду (МЗР) від діапазону вхідної напруги і т.п.

Таблиця 3.3.2 - Ряд числових співвідношень між кількістю розрядів і іншими параметрами.

Розряди (n)		Дозвіл	dB	1/n	%	ppm	Значення молодшого значущого розряду (МЗП)				
							20 В	10 В	5 В	2.5 В	1 В
2	2^{-2}	1/4	-12	0.25	25	250000	5 В	2.5 В	1.25 В	625 мВ	250мВ
4	2^{-4}	1/16	-24.1	0.625	6.2	62500	1.25 В	625 мВ	312 мВ	156 мВ	62.5 мВ
6	2^{-6}	1/64	-36.1	0.015625	1.6	15625	312 мВ	156 мВ	78.1 мВ	39.1 мВ	15.6 мВ
8	2^{-8}	1/256	-48.2	0.003906	0.4	3906	78.1 мВ	39.1 мВ	19.5 мВ	9.77 мВ	3.91 мВ
10	2^{-10}	1/1024	-60.2	0.0009766	0.1	977	19.5 мВ	9.77 мВ	4.86мВ	244 мВ	977 мВ

Головний перехід. Під цим розуміють зміну коду з 011 ... 111 на 100 ... 000 або навпаки.

Частота Найквіста. Це частота, рівна 1/2 частоти дискретизації.

Відношення «сигнал/шум» *(Signal to Noise Ratio, SNR)* - це відношення ефективного значення вхідного сигналу до середньоквадратичного значення «шуму». «Шум» складається з усіх інших спектральних компонент, включаючи

гармоніки, але виключаючи постійну складову. Рівень вхідного сигналу береться рівним (~ 1 дБ) від повної шкали.

Для ідеального АЦП визначається за формулою:

$$SNR = (6.02\ n + 1.76)\ дБ,$$

де n - кількість двійкових розрядів.

Таким чином, для ідеального 12-розрядного АЦП отримуємо:

$$SNR = 74\ дБ.$$

Інтермодуляційні викривлення *(IMD).* Коли на вхід подаються два гармонійних коливання з різними частотами (f_a і f_b), то на виході будь-якого пристрою з нелінійностямі будуть присутні викривлення порядку (m + n) на сумарних і різницевих частотах:

$$mf_a \pm n\,f_b,$$

де:

m, n = 0, 1, 2, 3 ...

Інтермодуляційними викривленнями називаються ті, для яких ні m, ні n не рівні нулю.

Інтегральна нелінійність *(Integral Non-Linearity, INL) (похибка нелінійності)* - це різниця між розрахунковим значенням вхідної напруги V_t' визначеним по лінеаризованій характеристиці перетворення АЦП, і дійсним значенням вхідної напруги V_t, відповідним заданій точці характеристики перетворення (рис. 3.3.4):

$$INL = V_t' - V_t$$

Диференціальна нелінійність *(Differential Non-Linearity, DNL)* - це різниця між значенням кванта перетворення (q_t) і середнім дійсним значенням кванта перетворення (q):

$$DNL = q_t - q = V_t - V_{t-1} - q$$

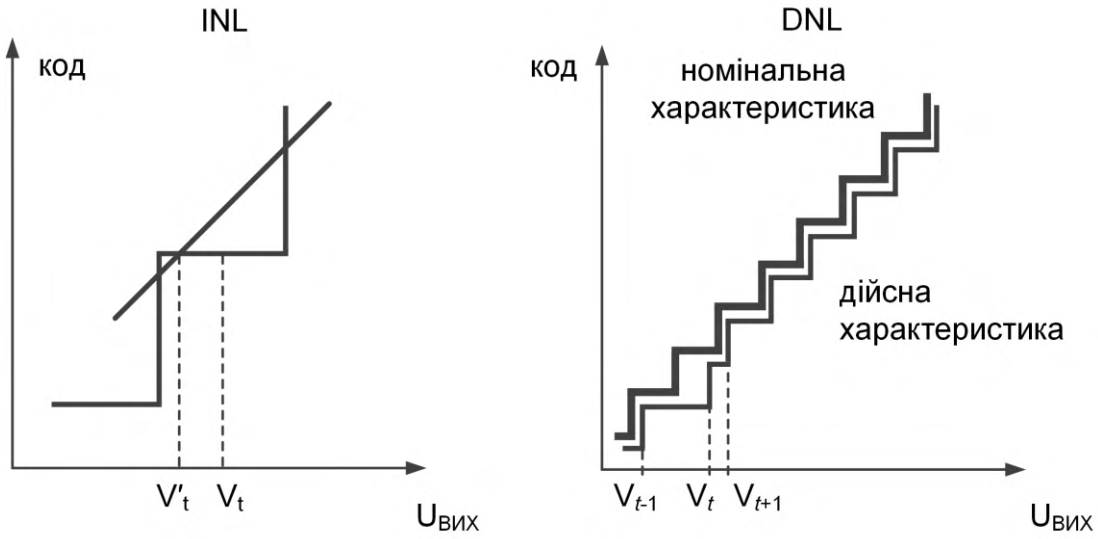

Рисунок 3.3.4 - Інтегральна та диференціальна нелінійність

Похибка нуля. Перемикання коду з головним переходом має відбуватися при рівні на аналоговому вході - 0,5 МЗР щодо аналогової землі AGND.

Похибка нуля визначається як відхилення фактичного рівня цього перемикання від наведеного значення.

Похибка нуля вказується при певній температурі, а температурний дрейф визначає її максимальну зміну у зазначеному температурному діапазоні.

Похибка біполярного нуля - це відхилення рівня, відповідного переключення коду у середині шкали, від ідеального значення (AGND - 0.5 МЗР).

Час відновлення після перенапруги - це час, який потрібен АЦП для досягнення номінальної точності після перенапруги на аналоговому вході (на 50% більше діапазону повної шкали) і вимірюється до того моменту, коли вхідна напруга повертається в межі вхідного діапазону АЦП.

Робоча смуга частот - це номінальний діапазон частот вхідного сигналу, заданий розробником приладу, в якому при заданій частоті дискретизації нормуються метрологічні характеристики. Вона вужче, ніж смуга пропускання частот повної потужності.

Час встановлення - це час, необхідний АЦП для досягнення номінальної точності після того, як на його вхід (або вхід вбудованого ПВЗ) був поданий ступінчастий сигнал, рівний повному діапазону вхідного сигналу.

Конвеєрна затримка (латентність) - це число тактових періодів між початком перетворення і появою на вихідних лініях відповідного результату. Після цього на кожному тактовому періоді виводиться новий результат перетворення.

Вхідний діапазон. Ця специфікація визначає мінімальні та максимальні вхідні напруги від нуля до повної шкали, які АЦП може сприймати і при цьому точно калібрувати посилення.

Класифікація АЦП

Всі АЦП можна розділити на дві групи, що істотно розрізняються між собою по нормованим характеристикам похибок і методів перевірки.

До першої групи належать АЦП, виконані у вигляді мікросхем (напівпровідникових, гібридних), які не є засобами вимірювань.

АЦП другої групи є засобами вимірювань.

АЦП першої групи широко використовуються не тільки для створення АЦП другої групи, але і в якості вузлів різних систем обробки аналогових сигналів.

До складу АЦП часто входять допоміжні вузли, що істотно поліпшують метрологічні характеристики і розширюють функціональні можливості АЦП:

- буферні підсилювачі;
- автоматичні перемикачі діапазонів;
- програмовані підсилювачі;
- пристрої вибірки-зберігання;
- схеми автокалібрування і автопідстроювання;
- екстраполятор;
- оперативні і постійні запам'ятовуючі пристрої;
- цифрові фільтри і т.п.

Практично всі АЦП орієнтовані на спільну роботу з мікропроцесорними системами і містять елементи інтерфейсу (буферні регістри, дешифратори адреси і т.п.).

Структурна схема модуля АЦП мікроконтролера серії PIC18FXX2 представлена на рис. 3.3.5.

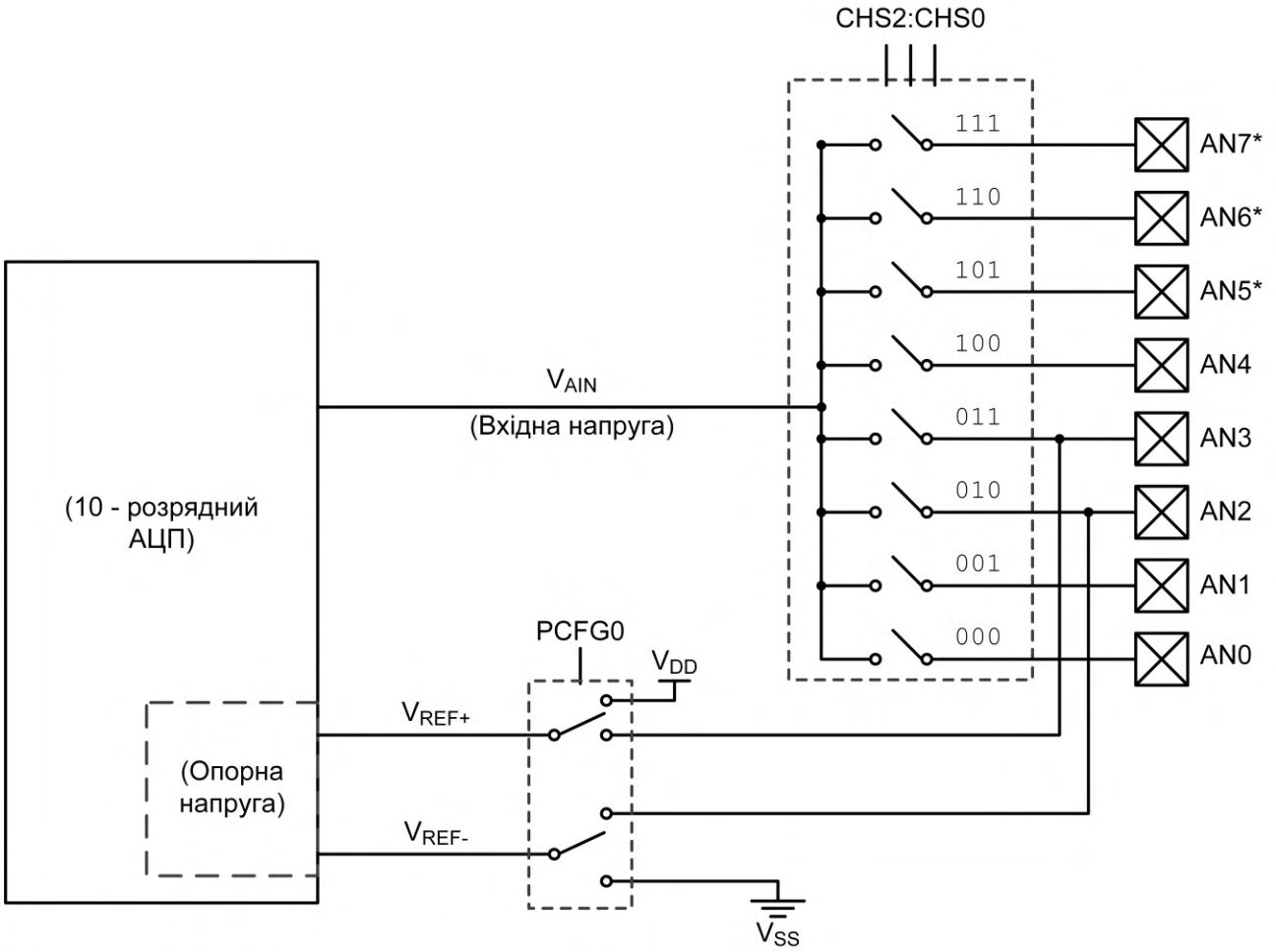

Рисунок 3.3.5 - Структурна схема модуля АЦП МК серії **PIC18FXX2**

ПРИКЛАД СТВОРЕННЯ ПРОЕКТУ У ІНТЕГРОВАНОМУ СЕРЕДОВИЩІ РОЗРОБКИ PCWH

Створення проекту у інтегрованому середовищі розробки PCWH за допомогою команди меню Project/Create

Для виконання лабораторної роботи **«ADC»**, новий проект можна створити вручну за допомогою команди меню **Project/Create** (рис. 3.3.6):

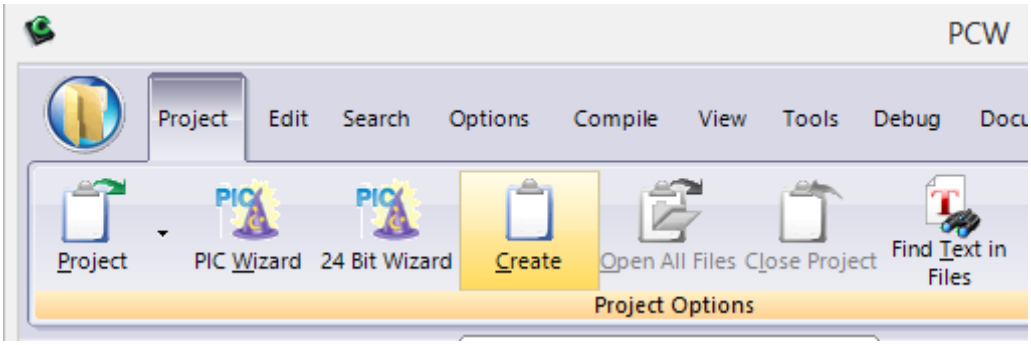

Рисунок 3.3.6 – Створення проекту у ручному режимі за допомогою команди меню **Project/Create**

Якщо новий проект створений у ручному режимі за допомогою команди меню **Project/Create**, для конфігурування мікроконтролера для робота з АЦП, у заголовному файлі ***.h** необхідно ввести рядок, що вказує розрядність АЦП:

```
#device adc=8 або #device adc=10
```

а у функції ініціалізації ввести рядки:

```
setup_adc_ports(ALL_ANALOG);
setup_adc(ADC_CLOCK_INTERNAL);
```

Також новий проект можна згенерувати автоматично за допомогою майстра **PIC Wizard**, який викликається по команді меню **Project › PIC Wizard**, або використовувати шаблон програми, що знаходиться у папці **«Proteus_students»**. Необхідно вибрати папку відповідної лабораторної роботи **«ADC»**.

Створення проекту у інтегрованому середовищі розробки PCWH за допомогою майстра PIC Wizard

Для запуску майстра **PIC Wizard** слід виконати відповідну команду меню **Project**, а потім вказати розміщення і ім'я головного файлу проекту.

В результаті відкриється вікно майстра, що складається з розділів з параметрами проекту (рис. 3.3.10).

Зовнішній вигляд вікна майстра може відрізнятися в залежності від версії компілятора **CCS-PICC** і обраного типу мікроконтролера.

1. Запустити **IDE PCWH**, натиснути кнопку **Projects** майстра **PIC Wizard** (рис. 3.3.7):

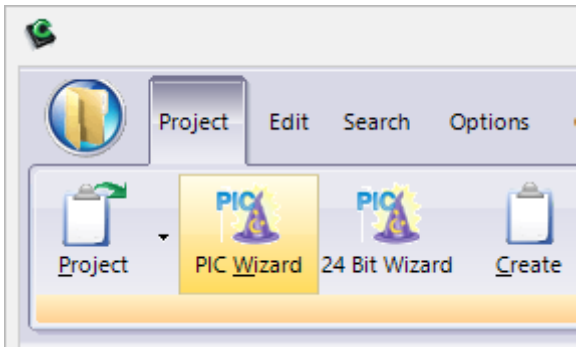

Рисунок 3.3.7 – Процес створення проекту у **IDE PCWH** за допомогою майстра **PIC Wizard**

або натиснути кнопку у вигляді відкритої папки  .

2. Вибрати: **New › Project Wizard** (рис. 3.3.8):

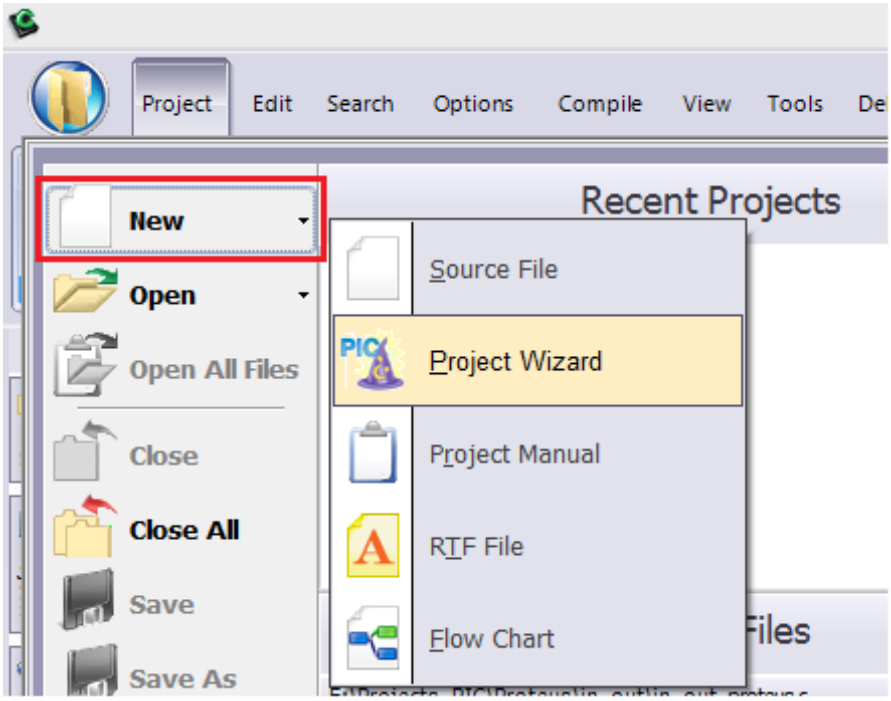

Рисунок 3.3.8 – Процес створення проекту у **IDE PCWH** за допомогою майстра **PIC Wizard**

3. Створити або вибрати свій директорій і записати у нього проект під будь-яким іменем латинським шрифтом, наприклад: «**adc.pjt**» (рис 3.3.9):

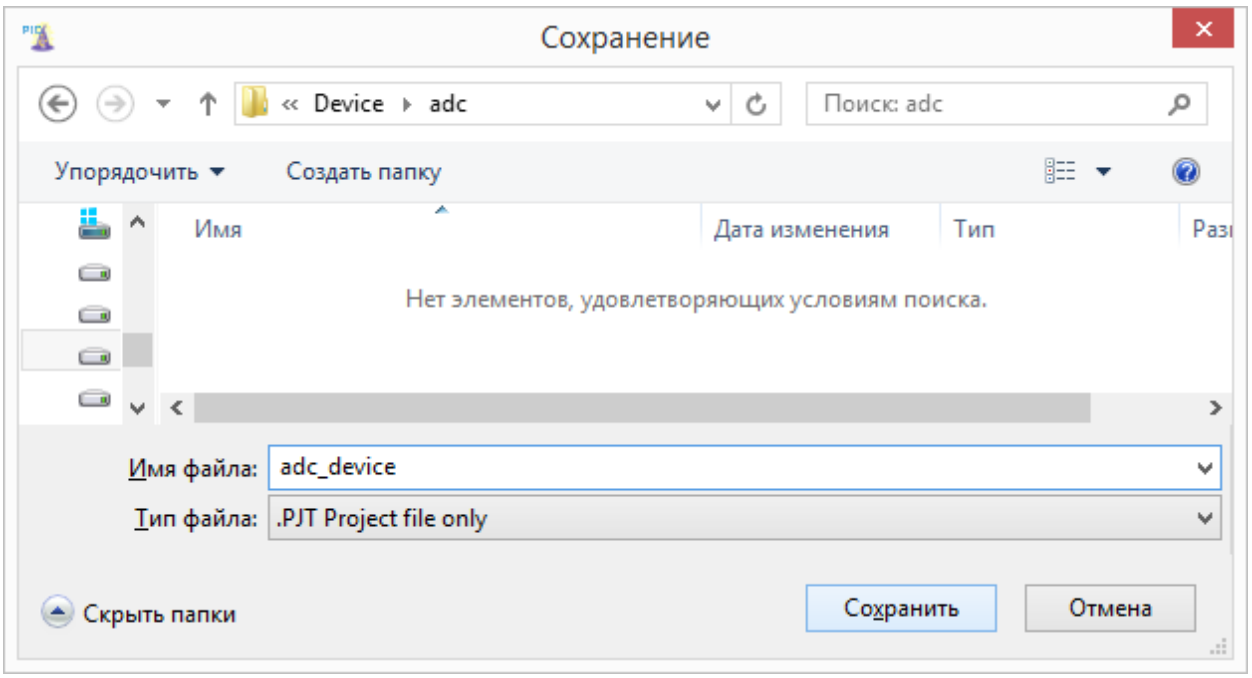

Рисунок 3.3.9 – Процес створення проекту у **IDE PCWH** за допомогою майстра **PIC Wizard**

У розділі **General** (рис 3.3.10) вибирається цільовий мікроконтролер (список **Device**, що розкривається), його робоча частота (поле **Oscillator Frequency**), а також настроюються загальні параметри, на зразок розрядів запобігання, типу джерела системної синхронізації, порогової напруги для скидання та ін.

Для перегляду програмного коду, який буде згенерований і доданий у вихідний файл відповідно до параметрів на поточній вкладці, слід вибрати вкладку **Code**.

4. У розділі **General › Device** (рис. 3.3.10) вибрати тип мікроконтролера, наприклад **PIC18F252**.

5. У розділі **General › Fuses** вибрати тип генератора **High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)**:

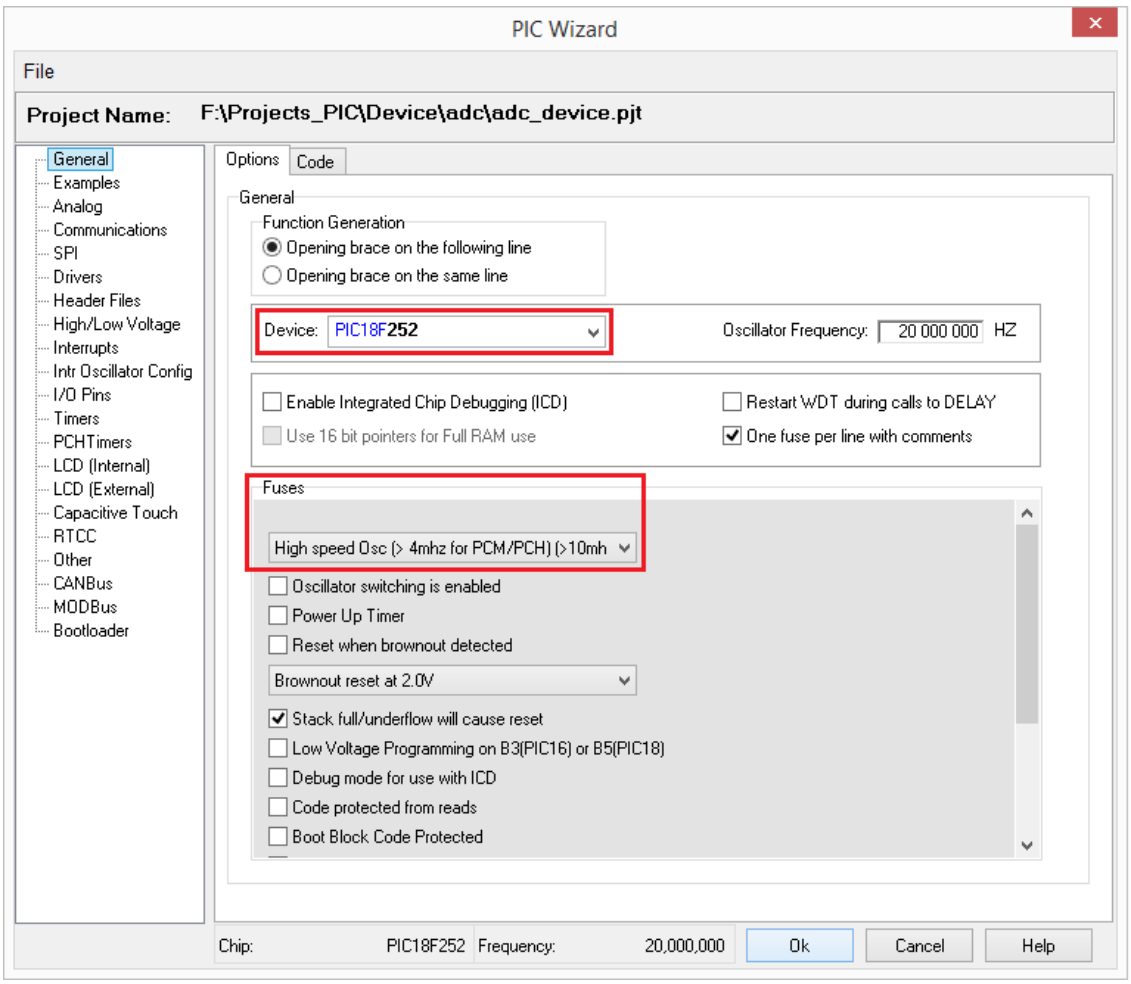

Рисунок 3.3.10 – Розділ **General** майстра **PIC Wizard**

У розділі **Communications** (рис. 3.3.11) налаштовуються параметри введення/виведення для інтерфейсів **RS-232** і **I²C** (швидкість обміну даними, відповідні лінії портів введення/виведення, перевірка помилок, режим **Master/Slave** і т.д.), а також активізується/відключається апаратний порт **PSP**.

6. У розділі **Communications** встановити мітку у полі **RS-232**:

- ☑ **Use RS-232**;

вказати:

- **Transmit:** **PIN C6** (передача);
- **Receive:** **PIN C7** (прийом).

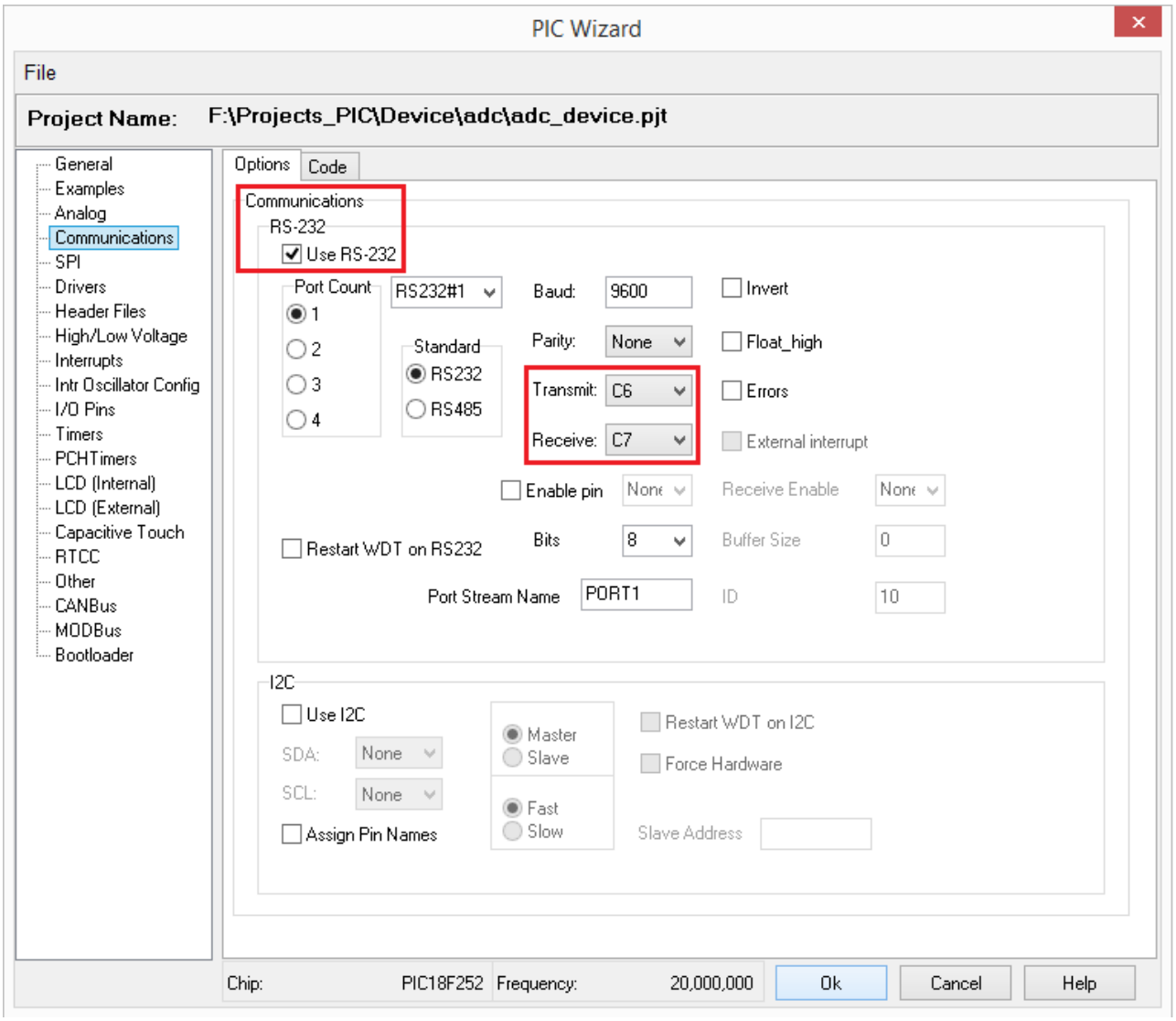

Рисунок 3.3.11 – Розділ **Communications** майстра **PIC Wizard**

Розділ **Analog** (рис. 3.3.12) служить для налаштування вбудованого АЦП (конфігурація аналогових входів, частота і розрядність перетворення).

7) У розділі **Analog** встановити:

у полі **Analog Input:**

- ◉ **A0 A1 A2 A3.**

Вказати:

розрядність АЦП: **8 (0-255)** або **10 (0-1023)** розрядів:

- **Units:** **0-255 / 0-1023**
- **Clock:** **4 us**

Конфігурування можна зробити з **IDE PCWH** при створенні проекту (рис. 3.3.12):

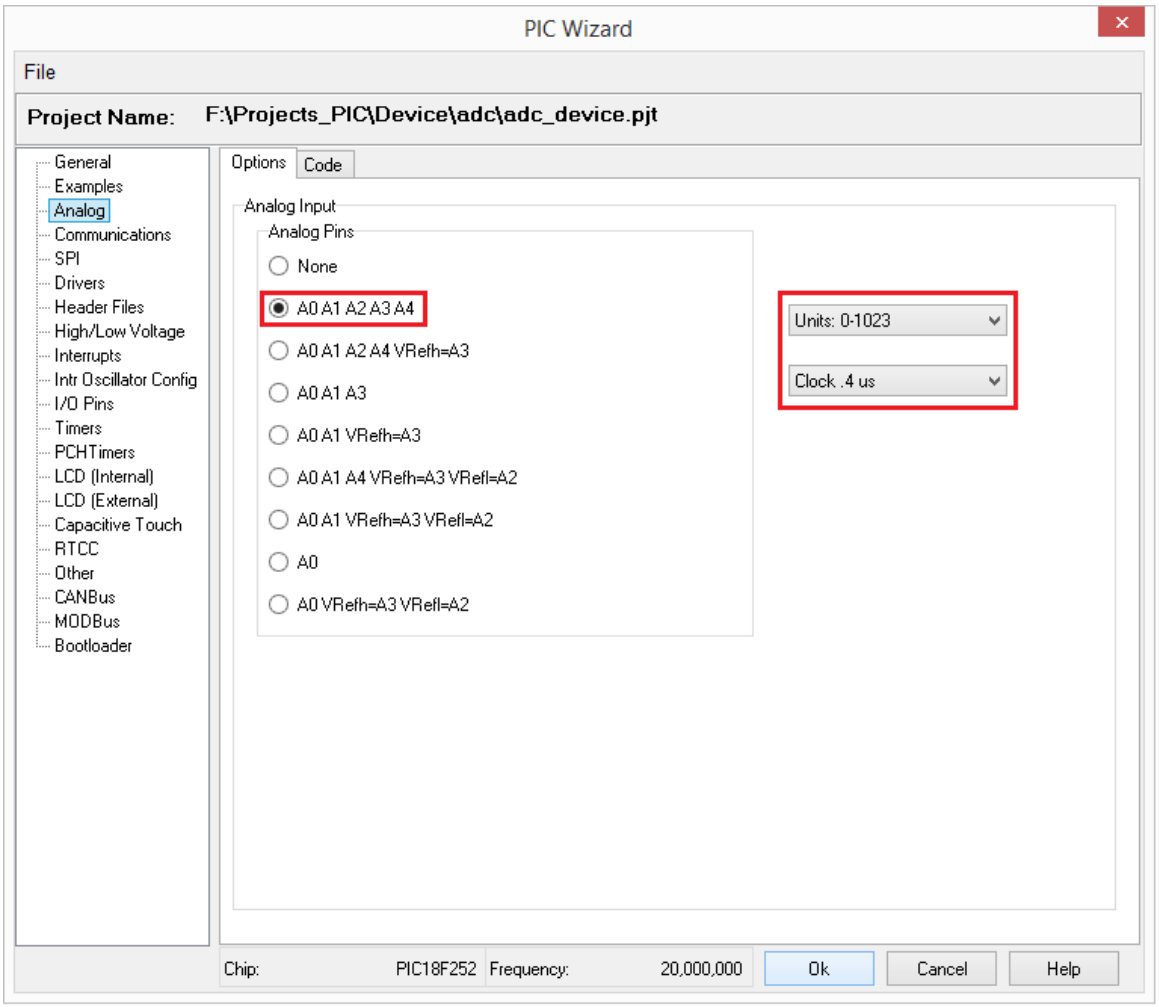

Рисунок 3.3.12 – Конфігурування АЦП із **IDE PCWH** при створенні проекту у розділі **Analog** майстра **PIC Wizard**

Буде згенерований і доданий програмний код у вихідний файл відповідно до параметрів на поточній вкладці.

Для перегляду програмного коду, слід вибрати вкладку **Code**.

8. Натиснути кнопку **OK**.

Буде згенерований наступний код (рис. 3.3.13):

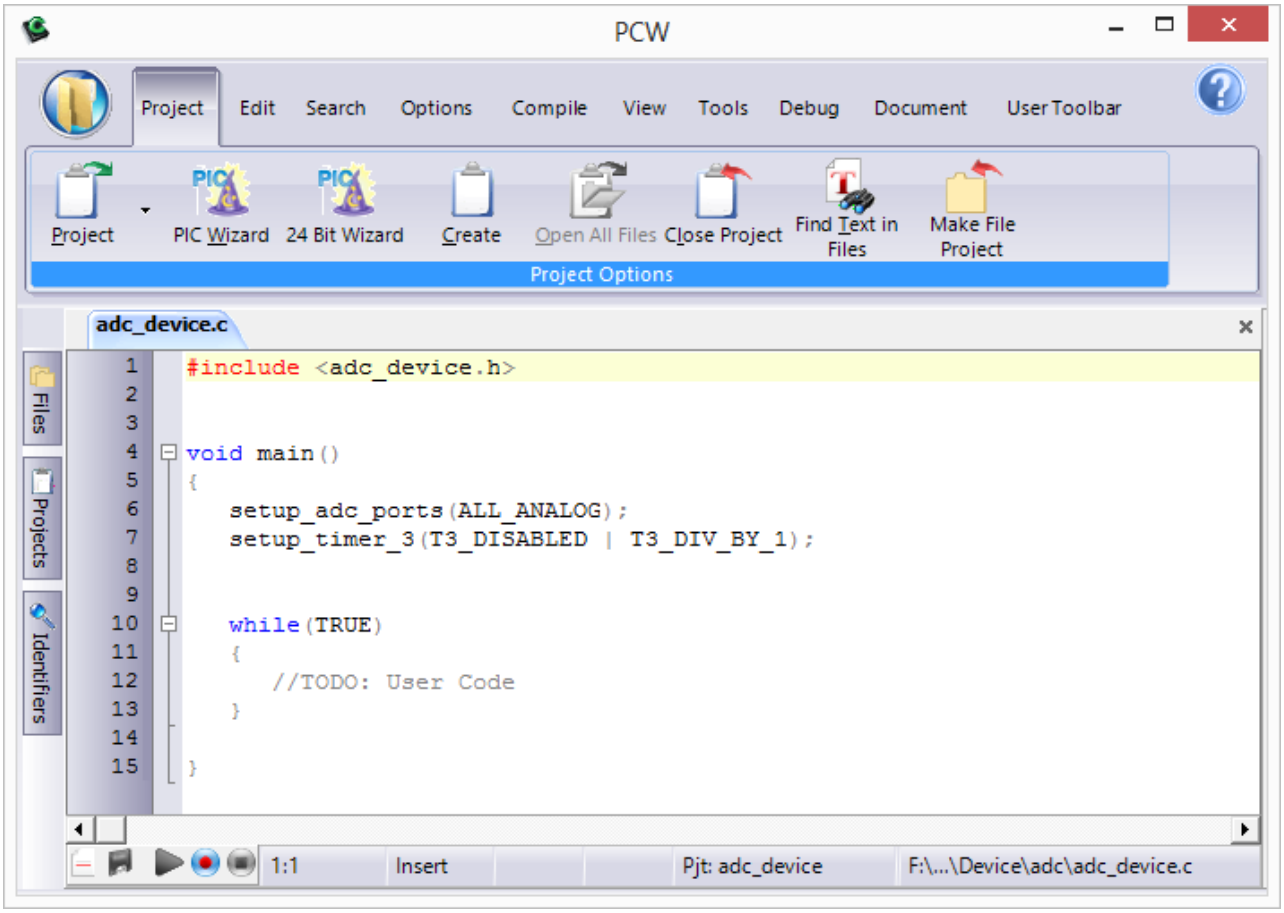

Рисунок 3.3.13 – Згенерований майстром **PIC Wizard** програмний код

Лістинг 3.3.1 – Приклад програмного коду, згенерованого майстром **PIC Wizard**

adc_device.c

```
#include <adc_device.h>

void main()
{
setup_adc_ports(ALL_ANALOG);
setup_timer_3(T3_DISABLED | T3_DIV_BY_1);
```

```
while(TRUE)
{
//TODO: User Code
}
}
```

adc_device.h

```
#include <18F252.h>
#device adc=10 //( або 8)

#FUSES NOWDT                        //No Watch Dog Timer
#FUSES WDT128                       //Watch Dog Timer uses 1:128
//Postscale
#FUSES HS     //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)
#FUSES NOBROWNOUT                   //No brownout reset
#FUSES NOLVP                        //No low voltage prgming, B3(PIC16)
                                    //or B5(PIC18) used for I/O

#use delay(clock=20000000)

#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8,stream=PORT1)
```

Проект готовий, можна приступати до написання програми.

ПОЯСНЕННЯ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ «ADC»

Приклад зчитування даних з АЦП з каналу 2 і перетворення у вольти:

```
float volt;
int16 adc_val;
//====== вибрати номер каналу
set_adc_channel(2);
//====== затримка для роботи АЦП
delay_us(50);
//====== прочитати значення
adc_val = read_adc();
//====== визначити вольти
volt = adc_val * ADC_CVANT;
```

Змінна `ADC_CVANT` є ціною ділення шкали АЦП і обчислюється як частка від ділення V_{REF} на розрядність АЦП, виражену у двійкових одиницях:

```
//====== опорна напруга = напруга живлення
const float REF_VDD = 5.00;
const float ADC_8 = 255;
const float ADC_10 = 1023;

//====== ділення шкали: 5В/шкала (розрядність АЦП)
//=== 8 розрядів
const float ADC_CVANT = REF_VDD/ADC_8;
//=== 10 розрядів
const float ADC_CVANT = REF_VDD/ADC_10;
```

Для визначення натиснутої кнопки на клавіатурі необхідно визначити рівень напруги на кожній клавіші:

```
//====== коди клавіш у вольтах відповідно до розведення плати
const float KBD_CVANT = REF_VDD/12;              //усього 12 кнопок

//=== кнопки за схемою відповідно до розведення плати
const float ADC_BTN_0 = KBD_CVANT*4;
const float ADC_BTN_1 = KBD_CVANT*3;
...

const float ADC_BTN_STAR = KBD_CVANT*8;
const float ADC_BTN_DIEZ = KBD_CVANT*12;

//=== не натиснута жодна кнопка
const float ADC_BTN_NOT = KBD_CVANT/2;
```

Потім порівнювати значення АЦП у вольтах при натисканні клавіші з рівнями напруги, визначеним для кожної клавіші.

При збігу значень здійснюється ідентифікація натиснутою клавіші:

```
if((pushed_button > ADC_BTN_STAR-LUFT)&&(pushed_button <
ADC_BTN_STAR+LUFT))
{
char_button = '*';
}

if((pushed_button > ADC_BTN_DIEZ-LUFT)&&(pushed_button <
ADC_BTN_DIEZ+LUFT))
{
char_button = '#';
}

//====== клавіша не натиснута
if((pushed_button < ADC_BTN_NOT))
{
char_button = 0;
}
```

Примітка: після виводу значень на LCD необхідно ввести затримку часу індикації:

```
delay_ms(100);
```

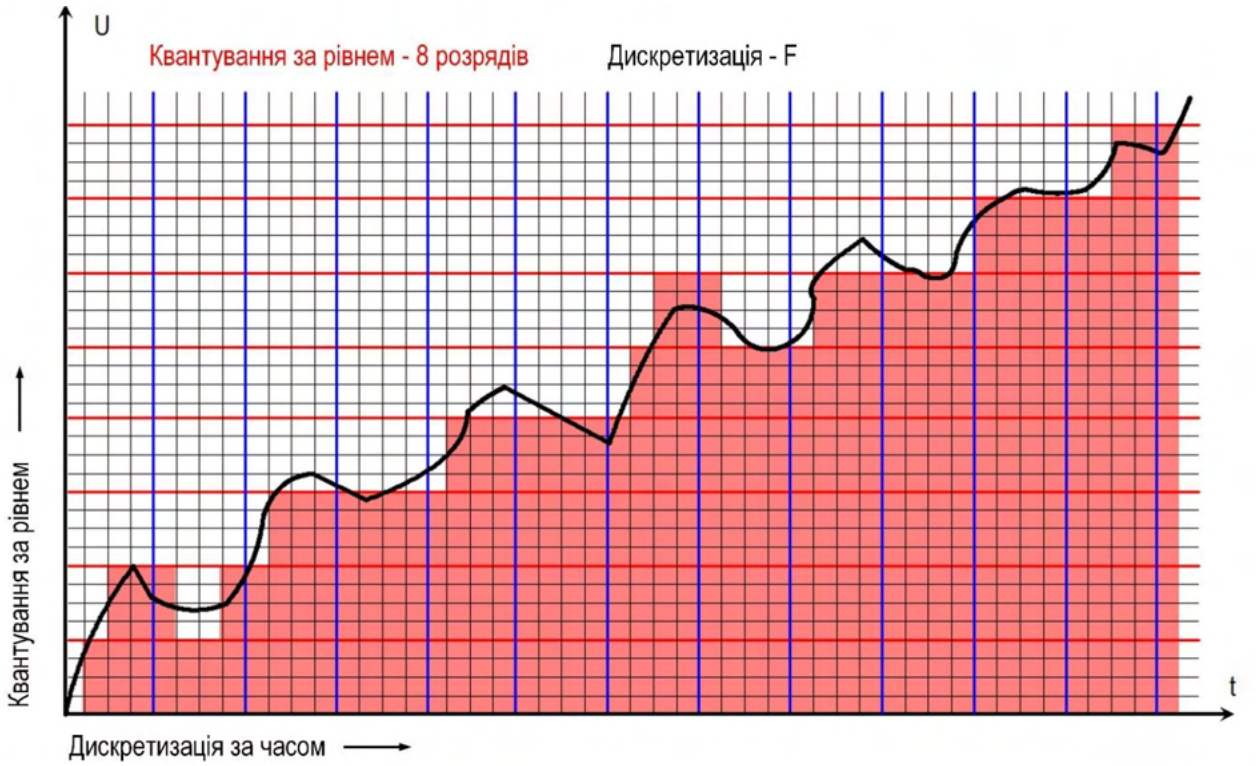

Рисунок 3.3.14 – Перетворення сигналу частотою дискретизації F і розрядністю = 8

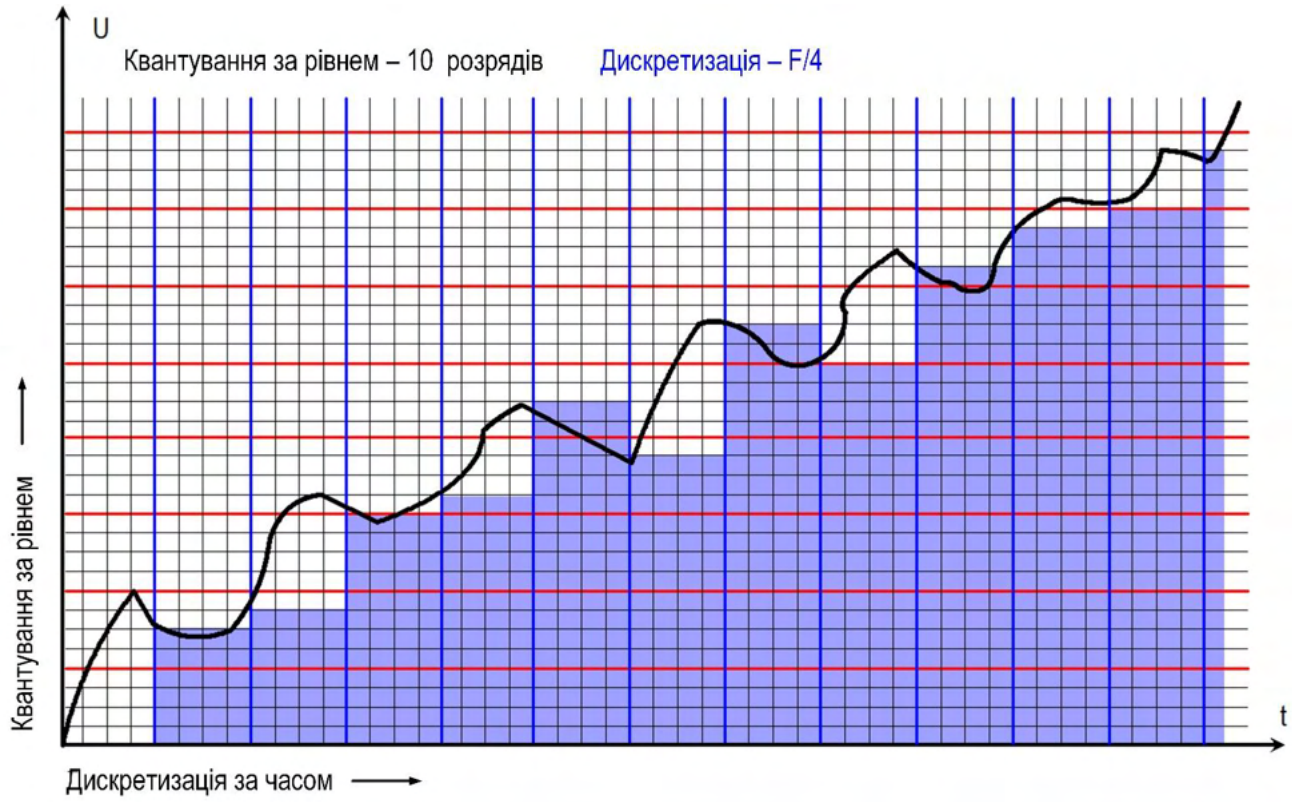

Рисунок 3.3.15 – Перетворення сигналу частотою дискретизації F/4 і розрядністю = 10

На рисунку 3.3.14 показані результати перетворення аналогового сигналу з низькою розв'язною здатністю АЦП – 8 розрядів (256 рівнів) і високою частотою дискретизації.

На рисунку 3.3.15 показані результати перетворення аналогового сигналу з більш високою розв'язною здатністю АЦП – 10 розрядів (1024) рівня і низькою частотою дискретизації.

Представлені графіки показують, що низька розв'язна здатність АЦП і низька частота дискретизації не дають можливості якісного перетворення аналогового сигналу.

Наприклад, у мікроконтролері **PIC18F252** розрядність АЦП – 8 і 10 розрядів. Частота дискретизації визначається програмістом.

Робота модуля АЦП

У мікроконтролері **PIC18F252** модуль аналого-цифрового перетворення (АЦП) має 5 аналогових входів (рис 3.3.16).

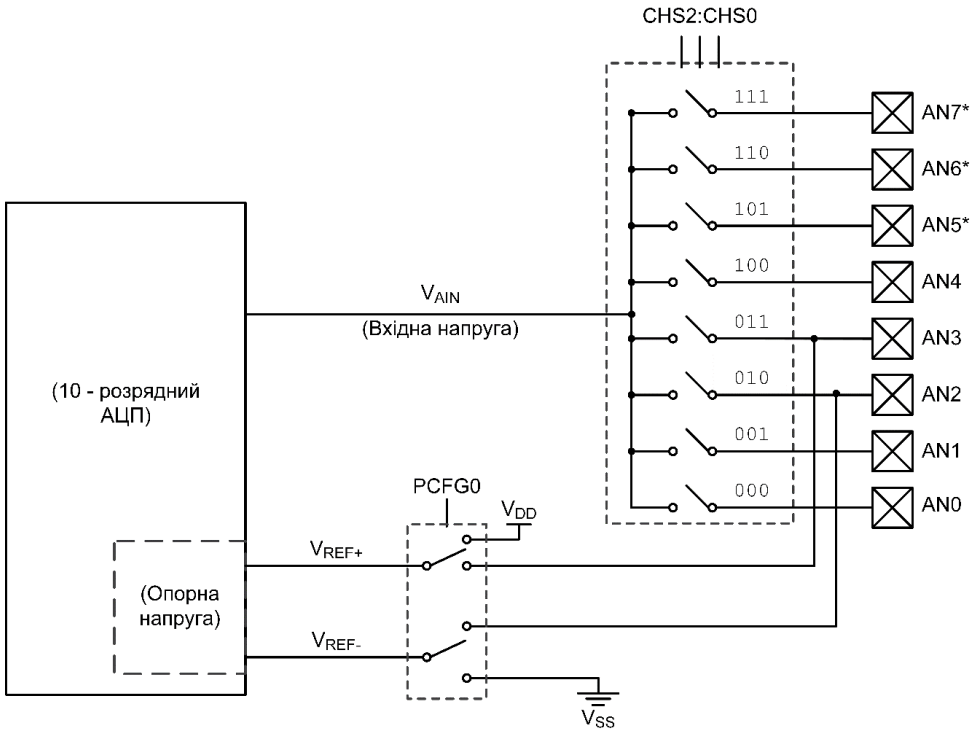

Рисунок 3.3.16 – Структурна схема модуля АЦП

Кожний канал порту, пов'язаний з модулем АЦП, може бути настроєний як аналоговий вхід (RA3 і RA2 як входи опорної напруги) або цифрового входу/виходу.

Вхідний аналоговий сигнал надходить на АЦП через комутатор каналів і перетворюється у відповідний 10-розрядний цифровий код.

Позитивний і негативний вхід опорної напруги може бути програмно обраний з виводів V_{DD} і V_{SS}, або з входом AN3 / V_{REF+} і AN2 / V_{REF-}

Допускається робота модуля АЦП у SLEEP режимі мікроконтролера, при цьому в якості джерела імпульсів для АЦП повинен бути обраний RS генератор.

Принцип перетворення представлений на рис. 3.3.17.

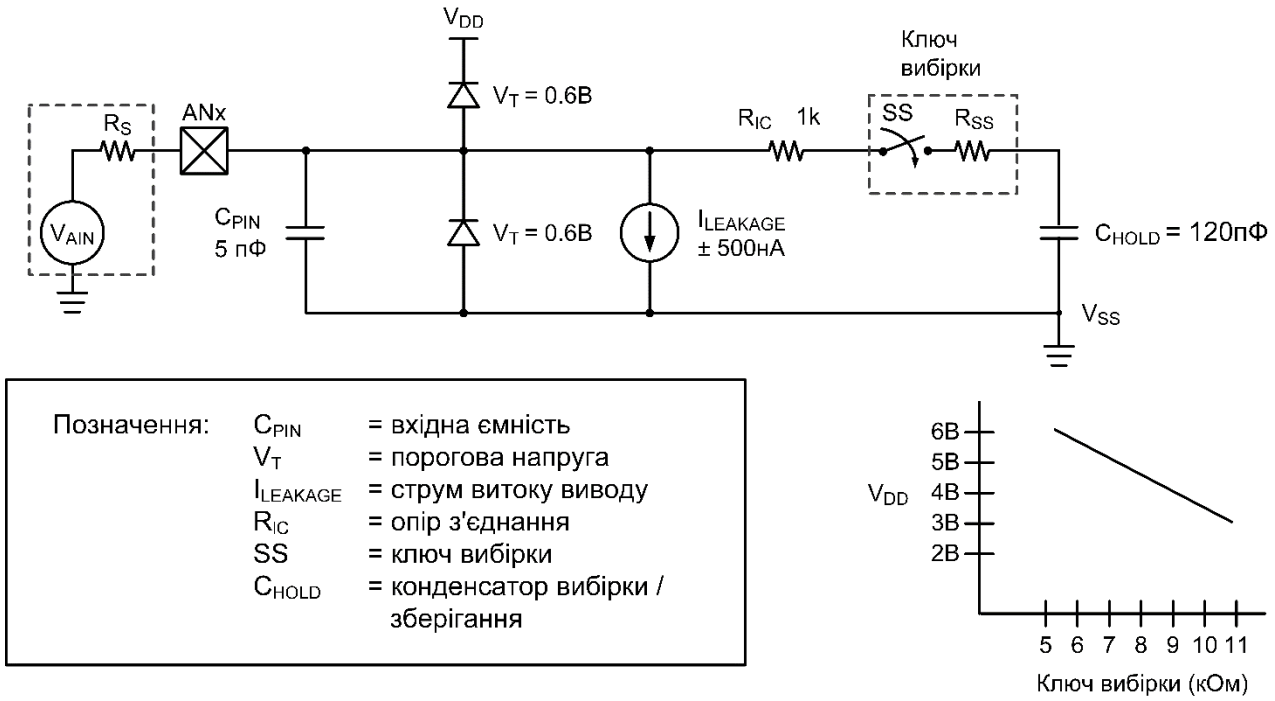

Рисунок 3.3.17 – Схема аналогового входу АЦП

Вхідний аналоговий сигнал через комутатор каналів заряджає внутрішній конденсатор АЦП C_{HOLD}.

Модуль АЦП перетворює напругу, яка утримувалась на конденсаторі C_{HOLD} у відповідний 10-розрядний цифровий код методом послідовного наближення.

При скиданні мікроконтролера значення всіх його регістрів встановлюються за замовчуванням.

Скидання виключає модуль АЦП, а також зупиняє процес перетворення, якщо він був початий.

Розташування виводів АЦП мікроконтролера представлено на рис. 3.3.18:

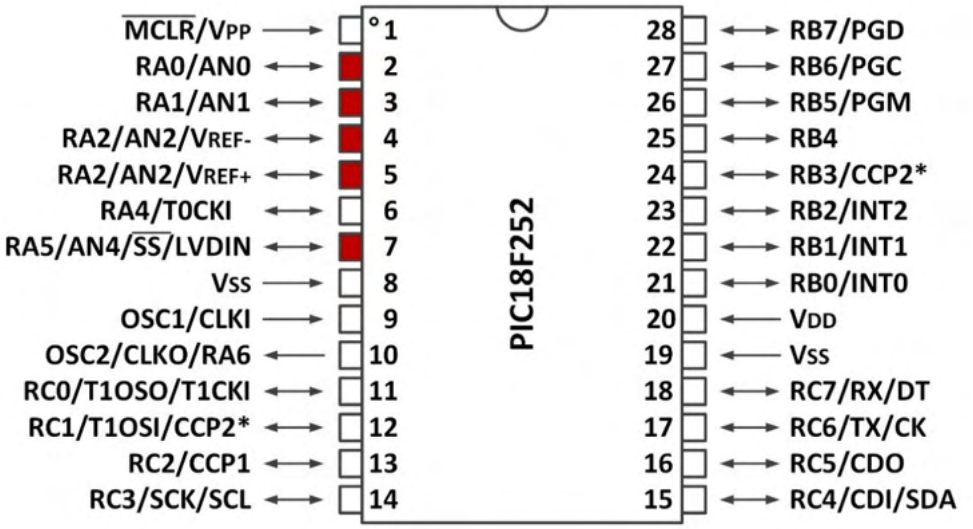

Рисунок 3.3.18 – Розташування виводів АЦП мікроконтролера

Приклад виконання лабораторної роботи «ADC» відповідно до варіанту завдання для лабораторної роботи «ADC»

Варіант завдання для виконання лабораторної роботи **«ADC»** представлений у таблиці 3.3.2.

Таблиця 3.3.3 – Варіант завдання для виконання лабораторної роботи **«ADC»**

Канал АЦП								**Розрядність АЦП**	
Потенціометр				**Клавіатура**					
A0	A1	A2	A3	A0	A1	A2	A3	8	10
			+			+			+

Принципова схема підключень для варіанту АПК

Принципова схема підключень відповідно варіанту завдання (таб. 3.3.3) для варіанту **АПК**, для виконання лабораторної роботи **«ADC»** виглядає у такий спосіб (рис. 3.3.19):

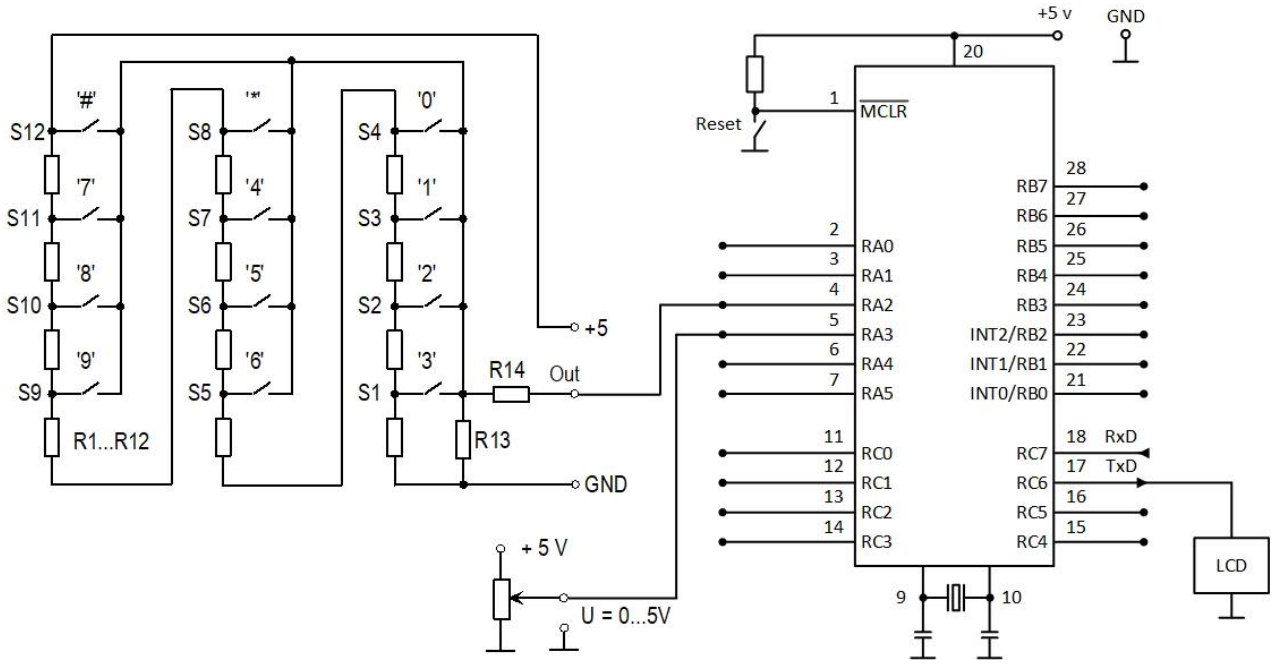

Рисунок 3.3.19 – Принципова схема підключень для лабораторної роботи **«ADC»** для варіанту **АПК**

Принципова схема підключень для варіанту «Proteus»

Принципова схема для варіанту **«Proteus»** відповідно варіанту завдання (таб. 3.3.3), для виконання лабораторної роботи **«ADC»** виглядає у такий спосіб (рис. 3.3.20):

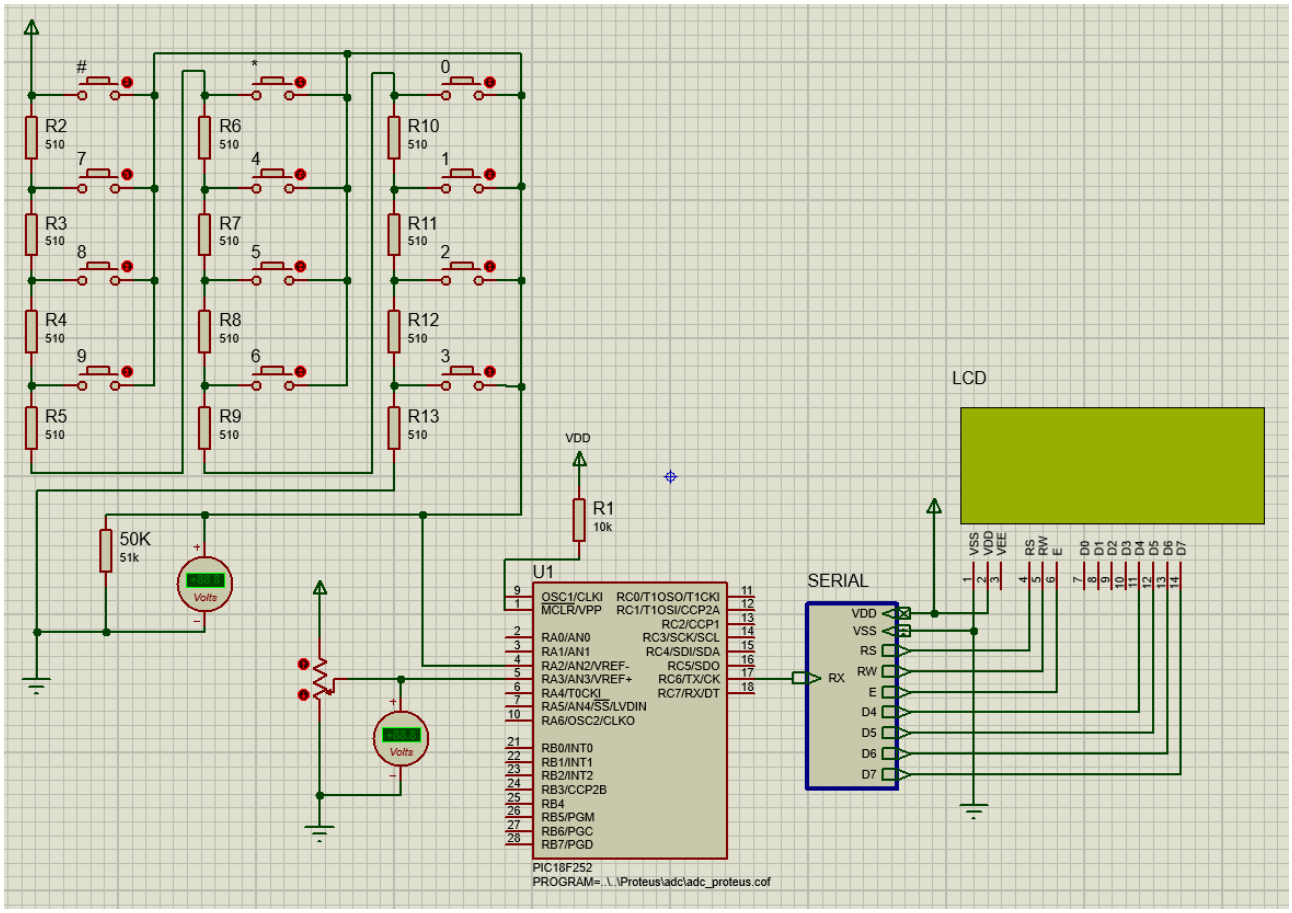

Рисунок 3.3.20 – Принципова схема підключень лабораторної роботи **«ADC»** для варіанту **«Proteus»**

Результат виконання лабораторної роботи «ADC» для варіанту «Proteus»

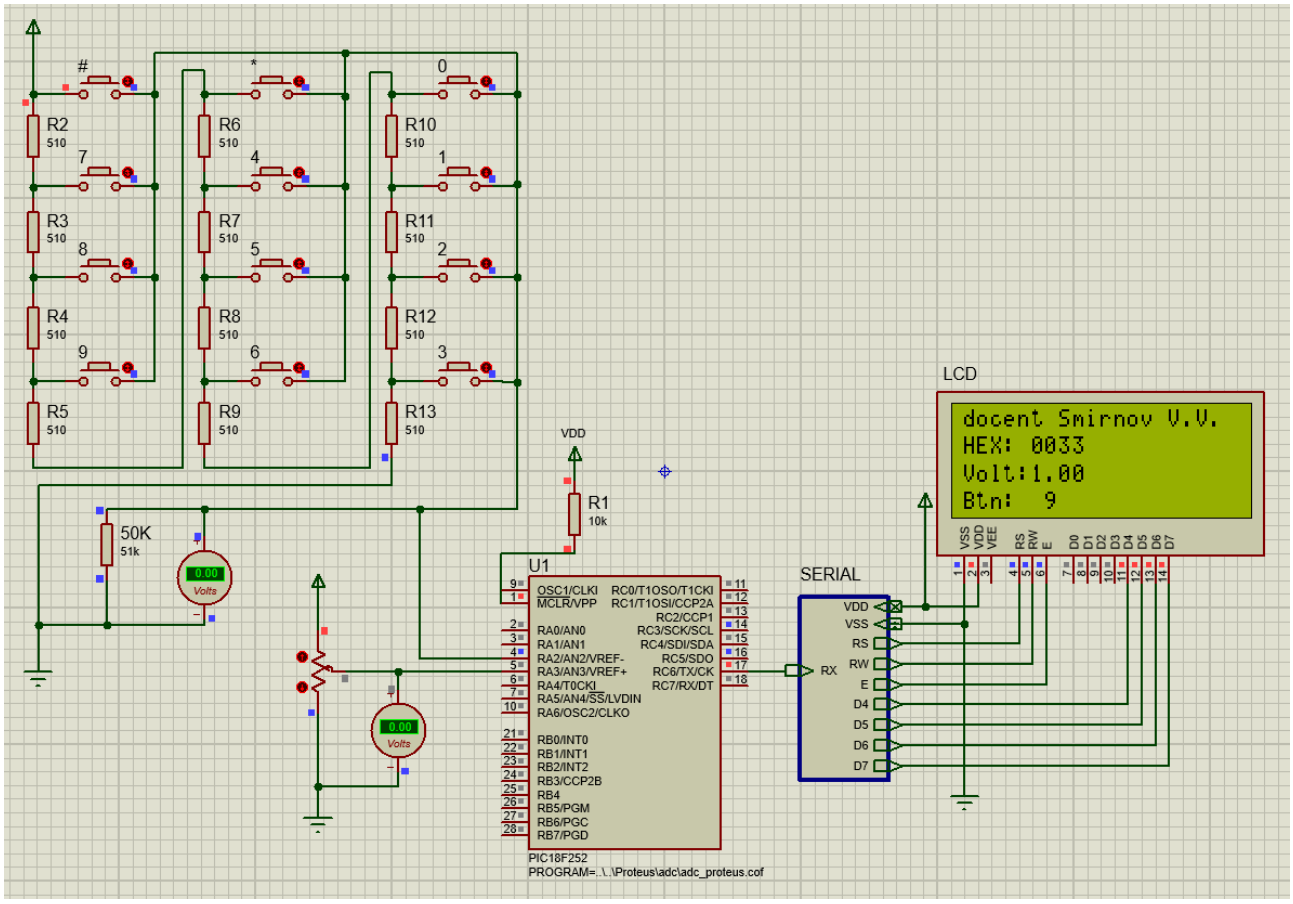

Рисунок 3.3.21 – Результат виконання лабораторної роботи **«ADC»** для варіанту **«Proteus»**

Контрольні питання

1) призначення АЦП і методи перетворення;
2) конфігурування мікроконтролера для роботи з АЦП;
3) вплив розрядності АЦП на точність перетворення;
4) опорна напруга VREF і його вплив на параметри АЦП;
5) як визначається ціна ділення (вага) шкали АЦП?

ЛАБОРАТОРНА РОБОТА № 4 - «DAC»

Тема: Робота з ЦАП

Ціль роботи:

Отримання навичок роботи із ЦАП MCP4921 і написання драйвера послідовного інтерфейсу SPI.

Завдання лабораторної роботи

- для варіанту **АПК** намалювати принципову схему підключень відповідно до варіанту для виконання лабораторної роботи (таб. 3.4.1);
- для варіанту **«Proteus»** використовувати готову схему відповідно до варіанту для виконання лабораторної роботи;
- створити алгоритм програми роботи з ЦАП по інтерфейсу SPI;
- написати програму драйвера послідовного інтерфейсу SPI;
- здійснити зчитування рівня напруги з потенціометра, оцифрувати за допомогою АЦП відповідно до варіанту для виконання лабораторної роботи;
- відобразити рівень поточної напруги у вольтах на LCD;
- вивести цифрове значення напруги на ЦАП через інтерфейс SPI;
- здійснити зчитування рівня напруги з виходу ЦАП, оцифрувати за допомогою АЦП;
- відобразити рівень напруги з виходу ЦАП у вольтах на LCD;
- здійснити виведення результатів на LCD відповідно до варіанту для виконання лабораторної роботи.

Виведення на LCD повинне містити:

- зчитані дані з АЦП у вольтах;
- зчитані дані з ЦАП у вольтах;
- групу студента;
- прізвище та ініціали студента.

Варіанти для виконання лабораторної роботи «DAC»

Таблиця 3.4.1 – Варіанти для виконання лабораторної роботи **«DAC»**

№	Канал АЦП							
	Потенціометр				ЦАП			
	A0	A1	A2	A3	A0	A1	A2	A3
1	+					+		
2		+					+	
3			+					+
4				+	+			
5		+			+			
6			+			+		
7	+						+	
8	+							+
9		+					+	
10			+		+			
11				+	+			
12	+					+		
13		+			+			
14			+					+
15				+			+	

Послідовність виконання лабораторної роботи для варіанту АПК

1) для варіанту **АПК** намалювати принципову схему підключень відповідно до варіанту для виконання лабораторної роботи та у відповідності зі структурною і функціональною схемою (рис. 8.1, рис. 3.4.2);
2) створити свій директорій для файлів проекту;
3) відкрити інтегроване середовище розробки **PCWH**;
4) створити проект у інтегрованому середовищі розробки **PCWH**;
5) створити алгоритм програми роботи з ЦАП по інтерфейсу SPI;
6) написати програму мовою програмування **C**, що реалізує створений алгоритм, відповідно до варіанту для виконання лабораторної роботи;

7) скомпілювати програму, одержати бінарні файли з розширеннями ***.HEX** і ***.COF** (файли з розширеннями ***.HEX** і ***.COF** створюються підчас компіляції програми);
8) завантажити бінарний файл з розширенням ***.HEX** у пам'ять програм мікроконтролера програмою **PICkit.exe**;
9) на **АПК** зкумутувати схему підключень. Підключити потенціометр, вольтметр АЦП і ЦАП відповідно до варіанту для виконання лабораторної роботи у відповідності з рис. 3.4.1, рис. 3.4.2;

Структурна схема підключень виглядає у такий спосіб (рис. 3.4.1):

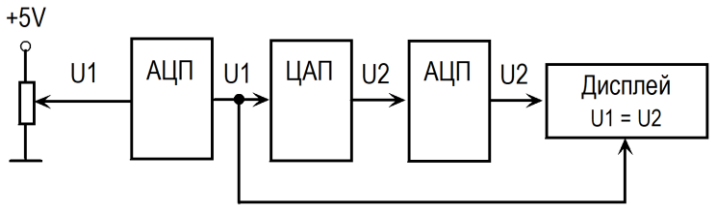

Рисунок 3.4.1 – Структурна схема підключень

Функціональна схема підключень виглядає у такий спосіб (рис. 3.4.2):

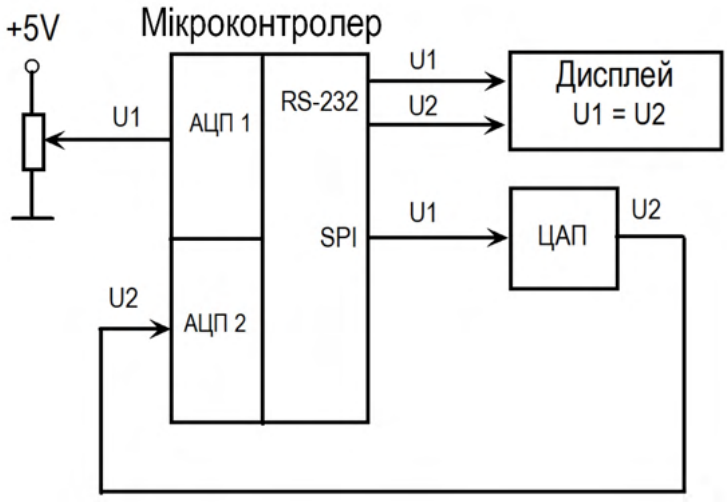

Рисунок 3.4.2 – Функціональна схема підключень

10) виконати програму.

Послідовність виконання лабораторної роботи для варіанту «Proteus»

1) для середовища моделювання **«Proteus»** використовувати готову схему;
2) створити свій директорій для файлів проекту;
3) відкрити інтегроване середовище розробки **PCWH**;
4) створити проект у інтегрованому середовищі розробки **PCWH**;
5) створити алгоритм програми роботи з ЦАП по інтерфейсу SPI;
6) написати програму мовою програмування **C**, що реалізує створений алгоритм, відповідно до варіанту для виконання лабораторної роботи;
7) скомпілювати програму, одержати бінарні файли з розширеннями ***.HEX** і ***.COF** (файли з розширеннями ***.HEX** і ***.COF** створюються підчас компіляції програми);
8) відкрити середовище моделювання **«Proteus»**;
9) у папці **«Proteus_students»** вибрати папку лабораторної роботи **«DAC»**;
10) відкрити файл проекту з розширенням ***.DSN**;
11) у середовищі моделювання **«Proteus»** змінити схему підключень відповідно до варіанту для виконання лабораторної роботи;
11) завантажити заздалегідь скомпільований бінарний файл з розширенням ***.COF** або ***.HEX** у мікроконтролер (файли з розширеннями ***.COF** і ***.HEX** створюються підчас компіляції програми);
12) у середовищі моделювання **«Proteus»** виконати програму.

Послідовність виконання лабораторної роботи для варіанту «Proteus» з використанням шаблону програми

1) для середовища моделювання **«Proteus»** використовувати готову схему;
2) відкрити папку **«Proteus_students»**;
3) відкрити інтегроване середовище розробки **PCWH**;
4) у папці **«Proteus_students»** вибрати папку лабораторної роботи **«DAC»**;
5) відкрити файл проекту з розширенням ***.pjt**;
6) у редакторі **IDE PCWH** відкрити шаблон файлу програми з розширенням ***.c**;
7) відкрити середовище моделювання **«Proteus»**;
8) у папці **«Proteus_students»** вибрати папку лабораторної роботи **«DAC»**;
9) відкрити файл проекту з розширенням ***.DSN**;
10) у середовищі моделювання **«Proteus»** змінити схему підключень відповідно до варіанту для виконання лабораторної роботи;
11) створити алгоритм програми роботи з ЦАП по інтерфейсу SPI;
12) у інтегрованому середовищі розробки **PCWH**, використовуючи шаблон програми самостійно написати програму мовою програмування **C**, що реалізує створений алгоритм, відповідно до варіанту для виконання лабораторної роботи;
13) скомпілювати програму, одержати бінарні файли з розширеннями ***.HEX** і ***.COF** (файли з розширеннями ***.HEX** і ***.COF** створюються підчас компіляції програми);
14) у середовищі моделювання **«Proteus»** виконати програму.

Примітка: *файл демонстрації виконання лабораторної роботи* ***«DAC»*** *розташований на сайті http://pksm.kntu.kr.ua/SCME.html.*

ТЕОРЕТИЧНІ ВІДОМОСТІ

Цифро-аналоговий перетворювач (ЦАП)

Цифро-аналоговий перетворювач *(ЦАП)* призначений для перетворення цифрового сигналу, заданого у вигляді двійкового *N*-розрядного коду, у відповідну напругу або струм.

Якість перетворення залежить від швидкості перетворення і розрядності ЦАП.

Характеристики ЦАП

Залежність вихідної напруги (струму) $U_{вих}(t)$ від вхідного цифрового сигналу $D(t)$, що змінюється від 0 до 2^{N-1}, називають характеристикою перетворення ЦАП. Ця характеристика може бути представлена у вигляді ступінчастої кривої. Величина сходинки відповідає одиниці молодшого значущого розряду (МЗР).

За відсутності апаратних похибок середні точки сходинок розташовані на прямій 1, якій відповідає ідеальна характеристика перетворення (рис. 3.4.3).

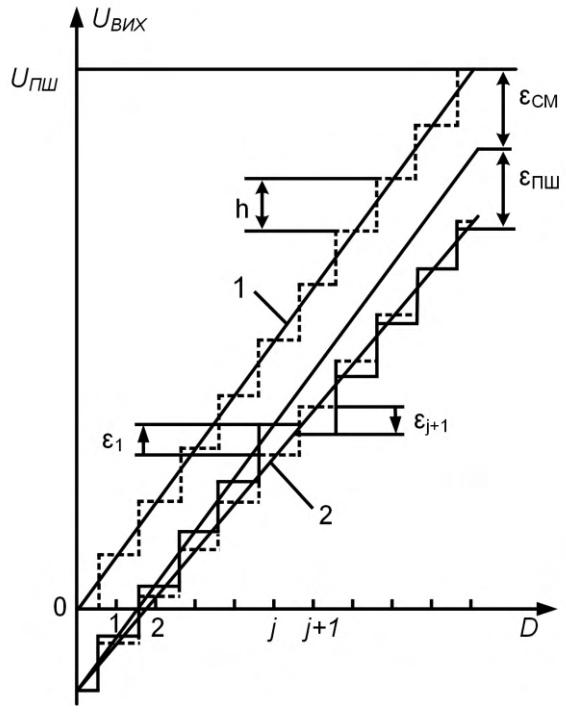

Рисунок 3.4.3 - Характеристика перетворення ЦАП

Реальна характеристика перетворення може відрізнятися від ідеальної розташуванням, розмірами і формою сходинок.

Для кількісного опису цих відмінностей використовуються характеристики, що називаються ***статичними***. Зміни вихідного сигналу в часі при стрибкоподібний зміні вхідного коду від мінімуму («всі нулі») до максимуму («всі одиниці») описуються динамічними характеристиками ЦАП.

Статичні характеристики ЦАП

Розрядність. Число розрядів (двійкових символів) N, що передають кодований вхідний сигнал. Йому відповідає число рівнів квантування 2^N.

Число каналів. Число аналогових виходів ЦАП, на яких формується вихідна напруга. Залежно від схеми побудови цифровий код для кожного з каналів подається або на різні цифрові входи ЦАП, або на один і той же з поділом за часом. Як правило, частота перетворення і напруга повної шкали для всіх каналів ЦАП рівні. Однак в деяких моделях ІМС ЦАП напруга повної шкали для кожного каналу задається окремо і залежить від опорної напруги.

Роздільна здатність. Приріст $U_{ВИХ}$ при збільшенні коду на одиницю називається *кроком квантування*. Номінальне значення кроку квантування становить:

$$h = \frac{U_{ПШ}}{2^N\text{-}1}$$

де:

$\boldsymbol{U_{ПШ}}$ - номінальна максимальна вихідна напруга ЦАП (напруга повної шкали);

$\boldsymbol{N}$ - розрядність ЦАП. Чим більше розрядність перетворювача, тим вища його роздільна здатність (менше h).

Похибка повної шкали. Відносна різниця між реальним і ідеальним значеннями межі шкали перетворення при відсутності зсуву нуля:

$$\delta_{ПШ} = \frac{\varepsilon_{ПШ}}{U_{ПШ}} \cdot 100\%$$

Вона є мультиплікативною складовою повної похибки. Іноді виражається відповідним числом молодшого значущого розряду.

Похибка зміщення нуля. Значення $U_{ВИХ}$, коли вхідний код ЦАП дорівнює нулю, називається *адитивною складовою повної похибки* ($\varepsilon_{СМ}$, рис. 3.4.3).

Зазвичай вказується в мілівольтах або у відсотках від повної шкали:

$$\delta_{ПШ} = \frac{\varepsilon_{СМ}}{U_{ПШ}} \cdot 100\%$$

Нелінійність *(інтегральна нелінійність. Integral Non-Linearity, INL).* Максимальне відхилення реальної характеристики перетворення $U_{ВИХ}(D)$ від оптимальної (лінія 2, рис. 3.4.3).

Оптимальна характеристика знаходиться емпірично так, щоб мінімізувати значення похибки нелінійності. Нелінійність зазвичай визначається у відносних одиницях.

Для характеристики, наведеної на рис. 3.4.3, цей параметр визначається за формулою:

$$\delta_{Л} = \frac{\varepsilon_j}{U_{ПШ}} \cdot 100\%$$

Диференціальна нелінійність *(Differential Non-Linearity, DNL).* Максимальна зміна (з урахуванням знаку) відхилення реальної характеристики перетворення $U_{ВИХ}(D)$ від оптимальної при переході від одного значення вхідного коду до іншого суміжного значення.

Визначається у відносних одиницях або у МЗР:

$$\delta_{ДЛ} = \frac{\varepsilon_j + \varepsilon_{j+1}}{U_{ПШ}} \cdot 100\%$$

Монотонність характеристики перетворення. Цей параметр характеризує збільшення (зменшення) вихідної напруги ЦАП $U_{ВИХ}$ при збільшенні (зменшенні) вхідного коду D. Якщо диференційна нелінійність більше відносного кроку квантування $h/U_{ПШ}$, то характеристика перетворювача немонотонна. У цьому випадку «оптимальна пряма» виходить за межі «сходинок».

Температурна нестабільність. Температурна нестабільність ЦАП характеризується температурними коефіцієнтами похибки повної шкали, похибки зміщення нуля і похибки нелінійності.

Похибки повної шкали та зміщення нуля можуть бути усунуті калібруванням (підстроюванням). Похибки нелінійності простими засобами усунути не можна.

Динамічні характеристики ЦАП

Частота перетворення. Максимальна частота, з якою ЦАП може формувати вихідну напругу або струм, відповідний вхідному коду.

Частота перетворення - це основний параметр, що характеризує швидкодію ЦАП

Час встановлення. Під цією величиною розуміють інтервал часу від моменту зміни вхідного коду ($t = 0$) до моменту, коли в останній раз виконується рівність (d - допуск на вихідну напругу або струм ЦАП):

$$\delta_{ДЛ} = \frac{\varepsilon_j + \varepsilon_{j+1}}{U_{ПШ}} \cdot 100\%$$

Зазвичай цей допуск рівний половині кроку квантування h.

Швидкість наростання. Це максимальна швидкість зміни $U_{ВИХ}(t)$ під час перехідного процесу, яка визначається як відношення приросту $\Delta U_{ВИХ}$ до часу Δt, за який відбулося це збільшення (рис. 3.4.4).

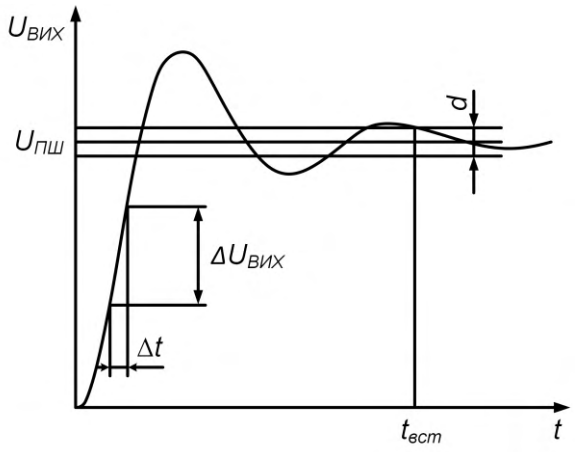

Рисунок 3.4.4 - Визначення часу встановлення і швидкості наростання

У ЦАП з струмовим виходом цей параметр у великій мірі залежить від типу вихідного операційного підсилювача.

Для перемножуючих ЦАП з виходом у вигляді напруги часто вказуються частота одиничного посилення і показники потужності смуги пропускання, які в основному визначаються властивостями вихідного підсилювача.

Шуми ЦАП. Шум на виході ЦАП може з'являтися з різних причин, що викликається фізичними процесами, які відбуваються в напівпровідникових пристроях.

Для оцінки якості ЦАП з високою роздільною здатністю прийнято використовувати поняття середньоквадратичного значення шуму.

Спектральну щільність шуму вимірюють зазвичай у $нВ/\sqrt{(Гц)}$ у заданій смузі частот.

Викиди *(імпульсні перешкоди)*. Викиди є круті короткі сплески чи провали у вихідній напрузі, що виникають під час зміни значень вихідного коду за рахунок несинхронності розмикання і замикання аналогових ключів в різних розрядах ЦАП.

Наприклад, якщо при переході від значення коду 011 ... 111 до значення 100 ... 000 ключ самого старшого розряду ЦАП з підсумовуванням вагових струмів відкриється пізніше, ніж закриються ключі молодших розрядів, то на виході ЦАП деякий час буде існувати сигнал, відповідний коду 000 ... 000.

Радикальним способом придушення викидів є використання пристроїв вибірки-зберігання. Викиди оцінюються по їх площі в одиницях $nB \cdot c$.

Класифікація ЦАП

За принципом роботи є дві основні групи ЦАП:

- послідовні;
- паралельні.

Перевагою послідовних ЦАП є простота схемотехнічної реалізації, недоліком невисока швидкодія.

Паралельні ЦАП забезпечують максимально можливу швидкодію, але це досягається за рахунок значного ускладнення схеми в порівнянні з послідовними ЦАП.

Класифікація ЦАП за схемотехнічними ознаками представлена на рис. 3.4.5

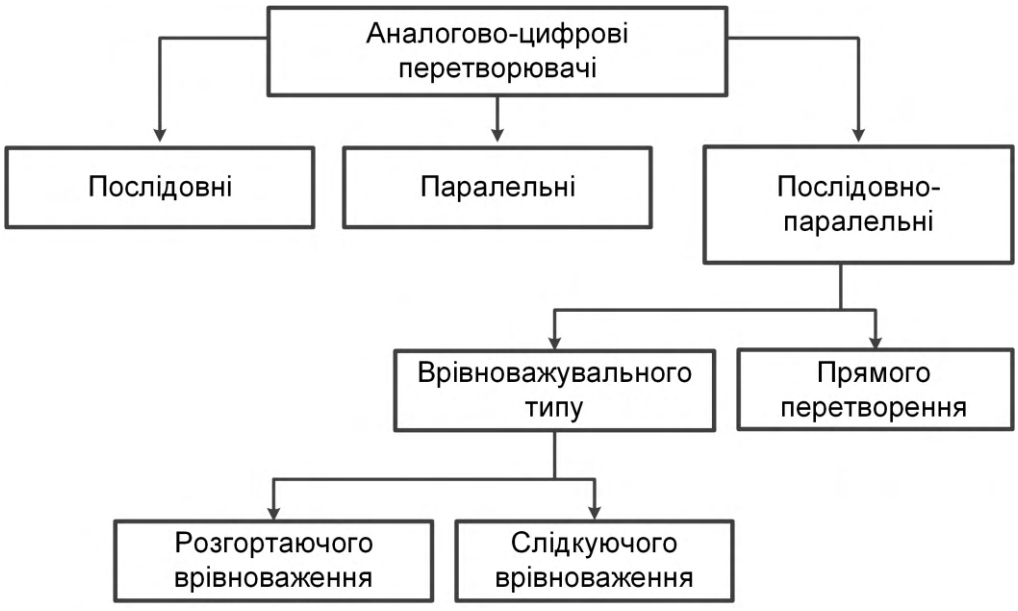

Рисунок 3.4.5 - Класифікація ЦАП за схемотехнічними ознаками

ЦАП можуть також класифікуватися по:

1. ***вигляду вихідного сигналу:***

- з струмовим виходом;
- з виходом у вигляді напруги;

2. ***полярності вихідного сигналу:***

- однополярні;
- біполярні;

3. ***характером опорного сигналу:***

- з постійним опорним сигналом;
- із змінним опорним сигналом;
- що множать;

4. ***швидкодії:***

- помірною швидкодією;
- високою швидкодією;

5. ***типу цифрового інтерфейсу (введення вихідною коду):***

- з послідовним введенням;
- з паралельним введенням;

6. ***числу ЦАП на кристалі:***

- одноканальні;
- багатоканальні.

ПРИКЛАД СТВОРЕННЯ ПРОЕКТУ У ІНТЕГРОВАНОМУ СЕРЕДОВИЩІ РОЗРОБКИ PCWH

Створення проекту у інтегрованому середовищі розробки PCWH за допомогою майстра PIC Wizard

Для запуску майстра **PIC Wizard** слід виконати відповідну команду меню **Project**, а потім вказати розміщення і ім'я головного файлу проекту.

В результаті відкриється вікно майстра, що складається з розділів з параметрами проекту (рис. 3.4.9).

Зовнішній вигляд вікна майстра може відрізнятися в залежності від версії компілятора **CCS-PICC** і обраного типу мікроконтролера.

1. Запустити **IDE PCWH**, натиснути кнопку **Projects** майстра **PIC Wizard** (рис. 3.4.6):

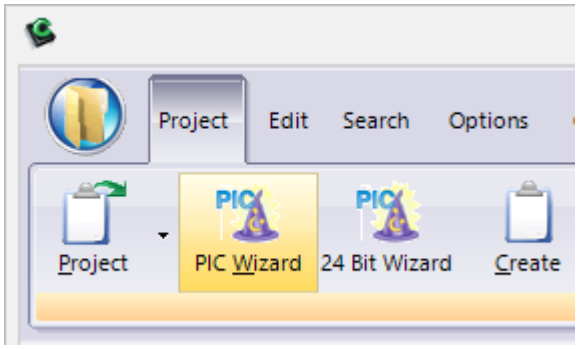

Рисунок 3.4.6 – Процес створення проекту у **IDE PCWH** за допомогою майстра **PIC Wizard**

або натиснути кнопку у вигляді відкритої папки 

2. Вибрати: **New › Project Wizard** (рис. 3.4.7):

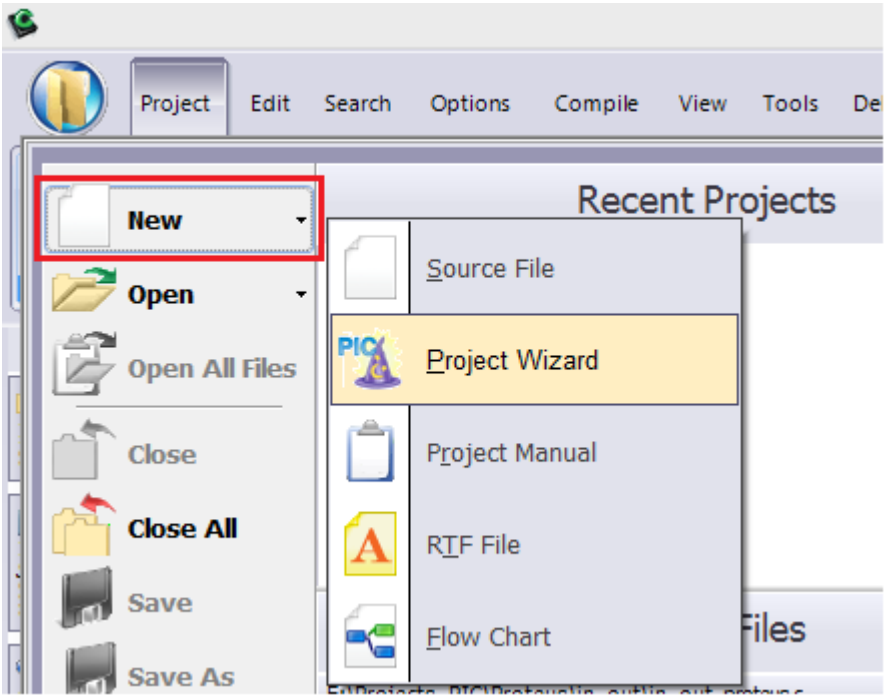

Рисунок 3.4.7 – Процес створення проекту у **IDE PCWH** за допомогою майстра **PIC Wizard**

3. Створити або вибрати свій директорій і записати у нього проект під будь-яким іменем латинським шрифтом, наприклад: «**dac.pjt**» (рис 4.8):

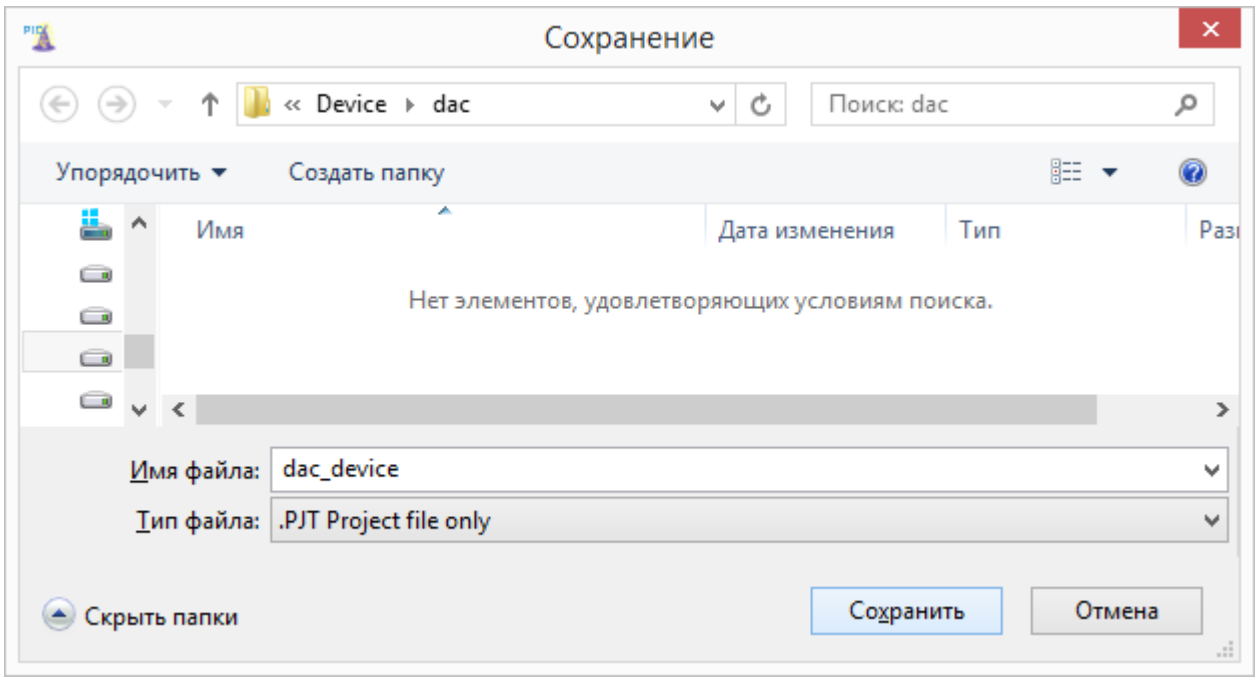

Рисунок 3.4.8 – Процес створення проекту у **IDE PCWH** за допомогою майстра **PIC Wizard**

У розділі **General** (рис 3.4.9) вибирається цільовий мікроконтролер (список **Device**, що розкривається), його робоча частота (поле **Oscillator Frequency**), а також настроюються загальні параметри, на зразок розрядів запобігання, типу джерела системної синхронізації, порогової напруги для скидання та ін. Для перегляду програмного коду, який буде згенерований і доданий у вихідний файл відповідно до параметрів на поточній вкладці, слід вибрати вкладку **Code**.

4. У розділі **General › Device** вибрати тип мікроконтролера, наприклад **PIC18F252**.

5. У розділі **General › Fuses** вибрати тип генератора **High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)**:

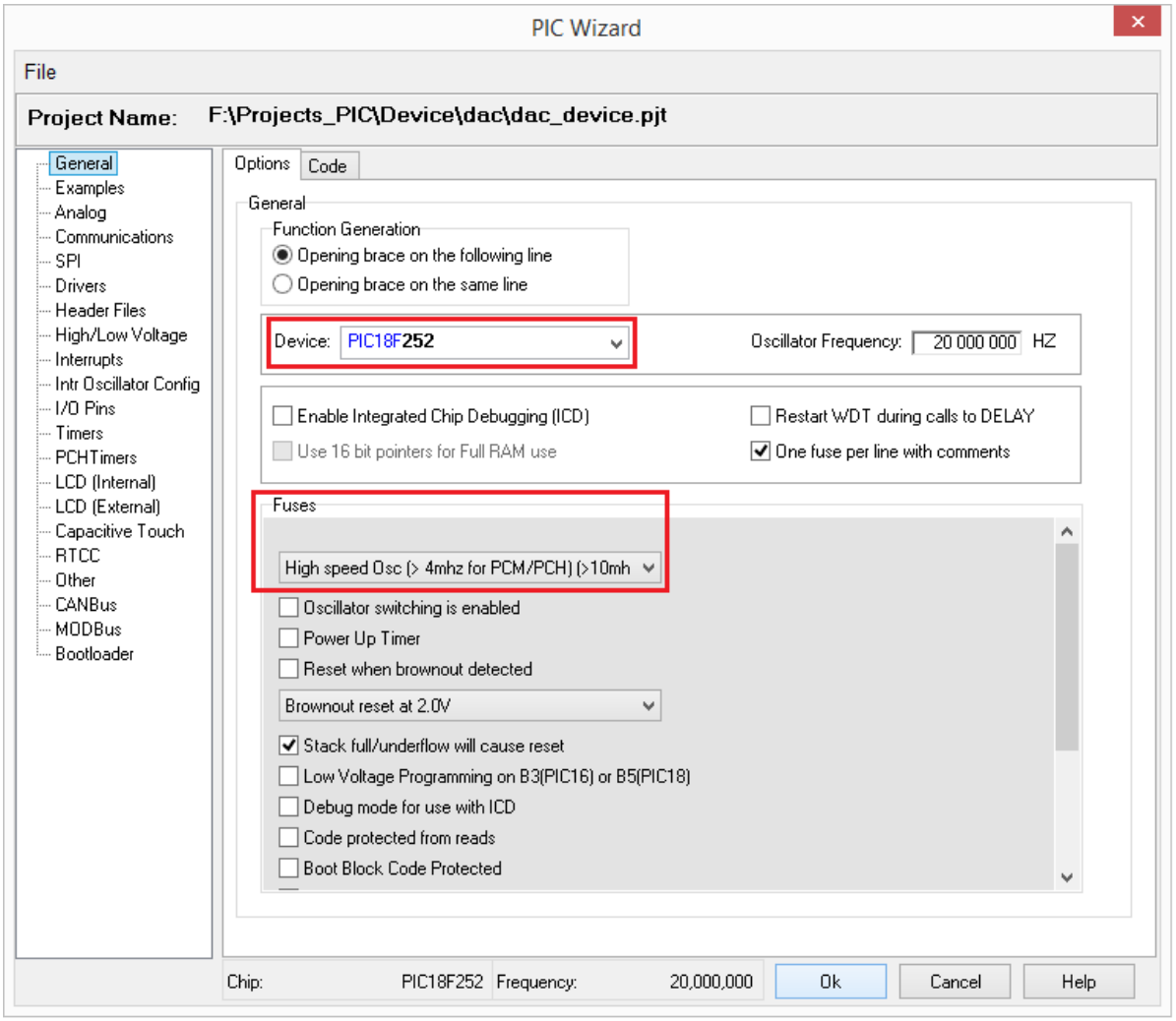

Рисунок 3.4.9 – Розділ **General** майстра **PIC Wizard**

У розділі **Communications** (рис. 3.4.10) налаштовуються параметри введення/виведення для інтерфейсів **RS-232** і **I²C** (швидкість обміну даними, відповідні лінії портів введення/виведення, перевірка помилок, режим **Master/Slave** і т.д.), а також активізується/відключається апаратний порт **PSP**.

6. У розділі **Communications** встановити мітку у полі **RS-232**:

- ☑ **Use RS-232**;

вказати:

- **Transmit:** **PIN C6** (передача);
- **Receive:** **PIN C7** (прийом).

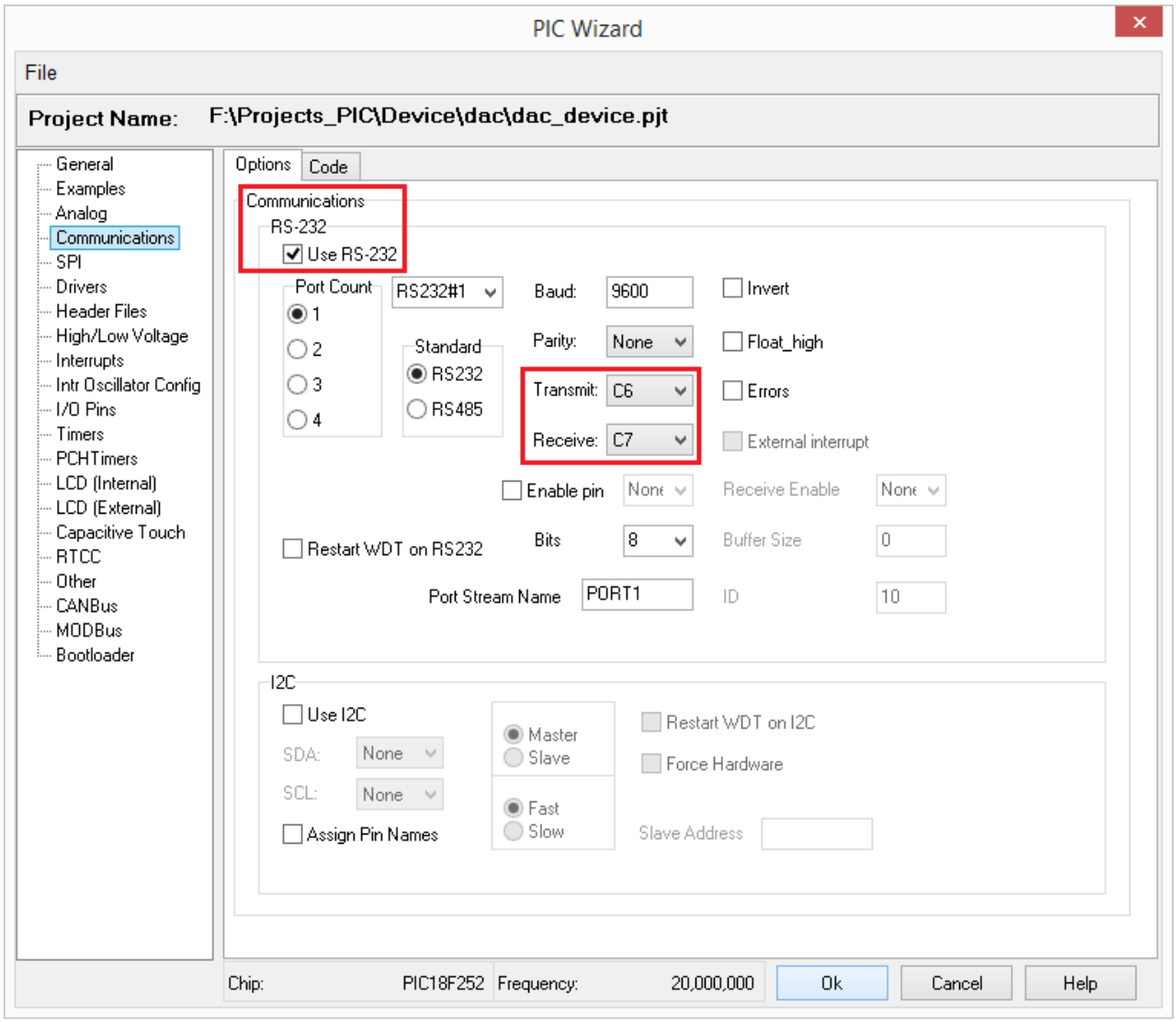

Рисунок 3.4.10 – Розділ **Communications** майстра **PIC Wizard**

Розділ **Analog** (рис. 3.4.11) служить для налаштування вбудованого АЦП (конфігурація аналогових входів, частота і розрядність перетворення).

7. У розділі **Analog** встановити:

   у полі **Analog Input:**

   - ◉ **A0 A1 A2 A3.**

   Вказати:

   розрядність АЦП: **8 (0-255)** або **10 (0-1023)** розрядів:

   - **Units:** **0-255 / 0-1023**
   - **Clock:** **4 us**

Конфігурування можна зробити з **IDE PCWH** при створенні проекту:

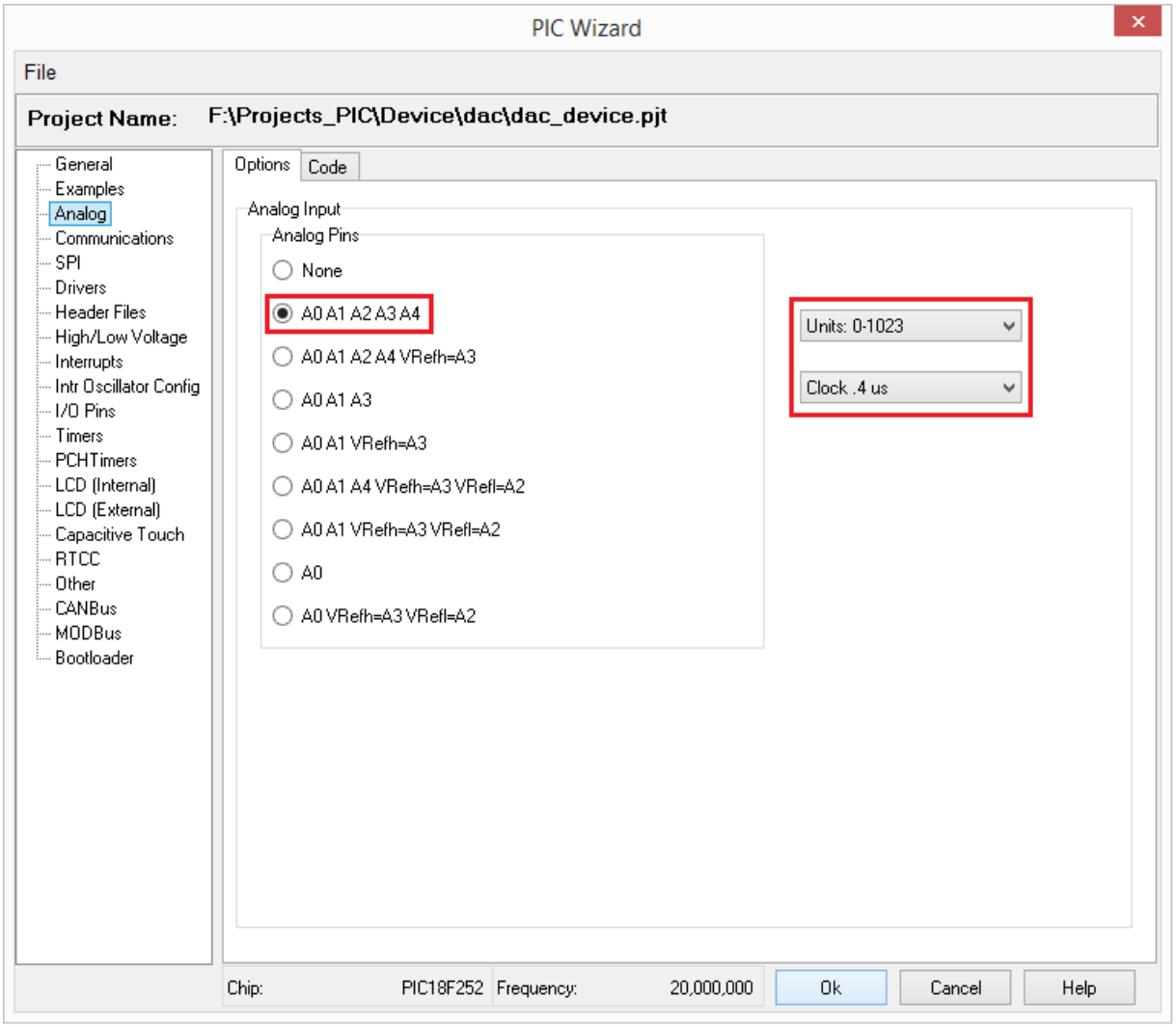

Рисунок 3.4.11 – Конфігурування АЦП із **IDE PCWH** при створенні проекту у розділі **Analog** майстра **PIC Wizard**

Буде згенерований і доданий програмний код у вихідний файл відповідно до параметрів на поточній вкладці.

Для перегляду програмного коду, слід вибрати вкладку **Code**.

8. Натиснути кнопку **OK**.

Буде згенерований наступний код (рис. 3.4.12):

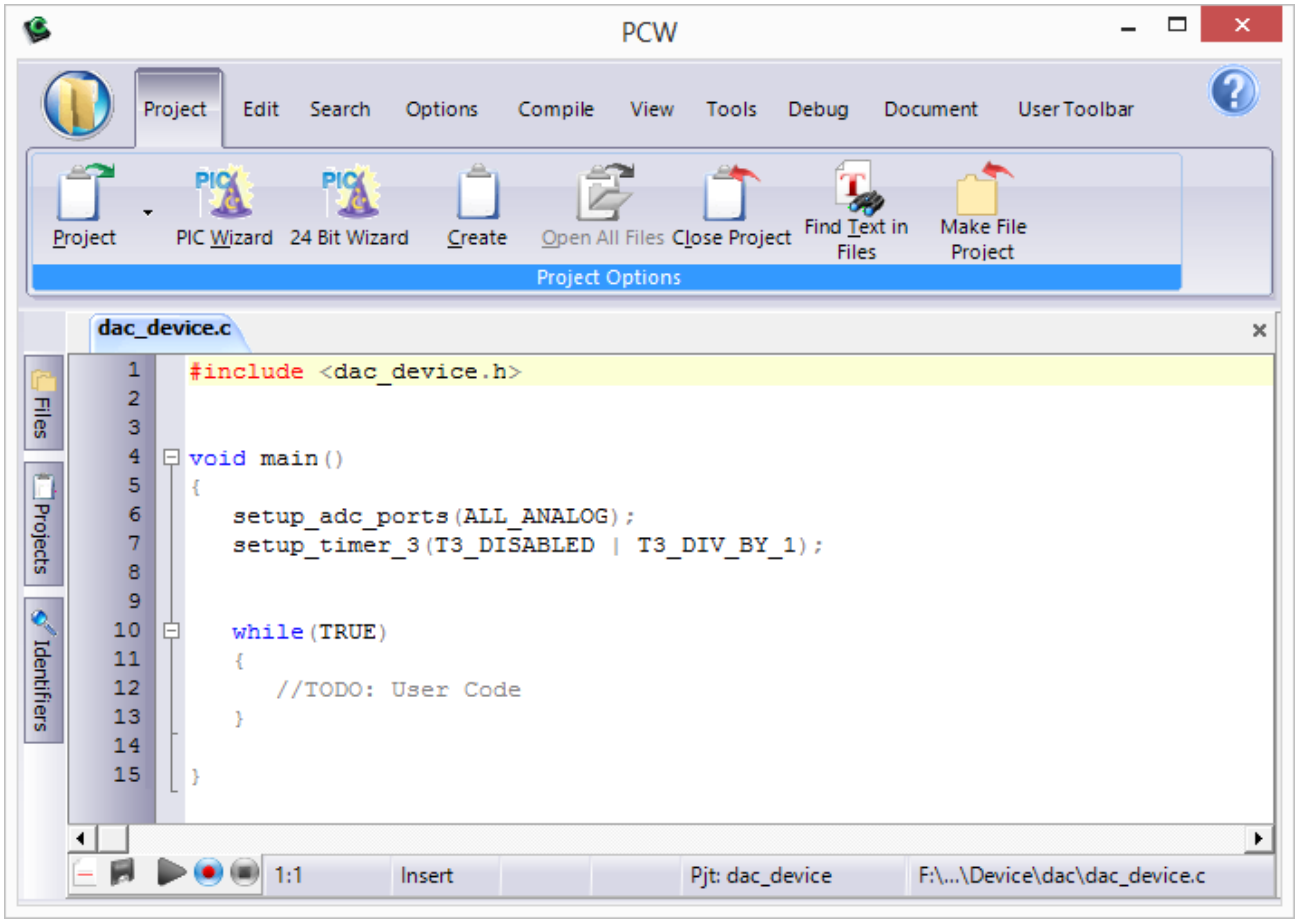

Рисунок 3.4.12 – Згенерований майстром **PIC Wizard** програмний код

Лістинг 3.4.1 – Приклад програмного коду, згенерованого майстром **PIC Wizard**

dac_device.c

```
#include <dac_device.h>
void main()
{
setup_adc_ports(ALL_ANALOG);
setup_timer_3(T3_DISABLED | T3_DIV_BY_1);\
while(TRUE)
{
//TODO: User Code }}
```

dac_device.h

```
#include <18F252.h>
#device adc=10

#FUSES NOWDT          //No Watch Dog Timer
#FUSES WDT128         //Watch Dog Timer uses 1:128 Postscale
#FUSES HS             //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for
PCD)
#FUSES NOBROWNOUT     //No brownout reset
#FUSES NOLVP          //No low voltage prgming, B3(PIC16)
//or B5(PIC18) used for I/O

#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8,stream=PORT1)
```

Проект готовий, можна приступати до написання програми.

ПОЯСНЕННЯ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ «DAC»

Цифро-аналоговий перетворювач MCP4921

MCP4921 пристрій являє собою одноканальний 12-бітний ЦАП, який використовує зовнішнє джерело опорної напруги. Цей пристрій забезпечує високу точність і низьке енергоспоживання і доступні в різних пакетах. Зв'язок з пристроєм здійснюється за допомогою простого послідовного інтерфейсу SPI. **MCP4921** пристрій є частиною сімейства, які використовують зовнішнє джерело опорної напруги (V_{REF}).

Ці пристрої забезпечують дуже високу точність і низький рівень шумів, і придатні для споживчих і промислових додатків. Низький рівень споживання енергії та малі варіанти пакета добре підходять для багатьох портативних і малопотужних додатків. Завдяки своїм компактним розмірам і малій потужності споживання, **ЦАП MCP4921** забезпечує значні переваги в просторі обмежених випадках, коли мінімальна потужність споживання має вирішальне значення.

Режими програмного забезпечення або апаратного виключення нададуть додаткову економію електроенергії, зменшення струму в режимі очікування до 0,5 мкА (тип.) при будь-якому виборі. Зв'язок з **MCP4921** здійснюється через 3-провідний SPI протокол.

Внутрішня структура **ЦАП MCP 4921** представлена на рис. 3.4.13.

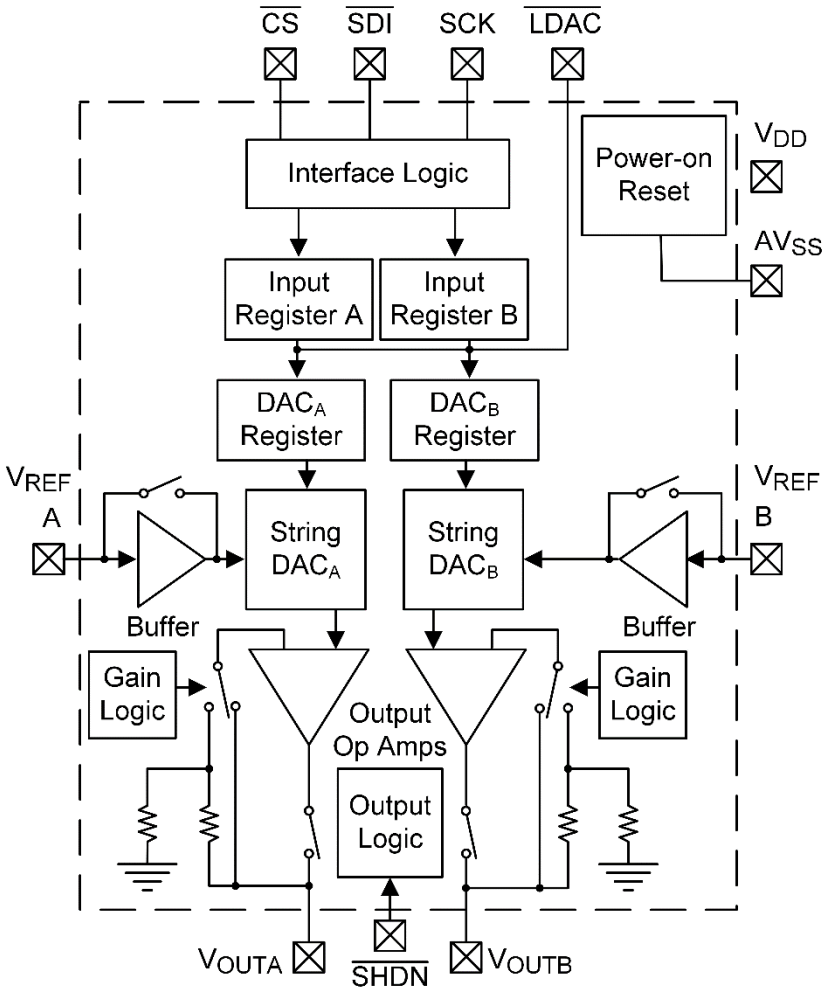

Рисунок 3.4.13 – Внутрішня структура **ЦАП MCP 4921**

Відмінні особливості:

- розрядність ЦАП: 12 розрядів;
- диференціальна нелінійність: ± 0,2 молодшого розряду (тип);
- інтегральна нелінійність: ± 2 молодших розряду (тип);
- виходи Rail-to-Rail;
- SPI-інтерфейс з частотою до 20 МГц;
- синхронні засувки даних на обох ЦАП;
- малий час встановлення: 4,5 мкс;
- вибір вихідного коефіцієнта посилення 1х або 2х;
- вхід для зовнішнього джерела опорної напруги V_{REF};
- діапазон напруги живлення: 2,7В ... 5,5В;

- кількість виводів: 8;
- тип входу: послідовний;
- температурний діапазон: –40 ° C ... + 125 ° C;
- корпуси: MSOP–8 (MS) і DIP–8 (P).

Розташування виводів **ЦАП MCP 4921** представлено на рис. 3.4.14.

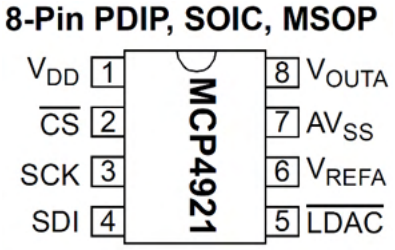

Рисунок 3.4.14 – ЦАП **MCP 4921**

Підключення **ЦАП** до мікроконтролера по інтерфейсу **SPI** представлено на рис. 3.4.15.

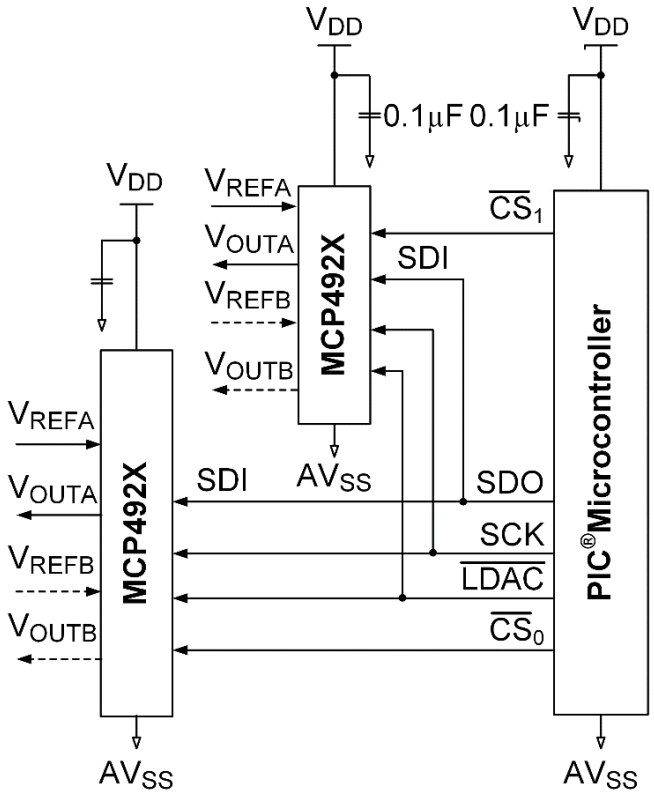

Рисунок 3.4.15 – Підключення **ЦАП** до мікроконтролера по інтерфейсу **SPI**

Таблиця 3.4.2 – Призначення виводів ЦАП **MCP 4921**

MCP4921 Pin No	**Symbol**	**Function**
1	V_{DD}	**Positive Power Supply Input (2.7V to 5.5V)** Споживана потужність живлення по відношенню до AV_{SS} може варіюватися від 2,7 до 5.5. Розв'язуючий конденсатор на V_{DD} рекомендується для досягнення максимальної продуктивності.
2	$\overline{CS}$	**Chip Select Input** Вхід вибору чіпу, який вимагає активний низький сигнал для включення послідовних годин і функцій даних
3	SCK	**Serial Clock Input** SPI сумісний послідовний ввід
4	SDI	**Serial Data Input** SPI сумісне послідовне введення даних
5	$\overline{LDAC}$	**Syncronization input used to transfer DAC settings from serial latches to the output latches** Засувка входу ЦАП. Передача введення регістрів засувки ЦАП при низькому логічному рівні
6	$VREF_A$	**DAC_A Voltage Input (AV_{SS} to V_{DD})** Аналоговий сигнал на цих виводах використовується для установки опорної напруги на рядок ЦАП. Вхідний сигнал може варіюватися в діапазоні від AV_{SS} до V_{DD}
7	AV_{SS}	**Analog ground** Аналогове заземлення
8	V_{OUTA}	**DAC_A Output** ЦАП підсилює сигнал на виводах в діапазоні AV_{SS}-V_{DD}

Приклади реалізації програми лабораторної роботи «DAC»

Конфігурування мікроконтролера для роботи із ЦАП по інтерфейсу SPI можна зробити з **IDE PCWH** при створенні проекту або у функції ініціалізації ввести рядок:

```
setup_spi(SPI_MASTER|SPI_L_TO_H|SPI_CLK_DIV_4);
```

Проте, написання драйвера інтерфейсу SPI для роботи у двобайтовому режимі є одним із завдань лабораторної роботи.

Приклад зчитування даних з АЦП з каналу 1 і передача даних у ЦАП:

```
float volt_in;                      //вихід потенціометра
float volt_out;                     //вихід ЦАП
int16 dac_hex;                      //вхід ЦАП

//====== послати значення у ЦАП
dac_hex = volt_in/DAC_CVANT;        //двійковий еквівалент напруги
SPI_send(dac_hex);                  //послати дані у ЦАП
```

Змінна **DAC_CVANT** є ціною розподілу шкали ЦАП і обчислюється як частка від ділення U_{REF} на розрядність АЦП, виражену у двійкових одиницях:

```
//====== опорна напруга = напруга живлення
const float REF_VDD = 5.00;
const float DAC_12 = 4095;

//====== розподіл шкали: 5В/шкала (розрядність ЦАП)
const float DAC_CVANT = REF_VDD/DAC_12;
```

Визначення двійкового значення величини `dac_hex` напруги $U_{ВИХ}$, що встановлюється:

```
dac_hex = volt_in/DAC_CVANT;
```

Відлік шкали перетворення проводиться відносно опорної напруги U_{REF}. Наприклад, якщо U_{REF} = 10В, а розрядність ЦАП = 8 розрядів (255 рівнів), то ціна (вага) одного розряду складе 10В/255 = 0.039В або 39 мВ.

Зазвичай U_{REF} встановлюють рівним напрузі живлення ЦАП, тобто:

$U_{ref} = v_{dd} = 5в.$

Примітка: після виведення значень на LCD необхідно ввести затримку часу індикації: `delay_ms(100);`

Послідовність побудови інтерфейсу SPI:

1) встановити виводи мікроконтролера, що беруть участь в роботі з SPI у режим передачі даних;
2) визначити інтерфейс:

```
#define CLK PIN_C3
#define CS PIN_C4
#define SDO PIN_C5
```

3) об`явити змінні для лічильника розрядів і затримки. Затримку прийняти рівною 25 мкс;
4) у пакет даних, що передається, вставити біт режиму `0x2000`;
5) перед початком передачі даних встановити сигнал CS у "0";
6) почати передачу пакета даних у циклі зі зсувом вліво. При передачі послідовно виставляти дані, затримки і строб відповідно до тимчасової діаграми на рис. 3.4.16.

```
//=== виставити біт
if(bit_test(val,c)){                //c - номер біту 0-15
       output_high(SDO);            //SDO = 1
}else{
       output_low(SDO);             //SDO = 0
}
```

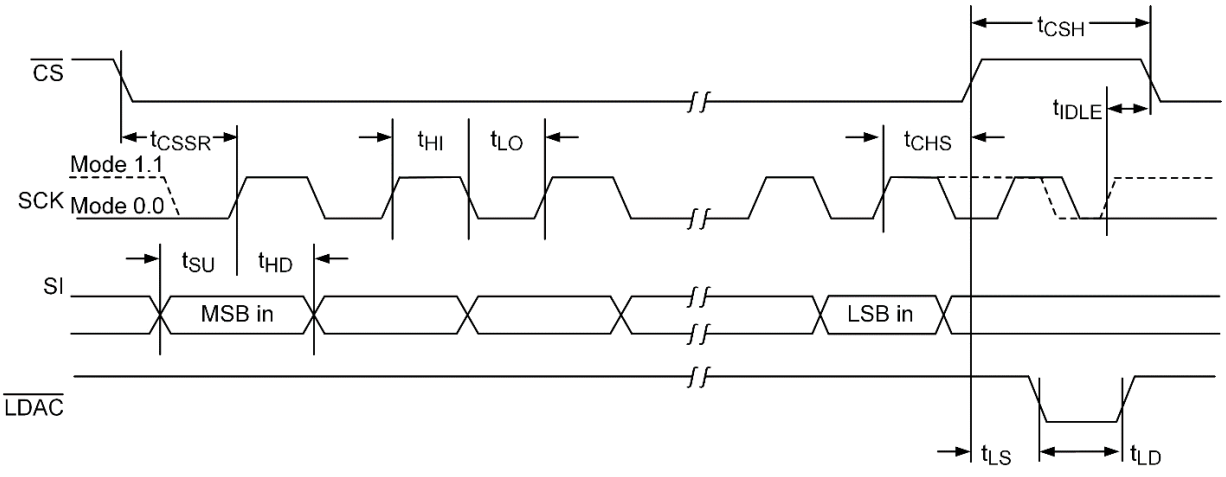

Рисунок 3.4.16 – Тимчасові діаграми роботи інтерфейсу SPI

7) після відправлення останнього біту встановити сигнал CS у "1" і виконати затримку.

На виході ЦАП з'явиться рівень напруги пропорційний заданому двійковому значенню.

Прототип функції:

```
void SPI_send(int16 val);
```

Процес пересилання пакета даних від мікроконтролера до ЦАП по інтерфейсу SPI представлений на рис. 3.4.17.

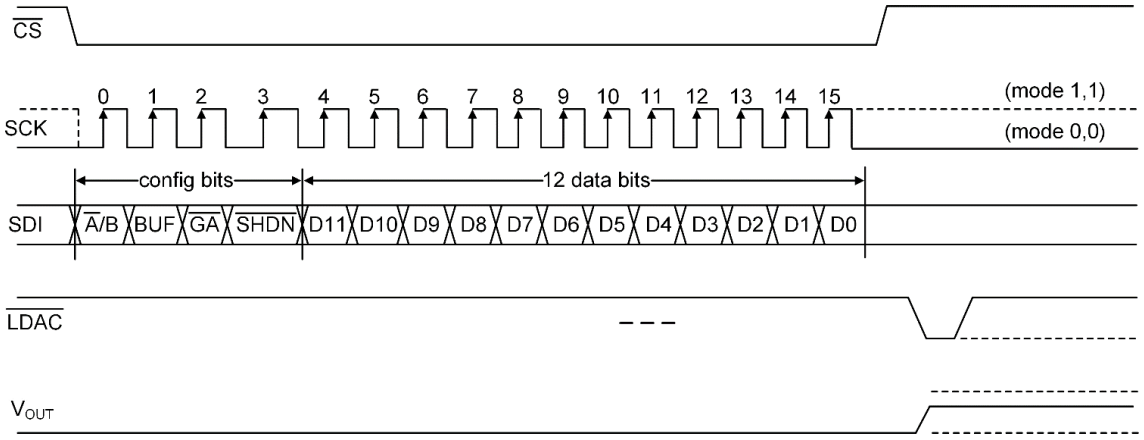

Рисунок 3.4.17 – Процес пересилання пакета даних від мікроконтролера до ЦАП по інтерфейсу SPI

Формат пакета даних:

Upper Half:							
W-x	W-x	W-x	W-0	W-x	W-x	W-x	W-x
$\overline{A}$/B	BUF	$\overline{GA}$	$\overline{SHDN}$	D11	D10	D9	D8
bit 15							bit 8

Lower Half:							
W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
D7	D6	D5	D4	D3	D2	D1	D0
bit 7							bit 0

bit 15 **A/B:** DAC_A or DAC_B Select bit

1 = Write to DAC_B

0 = Write to DAC_A

bit 14 **BUF:** V_{REF} Input Buffer Control bit

1 = Buffered

0 = Unbuffered

bit 13 **GA:** Output Gain Select bit

1 = 1x ($V_{OUT} = V_{REF} * D/4096$)

0 = 2x ($V_{OUT} = 2 * V_{REF} * D/4096$)

bit 12 **SHDN:** Output Power Down Control bit

1 = Output Power Down Control bit

0 = Output buffer disabled, Output is high impedance

bit 11-0 **D11:D0:** DAC Data bits

12 bit number "D" which sets the output value. Contains a value between 0 and 4095.

Таблиця 3.4.3 – Тимчасові характеристики роботи інтерфейсу SPI

Electrical Specifications: Unless otherwise indicated, V_{DD}= 2.7V – 5.5V, TA= -40 to +125°C. Typical values are at +25°C.						
Parameters	**Sym**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
Schmitt Trigger High-Level Input Voltage (All digital input pins)	V_{IH}	0.7 V_{DD}	—	—	V	
Schmitt Trigger Low-Level	V_{IL}	—	—	0.2 V_{DD}	V	

Electrical Specifications:

Unless otherwise indicated, V_{DD}= 2.7V – 5.5V, TA= -40 to +125°C.

Typical values are at +25°C.

Parameters	Sym	Min	Typ	Max	Units	Conditions
Input Voltage (All digital input pins)						
Hysteresis of Schmitt Trigger Inputs	V_{HYS}	—	0.05 V_{DD}	—		
Input Leakage Current	$I_{LEAKAGE}$	-1	—	1	µA	$\overline{SHDN}$ = $\overline{LDAC}$ = $\overline{CS}$ = SDI = SCK + V_{REF} = V_{DD} or AV_{SS}
Digital Pin Capacitance (All inputs/outputs)	C_{IN}, C_{OUT}	—	10	—	pF	V_{DD} = 5.0V, T_A = +25°C, f_{CLK} = 1 MHz
Clock Frequency	F_{CLK}	—	—	20	MHz	T_A = +25°C
Clock High Time	t_{HI}	15	—	—	ns	
Clock Low Time	t_{LO}	15	—	—	ns	
$\overline{CS}$ Fall to First Rising CLK Edge	t_{CSSR}	40	—	—	ns	Applies only when $\overline{CS}$ falls with CLK high.
Data Input Setup Time	t_{SU}	15	—	—	ns	
Data Input Hold Time	t_{HD}	10	—	—	ns	
SCK Rise to $\overline{CS}$ Rise Hold Time	t_{CHS}	15	—	—	ns	
$\overline{CS}$ High Time	t_{CSH}	15	—	—	ns	
$\overline{LDAC}$ Pulse Width	t_{LD}	100	—	—	ns	
$\overline{LDAC}$ Setup Time	t_{LS}	40	—	—	ns	
SCK Idle Time before $\overline{CS}$ Fall	t_{IDLE}	40	—	—	ns	

Приклад виконання лабораторної роботи «DAC» відповідно до варіанту завдання для лабораторної роботи «DAC»

Варіант завдання для виконання лабораторної роботи **«DAC»** представлений у таблиці 3.4.4.

Таблиця 3.4.4 – Варіант завдання для виконання лабораторної роботи **«DAC»**

Канал АЦП							
Потенціометр				**ЦАП**			
A0	A1	A2	A3	A0	A1	A2	A3
	+					+	

Принципова схема підключень для варіанту АПК

Принципова схема підключень відповідно варіанту завдання завдання (таб. 3.4.4) для варіанту **АПК**, для виконання лабораторної роботи **«DAC»** виглядає у такий спосіб (рис. 3.4.18):

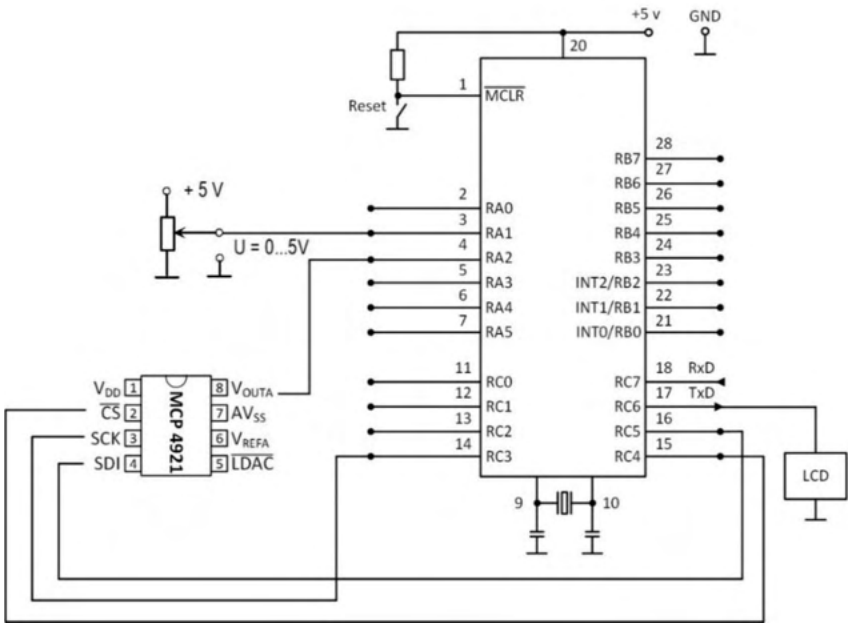

Рисунок 3.4.18 – Принципова схема підключень для лабораторної роботи **«DAC»** для варіанту **АПК**

Принципова схема підключень для варіанту «Proteus»

Принципова схема для варіанту **«Proteus»** відповідно варіанту завдання (таб. 3.4.4), для виконання лабораторної роботи **«DAC»** виглядає у такий спосіб (рис. 3.4.19):

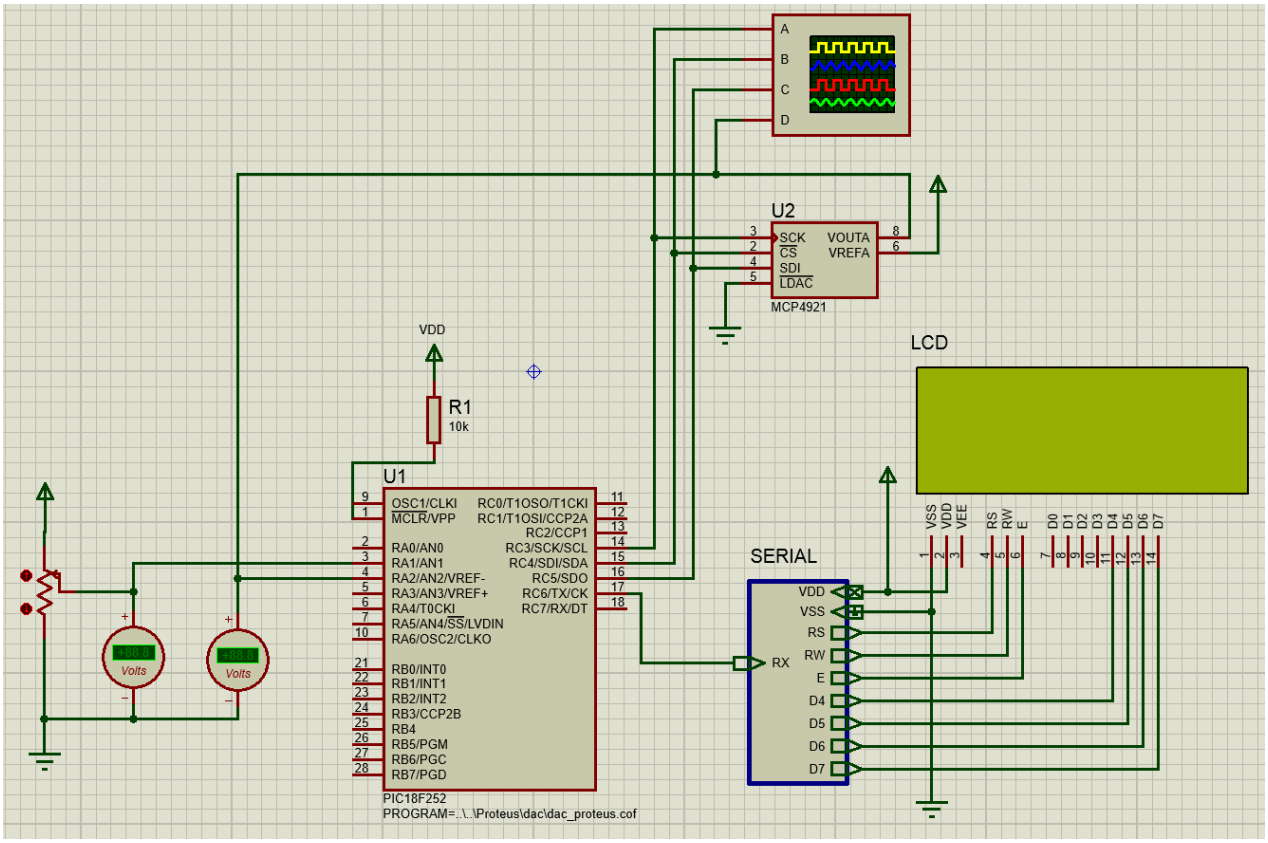

Рисунок 3.4.19 – Принципова схема підключень лабораторної роботи **«DAC»** для варіанту **«Proteus»**

Результат виконання лабораторної роботи «DAC» для варіанту «Proteus»

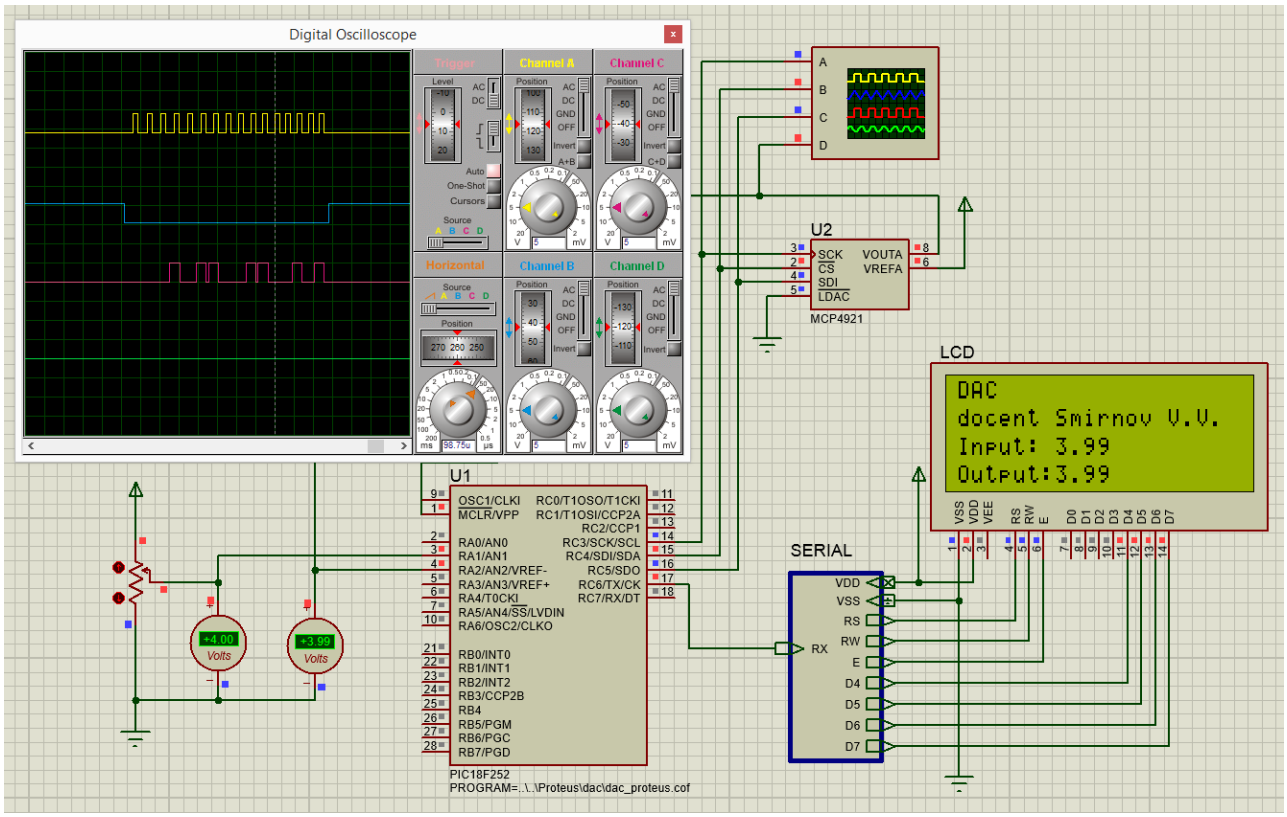

Рисунок 3.4.20 – Результат виконання лабораторної роботи **«DAC»** для варіанту **«Proteus»**

Контрольні питання

1) призначення ЦАП і методи перетворення;
2) статичні і динамічні характеристики ЦАП **MCP4921** (SPI);
3) призначення ліній інтерфейсу SPI;
4) структура пакета даних ЦАП **MCP 4921** (SPI);
5) як визначається величина двійкового еквівалента необхідної вихідної напруги ЦАП?

ТЕСТОВІ ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1. Коротка характеристика мікроконтролера PIC18F252:

a) периферійний інтерфейсний контролер;

b) універсальний мікропроцесор;

c) універсальний сигнальний процесор;

d) зв'язувальний контролер;

e) керуючий модуль.

2. Як управляти напрямком введення - виведення даних у мікроконтролері?

a) командою PUSH;

b) командою DIRECT;

c) командою DIRECTION;

d) командою TRIS;

e) командою POP.

3. Які команди здійснюють виведення інформації на LCD?

a) out_LCD;

b) send_LCD;

c) printf;

d) print_LCD;

e) plot.

4. Яким чином здійснюється запис програми у мікроконтролер?

a) ручкою;

b) лістингом;

c) файлом;

d) програматором;

e) записувачем.

5. У яку пам'ять завантажується код програми?

a) у внутрішню;

b) В EEPROM;

c) у пам'ять даних;

d) у пам'ять програм;

e) у пам'ять коду.

6. Зовнішні переривання мікроконтролера PIC18F252:

a) CCP;

b) USART;

c) таймер;

d) ШІМ;

e) ЦАП.

7. Як управляти (дозволяти, забороняти) перериваннями в мікроконтролері?

a) вмиканням живлення;

b) вимиканням живлення;

c) командою YES;

d) командою NOT;

e) командою ENABLE/DISABLE.

8. Чому в перериванні порту RS-232 необхідно очистити прийомний буфер?

a) щоб звільнити місце;

b) для запису відповіді;

c) щоб зняти переривання;

d) щоб переривання не скинулося;

e) щоб прочитати символ.

9. Якими по відношенню контролера бувають переривання?

a) вищерозміщені;

b) центральні;

c) правобічні;

d) лівосторонні;

e) зовнішні.

10. Призначення АЦП.

a) для перемикання каналів;

b) для швидкої реакції на переривання;

c) перетворення цифрового значення в аналоговий;

d) захист від зависання;

e) перетворення аналогового сигналу в цифровий.

11. Методи перетворення АЦП послідовного наближення?

a) прямий;

b) зворотний;

c) послідовний;

d) паралельного наближення;

e) послідовного наближення.

12. Конфігурування мікроконтролера для роботи з АЦП:

a) командою SET_float;

b) командою RESET;

c) командою SET_DIGIT;

d) командою RESET_DIGIT;

e) командою SET_ANALOG.

13. Вплив розрядності АЦП на точність перетворення:

a) більше – краще;

b) менше – краще;

c) більше – гірше;

d) не впливає;

e) впливає незначно.

14. Опорна напруга Vref і його вплив на параметри АЦП:

a) не потрібно;

b) не впливає;

c) впливає незначно;

d) визначає шкалу перетворення;

e) збільшує перешкодостійкість.

15. Як визначається ціна розподілу (вага) шкали АЦП?

a) зліва на право;

b) з права на ліво;

c) множенням Vref на розрядність;

d) поділом Vref на розрядність;

e) вирахуванням Vref з VDD.

16. Призначення ЦАП:

a) цифро-алфавітне перетворення;

b) цифро-послідовне перетворення;

c) цифро-паралельне перетворення;

d) цифро -аналогове перетворення;

e) цифро-аналогічне перетворення.

17. Інтерфейс SPI:

a) послідовно-паралельний;

b) тільки паралельний;

c) тільки послідовний;

d) режим паралельності задається;

e) паралельно-послідовний.

18. Призначення лінії CS інтерфейсу SPI:

a) скидання даних;

b) передача даних;

c) прийом даних;

d) очікування;

e) вибір чипа.

19. Призначення лінії SDI інтерфейсу SPI:

a) скидання даних;

b) передача даних;

c) прийм даних;

d) вмикання;

e) вимикання.

20. Призначення лінії SDO інтерфейсу SPI:

a) скидання даних;

b) передача даних;

c) приймання даних;

d) вмикання;

e) вимикання.

21. Зовнішні переривання мікроконтролера PIC18F252:

a) CCP;

b) USART;

c) таймер;

d) ШІМ;

e) ЦАП;

22. Чому в перериванні порту RS-232 необхідно очистити прийомний буфер?

a) Щоб звільнити місце

b) Для запису відповіді

c) Щоб зняти переривання

d) Щоб переривання не скинулося

e) Щоб прочитати символ

23. Архітектура мікроконтролера PIC18F252:

a) Фон – Неймана;

b) Неймана – Пірсона;

c) Гарварда;

d) Гарвардська;

e) Неймана – Гарварда;

24. Кількість виводів у мікроконтролера PIC18F252:

a) 16;

b) 18;

c) 22;

d) 28;

e) 34.

25. Об’єм пам'яті даних у мікроконтролера PIC18F252:

a) 128байт;

b) 256байт;

c) 1,5К;

d) 2К;

e) 4К.

26. Об’єм пам'яті програм у мікроконтролера PIC18F252:

a) 1К;

b) 2К;

c) 4К;

d) 16К;

e) 32К.

27. Об’єм пам'яті EEPROM у мікроконтролера PIC18F252:

a) 128байт;

b) 256байт;

c) 1k;

d) 2k;

e) 4k.

28. Кількість паралельних портів у мікроконтролера PIC18F252:

a) 0;

b) 1;

c) 2;

d) 4;

e) 8.

29. Кількість портів введення - вивдення у мікроконтролера PIC18F252:

a) 1;

b) 2;

c) 3;

d) 4;

e) 6.

30. Кількість USART – модулів у мікроконтролера PIC18F252:

a) 1;

b) 2;

c) 3;

d) 4;

e) 5.

31. Кількість USB – модулів у мікроконтролера PIC18F252:

a) 1;

b) 2;

c) 3;

d) 4;

e) 0.

32. Кількість SPI – модулів у мікроконтролера PIC18F252:

a) 1;

b) 2;

c) 3;

d) 4;

e) 5.

33. Кількість каналів ЦАП MCP4921:

a) 1;

b) 2;

c) 3;

d) 4;

e) 5.

34. Кількість каналів АЦП у мікроконтролера PIC18F252:

a) 1;

b) 2;

c) 3;

d) 4;

e) 5.

35. Архітектура мікроконтролера PIC18F252:

a) Фон – Неймана;

b) Неймана – Пірсона;

c) Гарварда;

d) Гарвардська;

e) Неймана – Гарварда.

36. Кількість зовнішніх переривань типу INT у мікроконтролері PIC18F252:

a) 1;

b) 2;

c) 3;

d) 4;

e) 5.

37. Зовнішні переривання типу INT у мікроконтролері PIC18F252 перебувають на порту:

a) A;

b) B;

c) C;

d) D;

e) E.

38. Розрядність АЦП у мікроконтролері PIC18F252 можна встановити:

a) 4 розрядів;

b) 5 розрядів;

c) 6 розрядів;

d) 7 розрядів;

e) 8 розрядів.

39. Кількість виводів у мікроконтролера PIC18F252:

a) 16;

b) 18;

c) 22;

d) 28;

e) 34.

40. По інтерфейсу SPI у мікроконтролері PIC18F252 можна в одній посилці передати максимум байт:

a) 2;

b) 3;

c) 4;

d) 8;

e) 16.

41. Функція value = getc() виконує:

a) передачу символу;

b) приймання символу;

c) передачу рядка;

d) приймання рядка;

e) приймання блоку.

42. Функція disable_interrupts (*level*) виконує:

a) дозволити переривання;

b) продовжити переривання;

c) заборонити переривання;

d) призупинити переривання;

e) скасувати переривання.

43. Функція output_x (*value*) виконує:

a) запис байта;

b) запис біта;

c) приймання байта;

d) приймання біта;

e) скидання байта.

44. Функція output_high (*pin*) виконує:

a) установка біта в «2»;

b) установка біта в «1»;

c) установка біта в «3»;

d) установка біта в «0»;

e) установка байта.

45. Функція output_low (*pin*) виконує:

a) установка біта в «2»;

b) установка біта в «1»;

c) установка біта в «3»;

d) установка біта в «0»;

e) установка байта.

46. Функція value = read_adc() виконує:

a) запис значення в АЦП;

b) запис біта в АЦП;

c) запис рядка в АЦП;

d) приймання біта з АЦП;

e) читання значення з АЦП.

47. Функція value = spi_read (*data*) виконує:

a) запис байта;

b) запис біта;

c) запис рядка;

d) приймання біта;

e) приймання байта.

48. Функція spi_write (*value*) виконує:

a) запис байта;

b) запис біта;

c) запис рядка;

d) приймання біта;

e) приймання байта.

49. Функція lcd_putc(int8 ch) виконує:

a) запис байта;

b) запис біта;

c) запис рядка;

d) приймання біта;

e) приймання байта.

50. Функція lcd_init() виконує:

a) скидання lcd;

b) ініціалізацію lcd;

c) завантаження даних в lcd;

d) запис біта в lcd;

e) приймання байта з lcd.

51. Функція lcd_goto(int8 line,int8 pos) виконує:

a) скидання lcd;

b) ініціалізацію lcd;

c) завантаження даних в lcd;

d) запис біта в lcd;

e) позиціонування курсору lcd.

52. Функція lcd_putc(int8 line, int8 pos, int8 ch) виконує:

a) запис даних в lcd у координатах;

b) ініціалізацію lcd;

c) завантаження даних в lcd;

d) запис біта в lcd;

e) позиціонування курсору lcd.

53. Функція lcd_clear_line(int8 line) виконує:

a) запис даних в lcd у координатах;

b) очищення рядка в lcd;

c) завантаження даних в lcd;

d) запис біта в lcd;

e) позиціонування курсору lcd.

54. Функція lcd_clear() виконує:

a) скидання lcd;

b) ініціалізацію lcd;

c) завантаження даних в lcd;

d) запис біта в lcd;

e) позиціонування курсору lcd.

55. Розрядність типу long:

a) 4;

b) 8;

c) 16;

d) 32;

e) 64.

56. Розрядність типу float:

a) 4;

b) 8;

c) 16;

d) 32;

e) 64.

57. Розрядність типу double:

a) 4;

b) 8;

c) 16;

d) 32;

e) 64.

58. Розрядність типу char:

a) 4;

b) 8;

c) 16;

d) 32;

e) 64.

59. Правильний варіант приведення типів до byte b:

a) b = (int)i;

b) b = (byte)i;

c) b = int i;

d) b = int(i);

e) b = byte(i).

60. byte b = 3. Виникне помилка часу виконання:

a) b *= 2;

b) b *= 4;

c) b *= 8;

d) b *= 100;

e) b += 200.

61. Операція инкремента:

a) ++;

b) --;

c) +=;

d) *=;

e) /=.

62. Операція декремента:

a) ++;

b) --;

c) +=;

d) *=;

e) /=.

63. Операція додавання:

a) ++;

b) --;

c) +=;

d) *=;

e) /=.

64. Операція розподілу:

a) ++;

b) --;

c) +=;

d) *=;

e) /=.

65. Операція множення:

a) ++

b) --

c) +=

d) *=

e) /=

Варіанти відповідей

1	a	**23**	d	**45**	d
2	d	**24**	d	**46**	e
3	c	**25**	c	**47**	e
4	d	**26**	e	**48**	a
5	d	**27**	b	**49**	a
6	b	**28**	a	**50**	b
7	e	**29**	c	**51**	e
8	e	**30**	a	**52**	a
9	e	**31**	e	**53**	b
10	e	**32**	a	**54**	a
11	e	**33**	a	**55**	e
12	e	**34**	e	**56**	d
13	a	**35**	d	**57**	e
14	d	**36**	c	**58**	b
15	d	**37**	b	**59**	b
16	d	**38**	e	**60**	d
17	c	**39**	d	**61**	a
18	e	**40**	a	**62**	b
19	c	**41**	b	**63**	c
20	b	**42**	c	**64**	e
21	b	**43**	a	**65**	d
22	e	**44**	b		

ТЕМИ РЕФЕРАТІВ

Мікроконтролери.

1) дати короткий огляд основних характеристик мікроконтролерів;
2) дати опис середовища розробки програм PSW CSS;
3) дати короткі рекомендації по написанню програми, компіляції та програмування контролера через програматор;
4) висновки.

АЦП мікроконтролера.

1) дати короткий огляд типів і видів АЦП;
2) дати опис принципів роботи квантування за рівнем, дискретизації за часом та залежності якості перетворення від розрядності АЦП;
3) дати короткі рекомендації по використанню переривання АЦП та програмування контролера для роботи з АЦП;
4) висновки.

Датчики.

1) дати короткий огляд сучасних датчиків фізичних величин;
2) дати опис принципів роботи датчиків:

- датчики тиску;
- датчики температури;
- датчики переміщення;
- датчики вологості;
- тензометричні датчики та енкодери;

3) дати короткі рекомендації по застосуванню датчиків;
4) висновки.

ЦАП. Інтерфейс SPI.

1) дати короткий огляд типів і видів ЦАП;
2) дати опис принципів роботи квантування за рівнем, дискретизації за часом та залежності якості перетворення від розрядності ЦАП;
3) дати короткі рекомендації по програмуванню контролера для роботи з ЦАП по інтерфейсу SPI;
4) висновки.

Семисегментний LED – дисплей.

1) дати короткий огляд семисегментних LED-дисплеїв;
2) дати опис принципів роботи статичної та динамічної індикації;
3) дати короткі рекомендації по схемі підключення LED-дисплея до контролера, кодування сегментів та програмування контролера для роботи з LED-дисплеєм;
4) висновки.

Модуль CCP.

1) дати короткий огляд модуля CCP;
2) дати опис принципів роботи модуля **capture**, модуля **compare** та модуля **PWM**;
3) дати короткі рекомендації вимірюванню тимчасових інтервалів, тривалості імпульсу, скважності та керування навантаженням;
4) висновки.

EEPROM – пам'ять.

1) дати короткий огляд EEPROM – пам'яті;
2) дати опис принципів роботи інтерфейсу I^2C, архітектури Master-Slave та тимчасових діаграм роботи інтерфейсу I^2C;
3) дати короткі рекомендації по програмуванню контролера для роботи з інтерфейсом I^2C;
4) висновки.

Зовнішній таймер з інтерфейсом I^2C.

1) дати короткий огляд зовнішнього таймеру з інтерфейсом I^2C;
2) дати опис принципів роботи інтерфейсу I^2C, адресації таймера, та доступу до модулів таймера;
3) дати короткі рекомендації по програмуванню контролера для роботи з таймером;
4) висновки.

LCD – дисплей.

1) дати короткий огляд LCD-дисплеїв;
2) дати опис принципів LCD-дисплея та інтерфейсу LCD-дисплея;
3) дати короткі рекомендації по програмуванню контролера для роботи з LCD;
4) висновки.

ГЛОСАРІЙ

Acquisition Time - час заряду конденсатора АЦП до рівня вхідного сигналу. На цей параметр впливає опір джерела сигналу. Чим більше опір, тим більший час необхідний для заряду конденсатора. При установці біту GO/DONE аналоговий сигнал відключається і починається процес аналого-цифрового перетворення.

ADC (Analog To Digital Converter) - АЦП (аналогово-цифровий перетворювач). Цей пристрій конструктивно може бути окремою мікросхемою або вбудованим модулем мікроконтролерів фірми Microchip. Для окремо стоячих АЦП розрядність становить 12-13 біт. Можливо диференціальне включення. Для вбудованого модуля розрядність становить 10 біт.

AUSART - Addressable Universal Synchronous Asynchronous Receiver Transmitter - адресований USART. Відмінність полягає в тому, що є можливість використовувати 9-бітний протокол передачі даних для визначення адреси/даних.

Bank - спосіб адресації пам'яті. Так як у сімействі PIC16 є 7 біт для прямої адресації пам'яті, то можна працювати з 128 байтами пам'яті (включаючи спец. регістри). Для використання більшої кількості пам'яті, весь об`єм пам'яті ділиться на безперервні банки, кожен по 128 байт. Для вибору визначеного банку, необхідно встановити біти регістрів RP0:RP1, тобто можна працювати з 4 банками пам'яті. У сімействі PIC18 банкова система пам'яті замінена на більш досконалу.

Baud - бод (одиниця виміру) - дане поняття служить для обчислення кількості одиниць інформації (біт), переданих по послідовному каналу за інтервал часу в 1 секунду, тобто біт в секунду (bps).

BCD - Binary Coded Decimal - двійково-кодована десяткова форма обчислення. У шістнадцятковій формі обчислення кожна тетрада (послідовність з 4 біт) може приймати значення від 0000 до 1111 (від 0 до 15). Дана форма зручна для мікроконтролера, але не зручна для людини. Тому прийнято декодувати шістнадцяткову форму в двійково-десяткову. При цьому кожна тетрада може

приймати значення від 0000 до 1001 (від 0 до 9). У цьому випадку одним байтом можна записати число від 0 до 99.

BOR - Brown-out Reset - даний модуль є складовою частиною більшості мікроконтролерів фірми Microchip. Цей модуль переводить мікроконтролер у стан RESET, якщо живлення знизилося нижче визначеної межі на визначений часовий інтервал. У більшості мікроконтролерів PIC16 дана межа встановлена апаратно. У мікроконтролерах PIC18 цю межу можна задавати на стадії програмування пристрою у слові конфігурації.

Brown-out - дане словосполучення означає, що сталося зниження напруги нижче визначеного робочого значення. Даний стан може виникати під час «осідання» напруги під час перемикання/включення потужного навантаження.

Capture - функція модуля CCP. У випадку включення модуля CCP у цей режим відбувається перезапис значення TMR1 у визначений регістр при виникненні деякої події. Такою подією може бути прихід фронту/зрізу імпульсу, прихід деякого числа імпульсів на ніжку модуля CCP.

CCP - Capture, Compare, Pulse Width Modulation (PWM) - захоплення, порівняння, широтно-імпульсна модуляція (ШІМ). Даний модуль є складовою частиною більшості мікроконтролерів фірми Microchip. Він служить для визначення довжини імпульсу, а також для реалізації широтно-імпульсної модуляції.

CERDIP - Ceramic dual in-line package - керамічний корпус з дворядним розташуванням ніжок. Відстань між ніжками становить 2.54мм.

CLC - Configurable Logic Cell - модуль, який дозволяє конфігурувати на апаратному рівні логічні елементи такі як «І», «АБО», «АБО, що виключає», різні тригери і т.д.

Common RAM - це область ОЗП, що має одне і теж розташування у всіх банках пам'яті. Цей вид ОЗП характерний для мікропроцесорів сімейства PIC16. «Загальний ОЗП» розташовується за адресами 70h-7Fh. Загальна область корисна для збереження необхідних змінних при перемиканні контексту, тому що не вимагає перемикання банків пам'яті. Ця частина пам'яті зазвичай

використовується для збереження вмісту акумулятора і деяких спеціальних регістрів.

Compare - порівняння - функція модуля CCP, сенс якої в тому, що пристрій здійснює визначену дію при збігу значень регістра таймера зі значенням у регістрі порівняння.

Configuration Word - слово конфігурації - спеціальний вид пам'яті, в якому розташовується інформація про різні режими роботи мікроконтролера. Цей блок пам'яті записується під час програмування мікроконтролера за допомогою програматора. В даному блоці розташовується інформація про тип генератора тактових імпульсів, про захист кристала від зчитування вмісту пам'яті програм / пам'яті даних EEPROM, про дозвіл роботи WDT і інших налаштуваннях. У різних мікроконтролерах різні слова конфігурації.

Conversion Time - час необхідний АЦП перетворення значення напруги на конденсаторі у цифрове значення. Даний час залежить від тактової частоти процесора, і обраного переддільника АЦП.

CPU - Central Processing Unit - центральний процесор

CTMU - CHARGE TIME MEASUREMENT UNIT - модуль для вимірювання заряду ємності і самої ємності. Основне застосування - це обробка сигналів ємнісних сенсорних панелей.

CWG - Complementary Waveform Generator - модуль який генерує комплементарний ШІМ сигнал з мертвим часом.

DAC - Digital-Analog Converter – цифро-аналоговий перетворювач (ЦАП). Даний модуль не є вбудованим модулем мікроконтролерів фірми Microchip. Він випускається у вигляді мікросхеми, що стоїть окремо. Однак цифро-аналогове перетворення можна виконати за допомогою модуля CCP, що працює у режимі PWM (ШІМ)

DIP - Dual In Package - корпус мікросхеми з дворядним розташуванням контактів.

Direct Addressing - пряма адресація. У цьому випадку адреса пам'яті знаходиться у команді/інструкції. Наприклад, `movwf 0x07`

DMA - Direct Memory Access - модуль для зчитування і запису RAM без задіяння процесора. Використовується для передачі і прийому пакетів даних через інтерфейси передачі даних, для збору даних від АЦП і т.д.

DSP - Digital Signal Processor - цифровий сигнальний процесор. Застосовуються для цифрової обробки сигналу, мають високу продуктивність ядра.

DCS - Digital Signal Controller - цифровий сигнальний контролер. Цей новий вид контролерів зараз починає випускати фірма Microchip під назвою dsPIC30F. Дані контролери з'єднують в одному ядрі функції властиві DSP і MCU.

ECAN - Enhanced Control Area Network - модуль для послідовної передачі даних між іншими CAN пристроями і мікроконтролерами. CAN надійніший ніж UART, так як має апаратну перевірку помилок передачі даних. В основному застосовується в автомобільній промисловості і для автоматизації на підприємствах.

ECCP - Enhanced Capture/Compare/PWM - покращений модуль CCP. Відрізняється від стандартного наявністю додаткових можливостей. Наприклад, даний модуль має можливість працювати з 2 або 4 вихідними модулями, змінювати полярність вихідного імпульсу, забезпечувати «мертву зону» для запобігання «наскрізних» струмів, забезпечувати екстрене відключення, автоматичне відключення і перезавантаження.

EEPROM - Electrically Erasable Programmable Read Only Memory - незалежний вид пам'яті. Дана пам'ять є складовою частиною більшості мікроконтролерів фірми Microchip. Однак її об`єм порівняно невеликий. Для мікроконтролерів сімейства PIC16 «типовим» значенням є 128 або 256 байт пам'яті EEPROM. Пам'яті такого об`єму цілком достатньо для зберігання калібрувальних констант, установочних даних та інших налаштувань користувача. Також дана пам'ять може служити «резервною областю», куди мікроконтролер може поміщати значення змінних у випадку критичного зниження напруги живлення.

EEPROM пам'ять також випускається у вигляді мікросхеми, що стоїть окремо з послідовним доступом (SPI або I^2C). У такому виконанні об`єм пам'яті може становити до 512Кбіт.

EPROM - Electrically Programmable Read Only Memory - електрично програмована постійна пам'ять. Пристрої даного типу пам'яті можна програмувати прямо в схемі. Стирання пам'яті проводиться опроміненням ультрафіолетом.

EMA - External Memory Addressing - модуль, що дозволяє мікроконтролеру виконувати програму, що знаходиться у зовнішній пам'яті.

EMI - Electromagnetic Interference - електромагнітна перешкода.

EXTRC - External Resistor-Capacitor (RC) - зовнішній ланцюг з резистора і конденсатора. У багатьох мікроконтролерах фірми Microchip є можливість підключення такого ланцюжка як генератор тактових імпульсів. Ця схема відрізняється дуже низькою ціною, але її стабільність набагато нижче, ніж у генератора на кварцовому резонаторі. У багатьох мікроконтролерах фірми Microchip є вбудований RC генератор - INTRC. Він калібрується на заводі-виробника і його точність становить 1%.

Flash memory - незалежний вид пам'яті. У процесорах фірми Microchip даний вид пам'яті використовується в якості пам'яті програм. Її можна програмувати і стирати прямо в схемі. Технологія даної пам'яті по функціональності майже еквівалентна EEPROM.

Fosc - частота тактового генератора. Крім свого «основного» значення - задавати швидкість роботи мікроконтролера, даний параметр відіграє важливу роль у багатьох вбудованих модулях мікроконтролерів фірми Microchip. Тому будьте уважні при зміні тактової частоти процесора!

GPIO - General Purpose Input/Output - введення/виведення загального призначення. Характерно для мікроконтролерів PIC12Fxxx.

GPR - General Purpose Register - регістр загального призначення. Частина пам'яті даних (один регістр), призначена для збереження в ній значень змінних.

HS - High Speed - висока швидкість. Один з режимів тактового генератора, в якому генератор налаштовується на роботу на високій частоті. Використовується для роботи з кварцами на частоті від 4 до 20 МГц. Встановлюється в слові конфігурації при програмуванні мікроконтролера.

IC - Integrated Circuit - інтегральна мікросхема.

ICD - In-Circuit Debugger - внутрішньосхемний відладчик. Вбудований модуль мікроконтролерів Microchip серій PIC12Fxx, Pic16F87x, Pic16F87xA, PIC18Fxxx. Наявність даного модуля дозволяє за допомогою спеціалізованого пристрою MPLAB ICD2 програмувати мікроконтролер і налагоджувати його внутрішньохемно, тобто мати доступ до всіх робочих регістрів, периферійним модулям і т.д.

ICSP - In-Circuit Serial Programming – внутрішньосхемне послідовне програмування. Для програмування мікроконтролера з цього протоколу використовується всього лише три виводи мікроконтролера + живлення. Завдяки наявності даного модуля програмування мікроконтролера можливо після установки на плату, що значно підвищує продуктивність при масовому виробництві.

IEEE - Institute of Electrical and Electronics Engineers - інститут інженерів з електротехніки та електроніки

IIC (I^2C) - Inter-Integrated Circuit - взаємно-інтегрований ланцюг, розроблений фірмою Philips. Один з режимів модуля SSP, що є вбудованим модулем мікроконтролерів фірми Microchip. Це двопровідний інтерфейс обміну даними між різними мікросхемами. Фірма Microchip випускає широкий спектр мікросхем, що працюють по протоколу I^2C.

IMC - інтегральні мікросхеми.

Indirect Addressing - непряма адресація. У цьому випадку адреса пам'яті не міститься в інструкції. Інструкція оперує INDF адресою, що представляє собою відображення адреси пам'яті у регістрі FSR. При виконанні команди дані беруться з комірки пам'яті, адреса якої вказується регістром FSR.

Instruction cycle - командний цикл - події, які відбуваються при виконанні команди. Основні етапи: Декодування, Читання, Виконання та Запис. Не всі етапи проходять всі команди. Один командний цикл Tcy виконується за чотири Tosc зовнішніх такти. Тобто кількість операцій за секунду дорівнює тактовій частоті, розділеної на чотири.

Interrupt - переривання - сигнал процесору, який направляє виконання програми по вектору переривання. Для мікроконтролерів сімейства PIC16 і PIC12Fxxx це адреса 0x0004H в пам'яті програм. Для мікроконтролерів фірми Microchip сімейства PIC18 існує два види переривань - переривання високого рівня і переривання низького рівня. Для цих переривань адреси рівні 0x0008H і 0x0018H відповідно. Перед зміною потоку виконання команди стан лічильника програми (Program Counter) заноситься в апаратний стек, і після обробки переривання виконання програми відновиться з цього ж місця. Під час виконання переривань користувач повинен подбати про збереження значення робочих регістрів.

IntRC - Internal Resistor-Capacitor (RC) - внутрішній ланцюг резистор-конденсатор. Багато мікроконтролерів фірми Microchip мають режим генератора, запуск якого здійснюється від внутрішнього RC-ланцюга. До таких мікроконтролерів відносяться практично всі пристрої, виконані за технологією NanoWatt. У таких пристроях внутрішній генератор калібрується на заводі і похибка не перевищує 1%. У даному режимі не потрібне підключення зовнішніх елементів, що дозволяє зменшити вартість виробу і звільнити ніжки, до яких раніше підключався кварцовий генератор або зовнішній RC ланцюжок. Встановлюється у слові конфігурації при програмуванні мікроконтролера.

LIN - Local Interconnect Network - протокол передачі даних по однопроводній шині.

LP - один з режимів генератора. Використовується для низькочастотних операцій в режимі зниженого енергоспоживання. Тактова частота - до 200 КГц. Встановлюється у слові конфігурації при програмуванні мікроконтролера.

LCD - Liquid Crystal Display - рідкокристалічний індикатор (РКІ).

LED - Light Emitting Diode - світлодіод

LSb - Least Significant Bit - найменш значущий біт.

LSB - Least Significant Byte - найменш значущий байт.

MI2C (MI2C) - Master Inter-Integrated Circuit - I^2C з апаратною підтримкою пристрою типу Master для реалізації топології мережі Multi-Master (в мережі присутні більше одного ведучого пристрою).

MOSFET - Metal Oxide Semiconductor Field Effect Transistor - польовий транзистор з МОН (метал-оксид-напівпровідник) структурою затвора. Даний вид пристроїв застосовується в якості ключового елемента в перетворювачах частоти, DC/DC перетворювачів, а також в пристроях управління двигунами. Фірма Microchip випускає даний вид транзисторів. Повний перелік знаходиться на сайті Microchip:
http://www.microchip.com/1010/pline/analog/anicateg/power/mosfet/index.htm

Motor control Kernel - апаратно-програмна реалізація управління двигуном. Включає в себе модулі ШІМ і програмне ядро, що дозволяє просто і зрозуміло програмувати ці модулі.

MSb - Most Significant bit - найбільш значущий біт.

MSB - Most Significant Byte - найбільш значущий байт.

NCO - NUMERICALLY CONTROLLED OSCILLATOR - модуль, що генерує сигнали прямокутної форми з малим відхиленням по частоті і з високою роздільною здатністю.

Op Amp - Operational Amplifier - операційний підсилювач.

OST - Oscillator Start-up Timer - таймер, який відповідає за надійний старт кварцового резонатора. При включенні живлення таймер чекає приходу 1024х імпульсів перед тим, як відпустити сигнал RESET.

OTP - одноразово програмована пам'ять програм

Pages - сторінки. Метод адресації пам'яті програми. Пристрої midrange мають 11-бітну адресацію на команди CALL і GOTO, що доводить довжину цих команд до 2К слів кожна. Для того щоб використовувати пам'ять великих об`ємів, пам'ять програми поділяється на безперервні сторінки, по 2К слів кожна.

Для звернення до тієї чи іншої сторінці необхідно встановити біти PCLATCH <5:4>. Якщо для маніпуляції є 2 біта, то відповідно до цього можна адресувати 4 сторінки.

PBOR - Programmable Brown-Out Detection/Reset - програмований модуль BOR. Даний модуль є складовою частиною мікроконтролерів Microchip серії PIC18. Цей модуль переводить мікроконтролер в стан RESET, якщо живлення знизилося нижче визначеної межі на визначений часовий інтервал. У мікроконтролерах PIC18 цю межу можна задавати на стадії програмування пристрою в слові конфігурації. Якість даного модуля в мікроконтролерах фірми Microchip настільки висока, що це дозволяє відмовитися від застосування зовнішніх супервізорів живлення.

PC - Program Counter - регістр, в якому знаходиться адреса пам'яті програми по якій знаходиться наступна команда.

PCB - Printed Circuit Board - друкована плата.

PSP - Parallel Slave Port - паралельний ведений порт. Паралельний порт використовується для взаємодії з мікропроцесорною шиною даних.

PDIP - Plastic DIP - пластиковий корпус з дворядним розташуванням ніжок. Відстань між ніжками становить 2.54мм.

PGA - Programmable-Gain Amplifier - підсилювач з програмованим підсиленням. Фірма Microchip випускає мікросхему MCP6S2x - операційний підсилювач з програмованим коефіцієнтом посилення і мультиплексором.

PLVD - Programmable Low-Voltage Detection - програмований модуль виявлення низької напруги. Даний модуль є складовою частиною мікроконтролерів фірми Microchip серії PIC18. При зниженні напруги нижче визначеного програмно-заданого рівня генерується переривання. Виникнення даного переривання може сигналізувати про початок відключення джерела живлення, наприклад, через відсутність напруги. Рекомендується зберегти всі важливі дані в незалежну пам'ять, наприклад EEPROM.

POR - Power-on Reset - вбудований модуль мікроконтролерів фірми Microchip, що робить скидання мікроконтролера після появи напруги живлення

і після перевищення ним визначеного рівня. Даний модуль покликаний забезпечити надійний старт мікроконтролера.

PPS - Peripheral Pin Select. У мікроконтролерах, які мають цю функцію, можна використовувати визначений набір периферійних пристроїв на більшості виводів мікроконтролера.

Pull-Up (resistor) - підтягуючий резистор. Резистор, що з'єднує будь-який ланцюг з джерелом живлення (Напр. +5).

Pull-Down (resistor) - підтягуючий резистор до землі. Резистор, що з'єднує будь-який ланцюг із землею.

PWM - Pulse Width Modulation - широтно-імпульсна модуляція. Ця функція в мікроконтролерах Microchip реалізується за допомогою CCP модуля.

PWRT - Power-up Timer - Внутрішній модуль мікроконтролерів фірми Microchip. Даний таймер утримує процесор в скинутому стані n мс (див. документацію на Ваш контролер), для того, щоб напруга живлення встигла піднятися до необхідного рівня. Робота даного модуля дозволяється/забороняється в слові конфігурації при програмуванні мікроконтролера.

RAM - оперативний запам'ятовуючий пристрій (ОЗП) - пам'ять даних мікроконтролера.

ROM - Read Only Memory - постійний запам'ятовуючий пристрій - пам'ять даних мікроконтролера (EEPROM)

RTCC - Real Time Clock and Calendar - модуль для відліку часу і дати, має також будильник.

Sampling Time - повний час аналого - цифрового перетворення. Включає час захоплення і перетворення.

SAW - Surface Acoustic Wave - поверхнева акустична хвиля (ПАХ)

SLAC - інтегруючий АЦП.

Sleep - режим низького енергоспоживання. У різних контролерах ця операція відбувається по-різному. У мікроконтролерах без технології nanoWatt цей режим досягається виключенням тактового генератора. Однак багато

модулів можуть зберігати свою працездатність. Наприклад продовжує працювати WDT, АЦП та інші модулі.

SOIC - тонкий корпус з відстанню між виводами 1.27 мм

SPI - Serial Peripheral Interface Protocol - протокол послідовного периферійного інтерфейсу. Даний вид протоколу є стандартним для багатьох мікроконтролерів фірми Microchip. Його наявність в мікроконтролері визначається наявністю модуля MSSP (SSP). Зазвичай це 3-х провідний інтерфейс - вихід, вхід, і тактова частота. Можливий синхронний обмін.

Tad - час необхідний для АЦ перетворення одного «біту» сигналу.

Tcy - час виконання команди. *Tcy = Fosc / 4.*

TSOP - Thin Small Outline Package - тонкий корпус із зменшеною відстанню між виводами.

USART - Universal Synchronous Asynchronous Receiver Transmitter - Універсальний синхронно-асинхронний приймач. Цей модуль є вбудованим модулем більшості мікроконтролерів фірми Microchip. Він може працювати як двобічний асинхронний порт або як напівдуплексний синхронний порт. В асинхронному режимі може бути підключений через перетворювач рівнів (наприклад, MAX232, ST232 і ін) до послідовного порту комп'ютера.

USB - Universal Serial Bus - універсальна послідовна шина. Даний вид передачі даних отримав широке розповсюдження в комп'ютерній техніці.

USB OTG - USB On-The-Go. На відміну від звичайного модуля USB, цей модуль дає можливість використовувати мікроконтролер не тільки як пристрій, але і як хост.

Vref - Voltage Reference - еталонна напруга для АЦП або напруга порівняння для компаратора. Фірма Microchip випускає дві мікросхеми такого характеру - MCP1525 і MCP1541. Також в деяких мікроконтролерах фірми Microchip є вбудований модуль Vref.

Wake-Up – «пробудження» мікроконтролера зі стану «sleep». Може бути викликано спрацьовуванням WDT або іншою зовнішньою (внутрішньою) подією.

WDT (Watch Dog Timer) - сторожовий таймер. Основне призначення - скидання процесора у випадку «зависання» програми. Скидання відбувається при переповненні таймера, для надійної роботи таймер має свій власний RC генератор. Для нормальної роботи програми його необхідно періодично обнуляти. Включається апаратно в слові конфігурації при програмуванні мікроконтролера.

XLP - eXtreme Low Power. Мікроконтролери з технологією XLP мають дуже низьке споживання в активному і сплячому режимі, що дозволяє їх застосовувати в пристроях, що живляться від батареї.

XT - один з режимів тактового генератора. Використовується для генерації від 100 КГц до 4 МГц. Встановлюється в слові конфігурації при програмуванні мікроконтролера.

ПЛМ - програмована логічна матриця (Programmable Logic Array, PLA) - функціональний блок, створений на базі напівпровідникової технології, для реалізації логічних цифрових схем.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Смірнов В.В., Смірнова Н.В., Пархоменко Ю.М. Архітектура та програмування периферійних інтерфейсних контролерів: підручник. Кропивницький : ЦНТУ, 2020. 278 с.
   URL: http://dspace.kntu.kr.ua/jspui/handle/123456789/10313
2. Сайт кафедри програмування комп'ютерних систем і мереж.
   URL: http://pksm.kntu.kr.ua/DEVELOPMENTS_2.html
3. Положення про критерії оцінювання знань здобувачів вищої освіти в Центральноукраїнському національному технічному університеті. – Кропивницький : ЦНТУ, 2020. – 15 с.
4. Смірнов В.В., Смірнова Н.В. Програмне забезпечення управляючих мікро-ЕОМ : метод. вказ. до виконан. лаб. робіт для студ. ден. та заоч. форми навч. за напрямом підготовки 6.050102 «Комп’ютерна інженерія»; Кіровоградський. нац. техн. ун-т (КНТУ). Кіровоград : КНТУ, 2015. 109 с.
   URL: http://dspace.kntu.kr.ua/jspui/handle/123456789/7600
5. Смірнов В.В., Смірнова Н.В. Програмне забезпечення управляючих мікро-ЕОМ : метод. вказ. до виконан. самост. робіт для студ. ден. та заоч. форми навч. за напрямом підготовки 6.050102 «Комп’ютерна інженерія» ; Кіровоградський. нац. техн. ун-т (КНТУ). – Кіровоград : КНТУ, 2015. – 41 с.
   URL: http://dspace.kntu.kr.ua/jspui/handle/123456789/7601
6. Microchip Technology Inc. PIC 18FXX2. Однокристальные 8-розрядные FLASH CMOS микроконтроллеры с 10 - разрядным АЦП компании Microchip Technology Incorporated / Microchip Technology Incorporated. USA : ООО «Микро-Чип» (www.microchip.ru), 2003. 299 с.
7. Microchip Technology Inc. PIC18FXX2. Data Sheet. High Performance, Enhanced FLASH. Microcontrollers with 10–Bit A/D / Microchip Technology Inc. Microchip Technology Incorporated, USA (www.microchip.com), 2002. 330 p.

8. Microchip Technology Inc. PIC18FXX2. Data Sheet. High Performance, Enhanced FLASH. Microcontrollers with 10–Bit A/D / Microchip Technology Inc. Microchip Technology Incorporated, USA (www.microchip.com), 2006. 332 p.
9. Microchip Technology Inc. MCP4921/4922. 12–Bit DAC with SPI™ Interface / Microchip Technology Inc. Microchip Technology Incorporated, USA (www.microchip.com), 2007. 42 p.
10. Microchip Technology Inc. 24AA256/24LC256/24FC256. Data Sheet. / 256K I^2C™ CMOS Serial EEPROM / Microchip Technology Inc. Microchip Technology Incorporated, USA (www.microchip.com), 2005. 26 p.
11. ООО «Микро–Чип» Программная реализация I^2C интерфейса (режим ведущего) / ООО «Микро–Чип». М.: ООО «Микро–Чип» (www.microchip.ru), 2001. 8 с.
12. ООО «Микро-Чип» Краткий обзор интерфейса I^2C / ООО «Микро–Чип». М.: ООО «Микро-Чип» (www.microchip.ru), 2001. 7 с.
13. ООО «Микро-Чип» Модуль 10–разрядного АЦП в микроконтроллерах PIC 18CXX2 / ООО «Микро–Чип». М.: ООО «Микро–Чип» (www.microchip.ru), 2001. 10 с.
14. Богатырев Е. А. Энциклопедия электронных компонентов. Большие интегральные схемы / Е.А. Богатырев, В.Ю. Ларин, А.Е. Лякин; под ред. А.Н. Еркина. - Т. 1. М. : ООО «МАКРО ТИМ», 2006. 224 с.
15. Брей Барри Применение микроконтроллеров PIC18. Архитектура, программирование и построение интерфейсов с применением С и ассемблера / Барри Брей. К. : «МК-Пресс»; СПб : «КОРОНА-ВЕК», 2008. 576 с.
16. Предко М. PIC-микроконтроллеры: архитектура и программирование / Майкл Предко. Саратов : Профобразование, 2010. 512 с.
17. Предко М. PIC-микроконтроллеры: архитектура и программирование / Майкл Предко. Саратов : Профобразование, 2017. 512 с.
18. Семенов Б.Ю. Шина I^2C в радиотехнических конструкциях / Б.Ю. Семенов; изд. 2-е. доп. М. : СОЛОН-Пресс, 2004. 224 с.

19. Стюарт Болл Р. Аналоговые интерфейсы микроконтроллеров / Стюарт Болл Р.; пер. с англ. С. Щербинин. М. : Додэка XXI, 2007. 362 с.
20. Шилдт Герберт Полный справочник по C++ / Герберт Шилдт; пер. с англ. к.ф.-м.н. Д.А. Клюшина; 4-е изд. М.: Издательский дом «Вильямс», 2006. 800 с.
21. Шилдт Герберт Полный справочник по C / Герберт Шилдт; пер. с англ. А.Г. Беляева, И.В. Константинова; 4-е изд. М.: Издательский дом «Вильямс», 2002. 704 с.
22. Шпак Ю. А. Программирование на языке C для AVR и PIC микроконтроллеров / Ю. А. Шпак; 2-е изд. К. : «МК-Пресс»; СПб. : «КОРОНА_ВЕК», 2011. 544 с.
23. Хофманн М. Микроконтроллеры для начинающих / М. Хофманн; пер. с нем. СПб. : БХВ-Петербург, 2010. 304 с.
24. Афанасьев Илья Применение модуля захвата, сравнения, ШИМ в контроллерах Microchip / Илья Афанасьев // Журнал «Компоненты и технологии» №3, 2004 г. 3 с.
URL: https://kit-e.ru/assets/files/pdf/2004_03_136.pdf, http://www.microchip.com.ru/Support/tipsCCP%201.html, http://www.radioradar.net/hand_book/documentation/microchip_shim.html,
25. Шина управления I^2C. URL: https://cxem.net/comp/comp67.php
26. SPI (Serial Peripheral Interface) // Материал из Национальной библиотеки им. Н.Э. Баумана. URL: https://ru.bmstu.wiki/SPI_(Serial_Peripheral_Interface)
27. Последовательный интерфейс SPI. URL: https://prog-cpp.ru/micro-spi/
28. SPI Block Guide V03.06 / Motorola, Inc. 38 с. URL: https://web.archive.org/web/20150413003534/http://www.ee.nmt.edu/~teare/ee308l/datasheets/S12SPIV3.pdf
29. Елисеев Н. Интерфейс 1-WIRE: устройство и применение / Н. Елисеев // ЭЛЕКТРОНИКА: Наука, Технология, Бизнес 8/2007. 6 с. URL: http://www.electronics.ru/files/article_pdf/0/article_657_119.pdf

30. Морозов Е. Обзор шины 1-WIRE для создания умного дома / Егор Морозов. URL: https://www.iguides.ru/main/gadgets/other_vendors/obzor_platformy_1_wire dlya_sozdaniya_umnogo_doma/
31. Термины и аббревиатуры. URL: http://www.microchip.ua/index.php?p=termins.php&leng=rus&tl=%D2%E5%F0%EC%E8%ED%FB%20%E8%20%E0%E1%E1%F0%E5%E2%E8%E0%F2%F3%F0%FB
32. C keywords. URL: http://en.cppreference.com/w/c/keyword/
33. Terminal 1.9b. URL: https://sites.google.com/site/terminalbpp/https://micro–pi.ru/terminal–1–9b–%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0% B0%D0%B5%D0%BC–com–%D0%BF%D0%BE%D1%80%D1%82%D0 %BE%D0%BC/
34. Terminal 1.9b – работаем с COM-портом. URL: : https://micro–pi.ru/terminal–1–9b–%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%D0%B5%D0%BC–com–%D0%BF%D0%BE%D1%80%D1%82%D0%BE%D0%BC/
35. COM port development tool. URL: https://sites.google.com/site/terminalbpp/
36. Как пользоваться терминальной программой Terminal 1.9b. URL: https://faq.radiofid.ru/knowledge–bases/2/articles/13–kak–polzovatsya–terminalnoj–programmoj–terminal–19b /
37. Абашкин Иван Как пользоваться терминальной программой Terminal 1.9b. URL: https://faq.radiofid.ru/knowledge–bases/2/articles/13–kak–polzovatsya–terminalnoj–programmoj–terminal–19b/
38. Proteus (система автоматизированного проектирования). URL: https://ru.wikipedia.org/wiki/Proteus_(система_автоматизированного_проектирования)https://ru.wikipedia.org/wiki/Proteus_(система_автоматизированного_проектирования)/
39. PSpice. URL: https://ru.wikipedia.org/wiki/PSpicehttps://ru.wikipedia.org/wiki/PSpice/

40. Описание цифро-аналогового преобразователя MCP4921. URL: https://studbooks.net/2341863/tehnika/ opisanie_tsifro_analogovogo preobrazovatelya_mcp4921
41. G–NOR Lighting LED TECHNOLOGY / GNT–5631Ax–Bx (TRIPLE DIGIT LED DISPLAY) / G–NOR ELECTRONICS CO.,LTD. – 1 с.
42. Трехразрядный светодиодный цифровой дисплей. URL: https://old.radiodetali.com/td/displ/gnt5631.htm/ GNT–5631
43. I²C. URL: https://ru.wikipedia.org/wiki/I%C2%B2C/ I²C
44. Ермолович А.В. Программи́руемая логи́ческая ма́трица URL: https://bigenc.ru/technology_and_technique/text/3178939
45. Современные цифровые технологии / Справочник специалиста в АЦП. URL: http://www.centeradc.ru/guide/#:~:text=%D0%9C%D0%BB%D0%B0%D0%B4%D1%88%D0%B8%D0%B9%20%D0%B7%D0%BD%D0%B0%D1%87%D0%B0%D1%89%D0%B8%D0%B9%20%D1%80%D0%B0%D0%B7%D1%80%D1%8F%D0%B4%20(%D0%9C%D0%97%D0%A0)&text=%D0%9C%D0%B8%D0%BD%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5%20%D0%B2%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5%20%D0%BD%D0%B0%D0%BF%D1%80%D1%8F%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5%20%D1%80%D0%B0%D0%B7%D1%80%D0%B5%D1%88%D0%B0%D0%B5%D0%BC%D0%BE%D0%B5%20%D0%90%D0%A6%D0%9F,%D0%98%D0%B7%D0%BC%D0%B5%D1%80%D1%8F%D0%B5%D1%82%D1%81%D1%8F%20%D0%B2%20%5B%D0%92%5D.

Навчальне електронне видання комбінованого використання.

Можна використовувати в локальному та мережному режимах

Смірнов Володимир Вікторович
кандидат технічних наук, доцент

Смірнова Наталія Володимирівна
кандидат технічних наук, доцент

Пархоменко Юрій Михайлович
кандидат технічних наук, доцент

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ УПРАВЛЯЮЧИХ МІКРО-ЕОМ

Навчальний посібник

Редактор В.В. Смірнов

Технічний редактор В.В. Смірнов

Верстальник Н.В. Смірнова

Видавець

Центральноукраїнський національний технічний університет
25006, м. Кропивницький. пр. Університетський, 8,

E-mail: swckntu@gmail.com