Мiнiстерство освiти і науки

Національний технічний університет України

«Київський політехнічний інститут імені ігоря сікорського»

**МЕТОДИЧНІ ВКАЗІВКИ**

до виконання курсової роботи

з дисципліни «Мікрокомп’ютерні та мікропроцесорні системи»

для студентів спеціальності

152 «Метрологія та інформаційно-вимірювальна техніка»

Київ – 2022

Методичні вказівки до виконання курсової роботи з дисципліни «Мікрокомп’ютерні та мікропроцесорні системи» для студентів спеціальності 152 «Метрологія та інформаційно-вимірювальна техніка»/ Уклад.: С.А.Богомазов, Ю.М. Самарцев – К.: НТУУ «КПІ», 2022. – 35 с.

*Затверджено*

*на засіданні кафедри*

*інформаційно-вимірювальних технологій*

*Протокол № 10/22 від 29 червня 2022 р.*

*Навчальне видання*

**МЕТОДИЧНІ ВКАЗІВКИ**

до виконання курсової роботи з дисципліни «Мікрокомп’ютерні та мікропроцесорні системи»

Укладач: Богомазов Сергій Анатолійович, канд. техн. наук, доц., Самарцев Юрій Миколайович, канд. техн. наук, доц.

Відповідальний

редактор: Шевченко Костянтин Леонідович, докт. техн. наук, доц.

Рецензент: Поворознюк Назар Іванович, канд. техн. наук, доц.

*Під редакцією укладачів*

*Надруковано з оригінал-макета замовника*

**Зміст**

Вступ 4

1 Мета та завдання курсової роботи 4

2 Вказівки до виконання розділів курсової роботи 5

3.1 Структура і зміст курсової роботи 5

3.2 Вимоги до текстової документації 6

3 Розробка апаратного забезпечення мікропроцесорних

пристроїв в середовищі пакету Proteus 8

4 Список рекомендованої літератури 34

**Вступ**

Курсова робота (КР) спрямована на здобуття умінь та навичок розробки апаратно-програмного забезпечення сучасних вбудованих комп’ютерних систем, мікропроцесорних обчислювальних, вимірювальних та керуючих засобів а також засвоєння методик проектування апаратних та програмних засобів мікропроцесорних систем обчислень, вимірювань та керування. Входить до дисципліни “Мікрокомп’ютерні та мікропроцесорні системи”.

Для виконання роботи необхідно знання студентами інформатики, електротехнічних та мікроелектронних засобів, системного та прикладного програмування. Робота є основою для вивчення інформаційно-вимірювальних систем, автоматизованого проектування вимірювальних систем, системних вимірювальних приладів, інтерфейсів вимірювальних систем, виконання дипломного проектування та магістерської дисертації.

1. **Мета та завдання курсової роботи**

Метою виконання курсової роботи є поглиблене вивчення принципів проектування апаратно-програмного забезпечення вбудованих мікропроцесорних засобів, що використовуються для керування інформаційно-вимірювальними системами а також для виконання обчислювальних операцій з обробки вимірювально-керуючої інформації. В результаті вивчення дисципліни студенти повинні:

- одержати поглиблені знання з побудови засобів спеціалізованої мікропроцесорної техніки;

- засвоїти технологію проектування компонентів вбудованих мікропроцесорних засобів систем вимірювань;

- одержати навички розробки програмного забезпечення мікропроцесорних засобів вимірювань;

- вміти самостійно проектувати системи вимірювань на базі сучасних мікропроцесорних засобів відомих світових виробників.

Під час виконання курсової роботи з курсу «Мікрокомп’ютерні та мікропроцесорні системи» необхідно спроектувати програмне забезпечення мікропроцесорного пристрою. Такий пристрій може бути частиною спеціалізованої мікропроцесорної або комп’ютерної системи.

При виконанні курсової роботи необхідно задовольнити таким вимогам:

* курсова робота повинна мати завдання ;
* в курсовій роботі повинна бути сформульована мета роботи;
* курсова робота повинна мати розділи, в кінці розділів - висновки;
* в кінці роботи повинні бути записані висновки за результатами виконаної роботи і огляду літератури ;
* результатом курсової роботи є пояснювальна записка, в якій повинні бути присутні огляд існуючих технічних рішень та опис розробленого програмного забезпечення з лістингами програмного забезпечення та знімками екранів з результатами виконання програм.

**2 Вказівки до виконання розділів курсової роботи**

2.1 Структура і зміст курсової роботи

Пояснювальна записка оформлюється на аркушах формату А4 у відповідності з ДСТУ 3008:2015 "Документація. Звіти в сфері науки і техніки". Об’єм пояснювальної записки повинен мати біля 30 сторінок основного тексту (без додатків). Об'єм текстової документації може бути зменшений за погодженням з керівником проекту та завідуючим кафедрою, якщо студент представляє на захист виготовлений ним або при його участі макет розробленого виробу.

Пояснювальна записка включає: титульний лист, завдання на курсову роботу, пояснювальна записка, додатки до пояснювальної записки.

2.2 Вимоги до текстової документації

Титульний лист оформляють згідно з вимогами ДСТУ 3008:2015 "Документація. Звіти в сфері науки і техніки”.

Пояснювальна записка повинна в короткій і чіткій формі розкрити результати теоретичної і практичної роботи студента над КР, суть методів дослідження та розрахунків, результати розрахунків, а також загальні висновки та рекомендації.

Загальні вимоги до пояснювальної записки: чіткість і логічна послідовність викладення матеріалу, впевненість і аргументація, стислість і точність формулювання, конкретний опис виконаної КР, обґрунтування висновків, рекомендацій і пропозицій.

За формою пояснювальна записка повинна відповідати звіту з науково-дослідної роботи згідно з ДСТУ 3008:2015.

Текст пояснювальної записки виконується в середовищі WORD, на одній сторінці аркушу, розмір шрифту – Times New Roman 14 пт. через 1,5 інтервалу. Вирівнювання основного тексту виконується по ширині. Абзацний відступ повинен складати 5 знаків. Структурні елементи: зміст, вступ, висновки, перелік джерел посилання не нумеруються, а назви оформлюються як розділи.

Розділ повинен починатися з нової сторінки. Назва розділу розміщується по центру рядка без абзацного відступу, виконується шрифтом 14 пт, напівжирним, великими літерами. Розділи мають просту арабську нумерацію 1, 2, 3 без крапки після номеру та без крапки в кінці назви розділу. Скорочення та перенесення слів в назвах розділів не допускається. Відступ до та після назви розділу 16 пт. Підрозділи нумеруються в межах розділу із зазначенням номеру розділу: 1.1, 1.2 і т. д. Назва підрозділу починається з абзацного відступу та виконується шрифтом розміром 14 пт, всі літери крім першої малі. Вирівнювання тексту по лівому краю. Відступ до та після назви підрозділу 10 пт. Пункти нумеруються в межах підрозділу із зазначенням номеру розділу та підрозділу: 1.1.1, 1.1.2 і т.д. Назва пункту починається з абзацного відступу та виконується шрифтом розміром 14 пт, всі літери крім першої малі. Вирівнювання тексту по лівому краю.

Після назви розділу, підрозділу, а також пункту й підпункту на сторінці повинно бути не менше двох рядків тексту.

Всі наведені рисунки повинні бути пронумеровані в межах розділу з вказуванням номеру розділу. Підпис рисунку повинен мати наступний вигляд: «Рисунок 2.1 – Назва рисунку…» та розміщуватися безпосередньо під рисунком. Рисунки розміщуються безпосередньо після першого згадування в тексті або на наступній сторінці. Умовні скорочення та позначення наведені на рисунку потрібно розшифрувати в тексті або в підписі. Рисунки розміщуються по центру рядка без абзацного відступу, відстань до та після рисунку 10 пт.

Таблиці повинні мати нумерацію в межах розділу та назву, розміщену зверху таблиці (текст вирівнюється по лівому краю з абзацним відступом), відстань до та після таблиці 10 пт. Назва таблиці повинна мати вигляд: «Таблиця 2.1 – Назва таблиці… » .

Курсова робота повинна складатися з наступних частин

1. Титульний лист.

2. Бланк завдання.

3.Розрахунок варіантів.

4.Список завдань

5. Пояснювальна записка.

Пояснювальна записка повинна складатися з наступних розділів

- ЗМІСТ

- ВСТУП

1 ОГЛЯД ІСНУЮЧИХ ТЕХНІЧНИХ РІШЕНЬ

2 ТЕХНОЛОГІЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОВОЮ АСЕМБЛЕР А51

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КЕРУВАННЯ ПЕРИФЕРІЙНИМИ ПРИСТРОЯМИ

- ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

- ДОДАТОК А. ЛІСТИНГИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**3 Розробка апаратного забезпечення мікропроцесорних пристроїв в середовищі пакету Proteus**

При розробці програмного забезпечення для керування периферійними пристроями потрібно скористатися засобами комп’ютерного моделювання мікропроцесорних систем, зокрема програмним пакетом **Proteus** фірми Labcenter Electronics.

Спочатку потрібно встановити на комп’ютер програмний пакет **Proteus.** **Proteus** складається з двох основних частин: ISIS і ARES. Для наших цілей ми будемо користуватися лише пакетом схемотехнічного моделювання **ISIS**.

Тепер необхідно налаштувати пакети Proteus і Keil так, щоб вони працювали синхронно за допомогою протоколу TCP/IP. Для цього потрібно зробити наступне.

1.У директорії MODELS інсталяції Proteus (за замовчуванням, наприклад "C:\Program Files\Labcenter Electronics\Proteus 6 Demonstration\MODELS") скопіюйте файл VDM51.dll до директорії BIN пакету Keil (наприклад C:\Keil\uVision2\C51\BIN).

2.У блокноті відредагуйте файл C:\Keil\Tools.ini наступним чином.

У секції [C51] знайдіть рядок

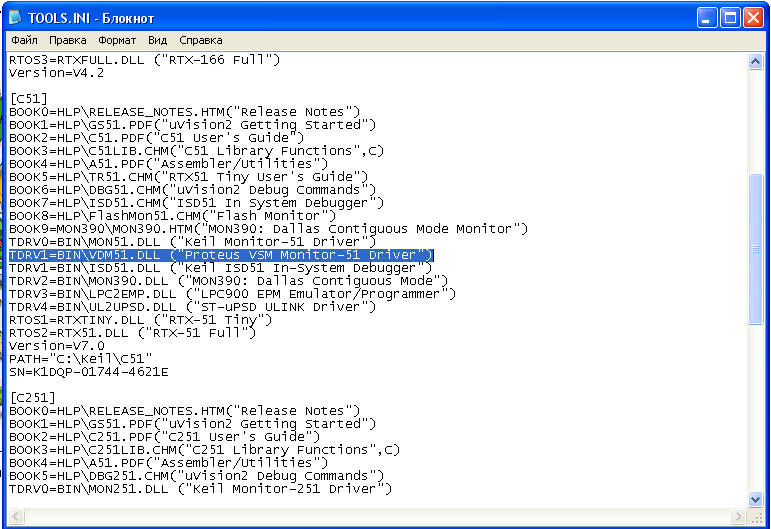
TDRV0=BIN\MON51.DLL ("Keil Monitor-51 Driver")

і вставте під ним наступний

TDRV1=BIN\VDM51.DLL ("Proteus VSM Monitor-51 Driver").

Збережіть файл.

(Іноді тут може виникнути помилка через кодування ASCII символів у WINDOWS. В цьому випадку необхідно у блокноті вручну замінити символи "" , рис.3.1).



### Рисунок 3.1

3. Запустіть ISIS і відкрийте файл Maket.DNS. У меню "Debug" оберіть опцію "Use Remote Debug Monitor". Якщо помилки немає, у командному рядку внизу з'явиться повідомлення "Virtual Debug Monitor server enabled"(рис. 3.2)

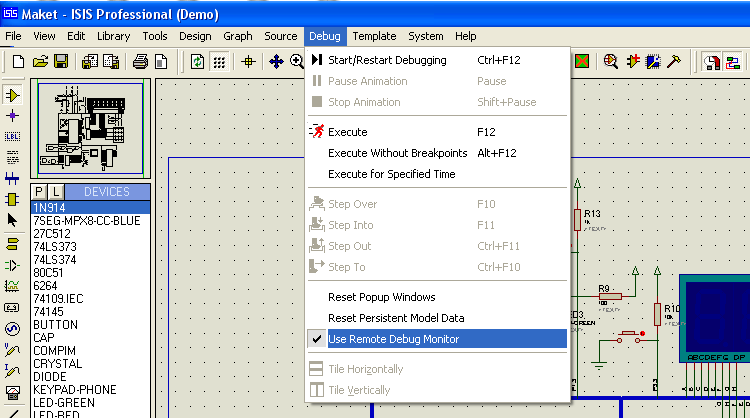


Рисунок 3.2

4. Запустіть Keil і завантажте Ваш проект. У меню Project-->Options for Target ' Target 1' у вкладці Debug виберіть опцію "Use Keil Monitor-51 Driver" і у випадаючому списку оберіть "Proteus VSM Monitor-51 Driver"(рис. 3.3).

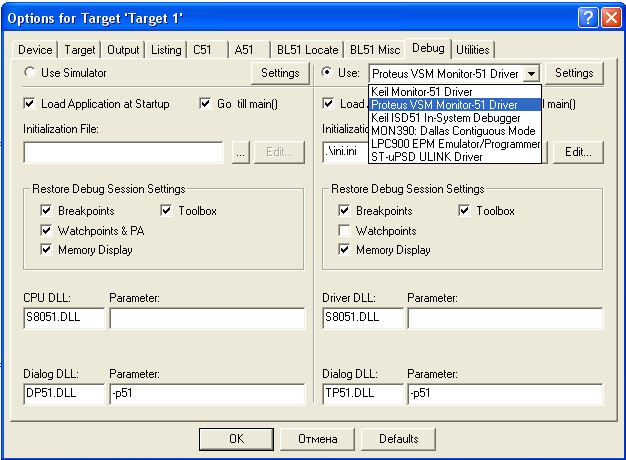


Рисунок 3.3

(якщо такого варіанту немає, то найімовірніше виникла проблема, яка описана вище. Закрийте Keil і ще раз відредагуйте Tools.ini).

Якщо Ви встановили більш нову версію Proteus, то імовірно ви не знайдете у каталозі MODELS файл VDM51.dll. У такому разі потрібно скопіювати цей файл з каталогу, що додається до мануалу, і розмістити його в описані вище каталоги.

При роботі з ISIS може з’явитися попередження наступного вигляду:

[SPICE] TRAN: Timestep to small; timestep=(значення): trouble with node #значення#branch

Ключовою у даному повідомлення є фраза: Timestep to small. Зазвичай цьому повідомленню передують ще кілька попереджень SPICE про DELMIN і GMIN. Для того щоб вирішити цю проблему необхідно внести деякі зміни у налаштування ISIS.

Заходимо System->Set animation option->Spice option->Transient (рис.3.4).

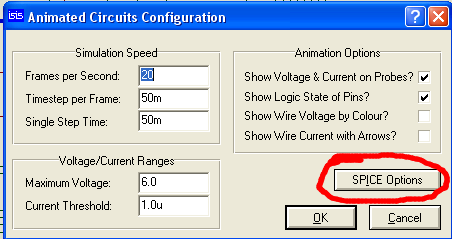
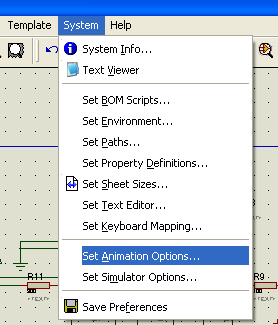


Рисунок 3.4

Змінюємо налаштування на ті, що вказані на рис. 3.5, рис.3.6.

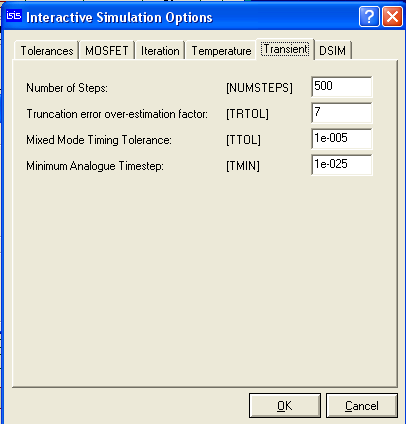
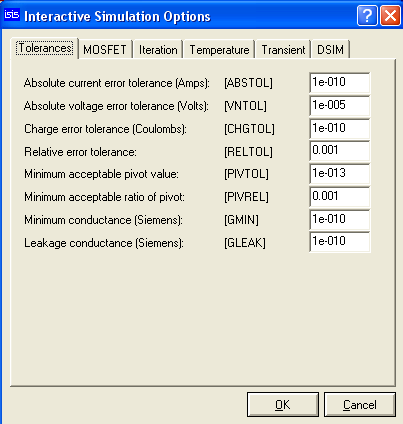


Рисунок 3.5

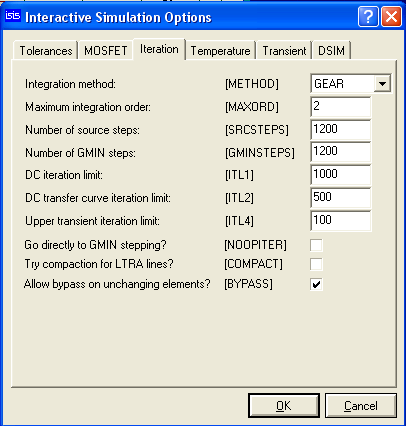


Рисунок 3.6

Схема макету мікропроцесорної системи в цілому виглядає наступним чином (рис. 3.7).

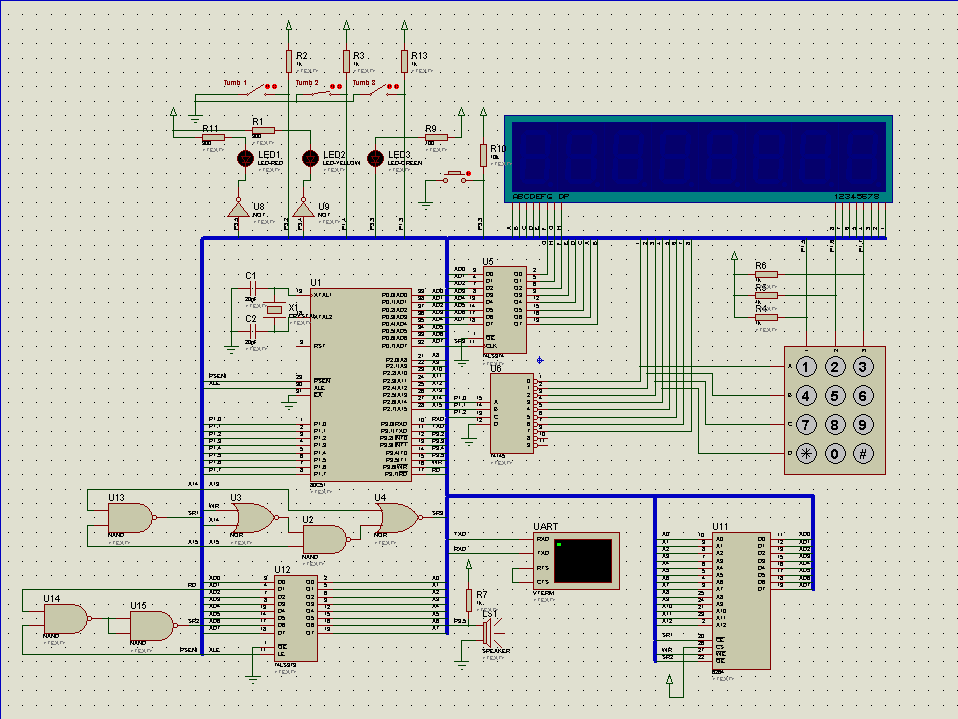


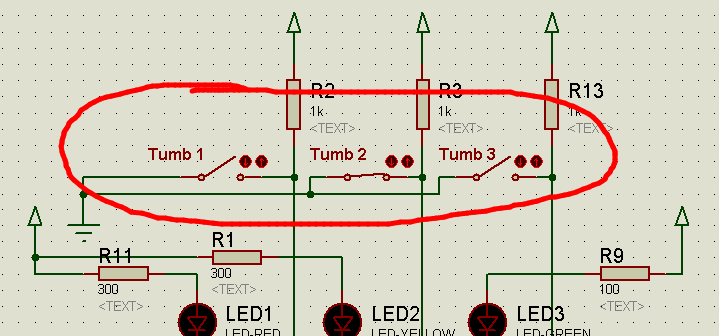
Рисунок 3.7

Отже, все підготовлено до роботи. Тепер детальніше ознайомимось із моделлю макета мікропроцесорної системи. Він містить такі самі засоби вводу-виводу, що і реальний лабораторний макет, а саме:

* три тумблери;
* три світлодіоди;
* кнопку INT1;
* семисегментний індикатор;
* 12-клавішну клавіатуру;
* асинхронний послідовний порт (UART);
* динамік.

Розглянемо складові частини макету системи докладніше.

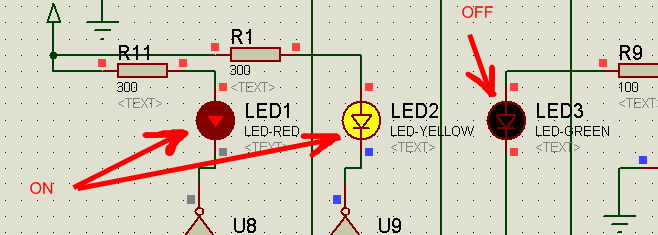
У якості тумблерів у віртуальному макеті використовуються ключі (рис.3.8).



#### Рисунок 3.8

Розімкнений ключ (Tumb 1) відповідає ввімкненому тумблеру, тобто коли тумблер 1 знаходиться у верхньому положенні і на відповідний порт подається логічна «1». І навпаки, замкнений ключ (Tumb 2) відповідає нижньому положенню тумблера 2, при якому на відповідний порт подається логічний «0». Перемикання положення ключа здійснюється клацанням мишкою на відповідну стрілку.

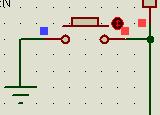
Світлодіоди на схемі виглядають так, як показано на рисунку 3.9.



#### Рисунок 3.9

При цьому, якщо символ залитий чорним кольором (LED3), то світлодіод не горить, і навпаки, якщо символ залитий іншим кольором (LED1, LED2), це означає, що світлодіод горить. Два ліві світлодіоди запалюються високим рівнем напруги (лог. «1»), а правий – інверсний і запалюється низьким рівнем (лог. «0»). На лабораторному макеті записана програма (Monitor), яка гасить перші два світлодіоди після запуску. На віртуальному макеті такої програми немає, тому на початку роботи ці світлодіоди світяться.

Кнопка INT1 на схемі знаходиться праворуч від світлодіодів і виглядає наступним чином (рис.3.10).

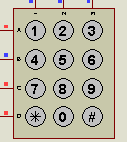
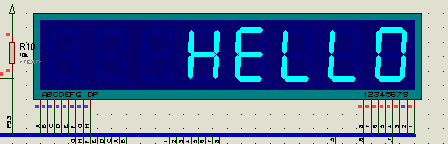


##### Рисунок 3.10

Натискати можна або на саму кнопку, або на стрілку праворуч від неї. Треба зауважити, що віртуальний процесор, на відміну від справжнього, не завжди самостійно скидає прапорець переривання після переходу до функції обробки переривання, тому необхідно робити це програмно, завжди додаючи у функцію обробки INT1 рядок:

IE1 = 0;

Семисегментий індикатор і клавіатура мають такі самі параметри як і лабораторні аналоги і виглядають наступним чином (рис.3.11).



##### Рисунок 3.11

Віртуальний термінал викликається після переходу програми в режим Debug (у пакеті Keil). Для цього в меню Debug в ISIS необхідно активувати опцію Virtual Terminal – UART, після чого з’явиться вікно терміналу для вводу і виводу інформації (рис.3.12).

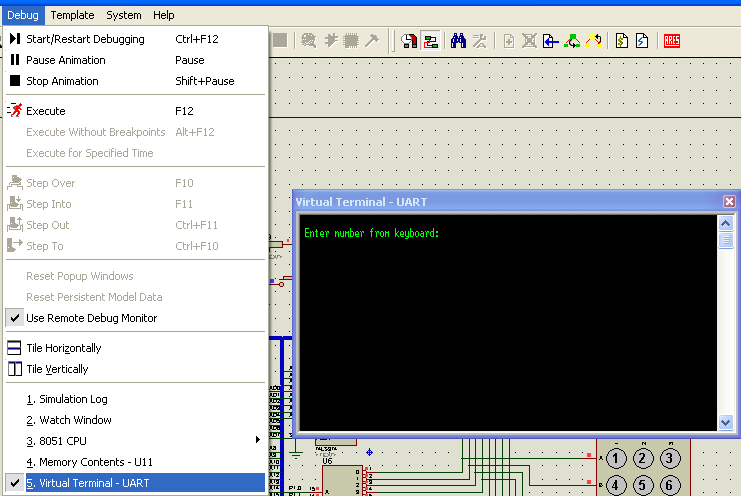


Рисунок 3.12

Враховуючи деякі відмінності віртуального терміналу від реального, необхідно точно витримувати швидкість передачі 4800 бод. Тому внутрішня частота процесора встановлена не 10 МГц, а 11,0592 МГц, що дозволяє отримати точну швидкість. Наслідком цього є те, що при налаштування таймера 1 для роботи з терміналом віртуального макету потрібно записувати у регістр TH1 не 0xF5, а 0xFA.

Динамік (SPEAKER) працює через звукову карту комп’ютера і через описану вище різницю в частоті може мати дещо інше звучання. Також різницю у частоті між справжнім і віртуальним макетом потрібно враховувати при розрахунках у розробках, пов’язаних з таймерами.

Отже, послідовність дій при роботі з віртуальним макетом наступна:

* створюєте проект у Keil, і вибираєте опцію Use Proteus VSM Monitor-51 Driver, як це описано вище;
* відкриваєте ISIS завантажуєте схему з макетом;
* у пакеті Keil переходите в режим Debug;
* кнопки управління налагодженням програми (Start/Pause/Stop) працюють як у Keil так і в ISIS, але для того щоб програма зупинилася на Breakpoint, запускати її потрібно саме в Keil.
* Для перезавантаження процесора достатньо просто зупинити виконання програми кнопкою Stop (рис.3.13).



##### Рисунок 3.13

Іноді Ви можете зустрітися з наступною проблемою: при переході в режим Debug у пакеті Keil програма «не завантажується» на макет. Ця проблема може виникнути після роботи з програмою в симуляторі, що використовується у Keil. У цьому випадку потрібно заново створити проект у Keil (файли з розширенням .c, .a51, .h, .ini можна при цьому залишити) і повторити спробу.

3.4 Розробка програмного забезпечення мікропроцесорних пристроїв в середовищі пакету **Proteus**

Розглянемо процес створення і від лагодження програми за допомогою віртуального макета на прикладі програми EXAMPLE\_ASM. Ця програма підраховує кількість натискань на кнопку INT1 і виводить це значення на індикатор. Код програми наведено нижче.

NAME EXAMPLE\_ASM

; Визначаємо константи

MON\_ADR EQU 8000h ; Адреса індикатора в адресній області

TEN EQU 10

DSEG AT 30h ; Змінна, що буде накопичувати кількість

NUM: DS 1 ; натиснень на кнопку INT1

BSEG AT 20h

OK: DBIT 1 ; Оголошуємо бітові змінні

LONG: DBIT 1

CSEG AT 0h

ORG 13h ; Вектор переривання INT1

JMP BUTTON

ORG 100h

TABLE: DB 0FCh, 0A0h ; Таблиця семисегментних кодів

DB 0D9h, 0F1h ; цифр від 0 до 9

DB 0A5h, 75h

DB 7Dh, 0E0h

DB 0FDh, 0F5h

ORG 200h

MAIN:

ORL TCON, #04h ; Переривання за зрізом

SETB EA ; Розблоковуємо всі переривання

SETB EX1 ; Розблоковуємо переривання INT1

JNB OK, $ ; Чекаємо поки ОК не дорівнює 1

CLR OK ; Обнуляємо змінну ОК

MOV NUM, #1 ; Фіксуємо в NUM перше натиснення

CALL INDICATE ; Викликаємо процедуру INDICATE

;------------------------------------------------

BUTTON: ; Процедура обробки переривання INT1

CLR IE1 ; Очищаємо прапорець (описано в теорії)

SETB OK ; Записуємо ОК = 1

INC NUM ; Інкрементуємо кількість переривань

RETI

;------------------------------------------------

INDICATE:

MOV A, NUM

CJNE A, #TEN, NEXT ; Перевіряємо чи досягли 10-го натиснення

SETB LONG ; Якщо так --> встановлюємо змінну LONG,

; що означає, що число вже містить 2 цифри

NEXT:

MOV A, NUM

MOV B, #TEN

DIV AB ; Виділяємо десятковий розряд

JNB LONG, SHORT ; Якщо число з 1 цифри --> переходимо

MOV DPTR, #TABLE ; Записуємо в DPTR адресу таблиці

; семисегментних кодів

MOVC A, @A+DPTR ; Завантажуємо в акумулятор код тієї

; цифри, що в ньому міститься

MOV R1, A ; Зберігаємо цей код в R1

MOV DPTR, #MON\_ADR ; Записуємо в DPTR адресу індикатора

CLR A

MOVX @DPTR, A ; Очищуємо індикатор

MOV P1, #0F1h ; Завантажуємо в Р1 знакомісце

MOV A, R1

MOVX @DPTR, A ; Виводимо цифру на індикатор

MOV R3, #0Fh

DJNZ R3, $ ; Затримка

SHORT:

MOV A, B ; Записуємо старший розряд

MOV DPTR, #TABLE

MOVC A, @A+DPTR ; Дістаємо код цифри

MOV R1, A

MOV DPTR, #MON\_ADR

SHORTLOOP:

CLR A ; Очищаємо індикатор від

MOVX @DPTR, A ; попереднього символу

MOV P1, #0F0h ; Завантажуємо в Р1 знакомісце

MOV A, R1

MOVX @DPTR, A ; Виводимо символ на індикатор

MOV R3, #0Fh

DJNZ R3, $ ; Затримка

JB OK, INDICATE ; Якщо було натискання на INT1, то

; переходимо на початок процедури

JNB LONG, SHORTLOOP; Якщо число з однієї цифри - короткий цикл

JMP INDICATE ; Інакше - на початок процедури

RET

END

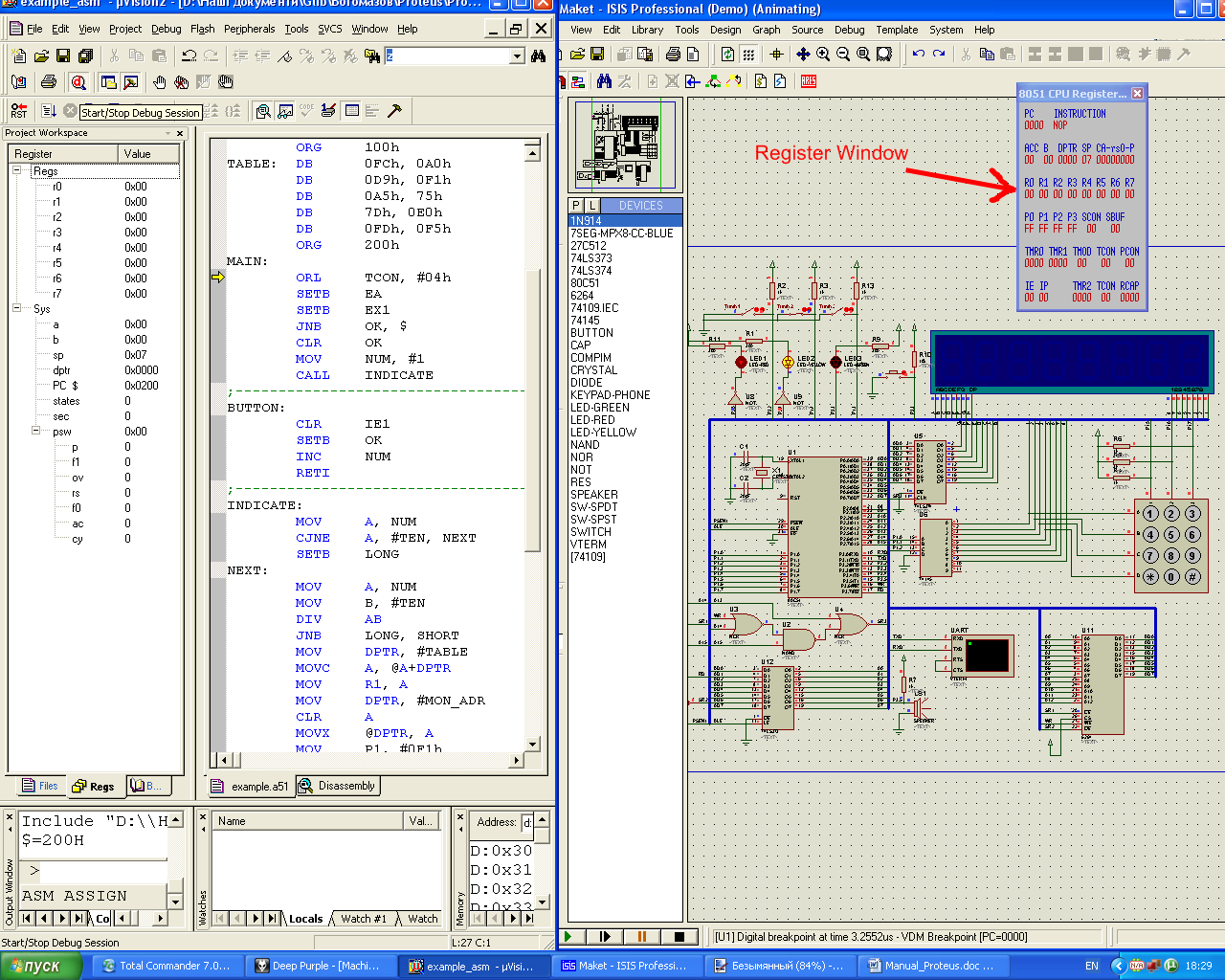
Отже, створюємо новий проект у Keil і додаємо до нього файл EXAMPLE\_ASM.a51. Заходимо у властивості проекту і у вкладці Debug вибираємо Use: “Proteus VSM Monitor-51 Driver”, як це було описано раніше.

Додаємо до проекту ini-файл з рядком:

$ = 0200h

(При роботі з реальним макетом в лабораторії до адрес в пам’яті кодів необхідно буде додати зміщення 0c000h).

Запускаємо ISIS і відкриваємо файл Maket.DNS. Для зручності можна розташувати обидва вікна відкритими одночасно (рис.3.14).



##### Рисунок 3.14

Вікно регістрів, позначене на малюнку, містить інформацію про регістри загального користування R0 – R7, регістри портів P0 – P3, акумулятор АСС та його розширювач В, вміст програмного лічильника РС, регістр DPTR, регістр прапорців PSW (CA-RS0-P) та ще деякі важливі системні регістри. Його можна вивести на екран в режимі Debug, вибравши меню Debug -> 8051 CPU -> Registers. Також можна вивести вікна що відображають вміст різних областей пам’яті, Watch Window для відображення необхідних регістрів та змінних.

Запускаємо програму. Натискаємо мишкою на кнопку INT1 і бачимо, що на індикаторі з’явився символ одиниці (рис.3.15).



##### Рисунок 3.15

Натискаємо ще раз, і на індикаторі з’являється двійка (рис.3.16).



##### Рисунок 3.16

Розглянемо ще один приклад. Створимо проект з програмою example\_c.

#include "kscanf.h"

#include "pprintf.h"

char xdata string[MAX\_CHARS];

void main(void)

{

while(1)

{

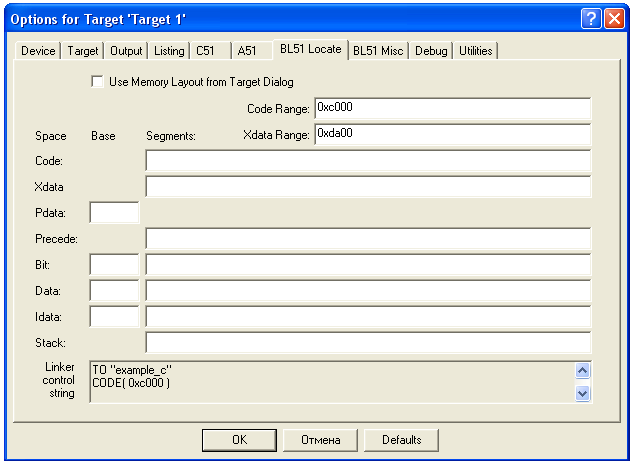
kscanf(&string);

pprintf(string);

}

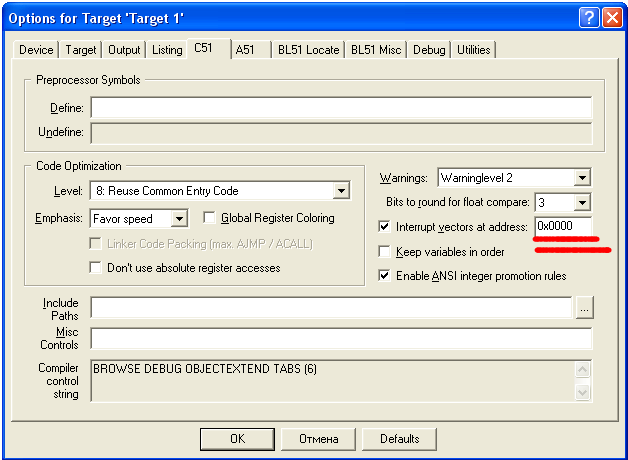
}

Підключаємо до проекту файл DEBUG.A51 та копіюємо в цю папку всі необхідні h-файли. Заходимо в налаштування проекту та змінюємо необхідні параметри (рис.3.17).



##### Рисунок 3.17

Змінювати початкову адресу векторів переривань не потрібно, тому що на віртуальному макеті вони починаються з адреси 0х0000, а не 0хС000 (рис.3.18).



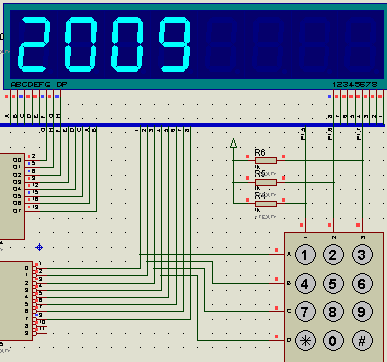
##### Рисунок 3.18

Вибираємо Use Proteus Monitor і компілюємо програму. Прикріплюємо до проекту ini-файл з текстом:

$ = 0c000h

g, main

Заходимо в режим Debug і запускаємо програму. Функція kscanf одразу відображає на індикатор цифри введені на клавіатурі макету. Клавіша дозволяє стирати попередній введений символ, а клавіша записує введений рядок чисел у масив, адреса якого є вхідним параметром цієї функції. Функція pprintf виводить на індикатор семисегментне зображення масиву ASCII-символів, що містяться у вхідному масиві (це можуть бути не тільки числа, а й символи латинського алфавіту) (рис.3.19).



##### Рисунок 3.19

Принципи роботи з таймером продемонстровано в прикладі Timer0-asm.

;===============================================================

;

; Author : ADI - Apps

;

;

; File : Tim2\_AR.asm

;

; Hardware : MCS-51

;

; Description : Flashes light every 256\*128 machine cycles

; using timer 0 in Mode 1.

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

;Use 8051 predefined Symbols

LED EQU P3.5

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; BEGINNING OF CODE

CSEG AT 0000H

ORG 0000H

JMP MAIN

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; TIMER 0 INTERRUPT ROUTINE

ORG 0000Bh

CPL LED ; Flash the light

CLR EA ; Disable interrupts

; Initialise timer 0 registers

MOV TH0, #80h

MOV TL0, #00h ; values 128\*256 machine cycles

SETB EA ; Enable interrupts

RETI

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ; MAIN PROGRAM

ORG 00060h ; Start code at address above interrupts

MAIN: ; Main program

SETB ET0 ; Enable timer0 interrupt

MOV TH0, #80h

MOV TL0, #00h ; Initialise timer 0 registers

;values to every 128\*256 machine

;cycles

ORL TMOD,#01h ; Mode 1

SETB EA ; Enable interrupts

SETB TR0 ; Start timer 0 in Mode 1

JMP $ ; Wait here for timer Interrupts

END

Принципи роботи з роботу з послідовним портом продемонстровано в прикладі UART1-asm.

;====================================================================

;

; Author : ADI - Apps

;

; File : UART.asm

;

; Hardware : MCS-51

;

; Description : This Program saves 16 numbers in order initially

; starting with 0 into memory locations 40h to 50h.

; When finished the values in these locations are

; transmitted down the UART in ASCII form to the PC

; where they can be viewed using the preconfigured

; Hyperterminal program. (c:\ADuC\9600com1.ht)

;

; After the transmission of the 16 bytes a 5 second

; delay is called and the process is repeated, this

; time starting with the saving of 10h to location

; 40h.

;

;====================================================================

;

;Use 8051 predefined Symbols

LED EQU P3.5

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; BEGINNING OF CODE

CSEG

ORG 00000H

JMP MAIN

ORG 00060H ; Start code at address above interrupts

MAIN: ; Main program

MOV SCON,#52h ; config UART for 4800baud

ORL TMOD,#20h ; for 10 MHz

MOV TH1,#0FAh

SETB TR1

MOV R0, #00 ; start output data at 0

MOV R1, #40h ; initialise R1 to 40 to store the

; input data from memory location 40

SAVENOS:

MOV A,R0

MOV @R1, A ; move R0 into memory location R1

INC R1 ; increment memory location and data so

; new data is stored in new address

INC R0

CJNE R1, #50H, SAVENOS ; reset memory location to 40h

; when memory location reaches 50h

; saving 16 bytes of data

; Transmit the values in locations 40h->50h up the UART wait for

; 4 seconds and then repeat

START: CPL LED ;CPL LED with each transmission

MOV DPTR, #TITLE

CALL SENDSTRING ; write title block on screen

MOV R1, #40h ; move value at address 40 into R2

MOV A, @R1

MOV R2, A

NEXT: ; Put new value on a new line

MOV A, #10 ; Transmit a linefeed (= ASCII 10)

CALL SENDCHAR

MOV A, #13 ;Transmit a carriage return (=ASCII 13)

CALL SENDCHAR

MOV A, R2 ;Transmit R2 i.e. value @ address R1

CALL SENDVAL

INC R1 ; Increment address

MOV A, @R1

MOV R2, A ; R2 holds the value @ addrR1

MOV A, R1 ; Check if at address 50h

CJNE A, #50h, NEXT ; if not jump to Next

JMP WAIT5S ; if so wait ~5s and repeat

WAIT5S: MOV A, #200

CALL DELAY ; Wait ~4 seconds

MOV R1, #40h

JMP SAVENOS ; Resave new numbers to same addresses

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; SENDSTRING

SENDSTRING: ; sends ASCII string to UART starting at location

; DPTR and ending with a null (0) value

PUSH ACC

PUSH B

CLR A

MOV B,A

IO0010: MOV A,B

INC B

MOVC A,@A+DPTR

JZ IO0020

CALL SENDCHAR

JMP IO0010

IO0020: POP B

POP ACC

RET

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; SENDCHAR

SENDCHAR: ; sends ASCII value contained in A to UART

JNB TI,$ ; wait til present char gone

CLR TI ; must clear TI

MOV SBUF,A

RET

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; SENDVAL

SENDVAL: ; converts the hex value of A into two ASCII chars,

; and then spits these two characters up the UART.

; does not change the value of A.

PUSH ACC

SWAP A

CALL HEX2ASCII

CALL SENDCHAR ; send high nibble

POP ACC

PUSH ACC

CALL HEX2ASCII

CALL SENDCHAR ; send low nibble

POP ACC

RET

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; HEX2ASCII

HEX2ASCII: ; converts A into the hex character representing the

; value of A's least significant nibble

ANL A,#00Fh

CJNE A,#00Ah,$+3

JC IO0030

ADD A,#007h

IO0030: ADD A,#'0'

RET

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; DELAY

DELAY: ; Delays by 100ms \* A

; 100mSec based on 1.5728MHZ

; Core Clock

; i.e. default ADuC824 Clock

MOV R1,A ; Acc holds delay variable

DLY0: MOV R2,#019h ; Set up delay loop0

DLY1: MOV R3,#0FEh ; Set up delay loop1

DJNZ R3,$ ; Dec R3 & Jump here until R2 is 0

DJNZ R2,DLY1 ; Dec R2 & Jump DLY1 until R1 is 0

DJNZ R1,DLY0 ; Dec R1 & Jump DLY0 until R0 is 0

RET ; Return from subroutine

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TITLE: DB 10,10,13,'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_',10,13

DB 'Analog Devices MicroConverter ADuC824',10,13

DB ' UART Demo Routine',10,13

DB ' Data Stored in Memory in Hex Form',10,13,0

END

Принципи роботи з матричною клавіатурою продемонстровано в прикладі one\_key\_asm.

;====================================================================

;

; Author : Korotenko S.A. VA-12

;

; File : Keys1.a51

;

; Hardware : MCS-51

;

; Description : Indication of pushed button of keyboard

; in right size of indicator.

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

;Use 8051 predefined Symbols

dseg at 30h

mas: ds 15

sdvig: ds 1

cseg at 0000h

org 0000h

jmp main

mas2: DB 10h,0FCh,0A0h,0D9h,0F1h,0A5h,75h,7Dh,0E0h,0FDh,0F5h,39h,51h

; 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \*, #

main:

call mas\_to\_memory ;(подпрограмма перенесення з "mas2" в

;"mas" )

mov sdvig,#0

cycl: mov DPTR,#8000h

mov A,#0

movx @DPTR,A ; погасили сегменти

; операция сканирования клавиатуры

anl p1,#11111000b ; встановили 0 на дешифраторі, перевіряемо

; перший рядок

jnb p1.5,pressed\_1 ; перевірка ліній повернення,при 0 перехід

; на мітку

jnb p1.6,pressed\_2 ;

jnb p1.7,pressed\_3 ;

anl p1,#11111000b

orl p1,#01b ; встановили 1 на дешифраторі, перевіряемо

; другий рядок

jnb p1.5,pressed\_4

jnb p1.6,pressed\_5

jnb p1.7,pressed\_6

anl p1,#11111000b

orl p1,#10b ; встановили 2 на дешифраторі, перевіряемо

; третій рядок

jnb p1.5,pressed\_7

jnb p1.6,pressed\_8

jnb p1.7,pressed\_9

anl p1,#11111000b

orl p1,#11b ; встановили 3 на дешифраторі, перевіряемо

; четвертий рядок

jnb p1.5,pressed\_star ;\*

jnb p1.6,pressed\_0

jnb p1.7,pressed\_num ;#

; операція виводу на індикатор символу натисненої клавиші

mov A,#mas ;завантаження адреси масиву

add A,sdvig ;зсув адреси в залежності від клавиші

mov R0,A

mov A,@R0 ;завантаження в А значення з масиву

anl p1,#11111000b ; выбір першого індикатора

movx @DPTR,A ; вывід на индикатор значення клавиші

jmp cycl

pressed\_1:

jnb p1.5,pressed\_1

mov sdvig,#2

jmp cycl

pressed\_2:

jnb p1.6,pressed\_2

mov sdvig,#3

jmp cycl

pressed\_3:

jnb p1.7,pressed\_3

mov sdvig,#4

jmp cycl

pressed\_4:

jnb p1.5,pressed\_4

mov sdvig,#5

jmp cycl

pressed\_5:

jnb p1.6,pressed\_5

mov sdvig,#6

jmp cycl

pressed\_6:

jnb p1.7,pressed\_6

mov sdvig,#7

jmp cycl

pressed\_7:

jnb p1.5,pressed\_7

mov sdvig,#8

jmp cycl

pressed\_8:

jnb p1.6,pressed\_8

mov sdvig,#9

jmp cycl

pressed\_9:

jnb p1.7,pressed\_9

mov sdvig,#10

jmp cycl

pressed\_star:

jnb p1.5,pressed\_star

mov sdvig,#11

jmp cycl

pressed\_0:

jnb p1.6,pressed\_0

mov sdvig,#1

jmp cycl

pressed\_num:

jnb p1.7,pressed\_num

mov sdvig,#12

jmp cycl

mas\_to\_memory: ;підпрограмма перенесення з "mas2" в "mas"

mov DPTR, #mas2 ;завантаження адреси масиву с даними

mov R0, #mas ; завантаження адреси масиву в пам’яті

; даних

mov R1,#15 ; кількість элементів масиву

next\_element:

mov A,#0 ; очищення А

movc A,@A+DPTR ; в А i-те значення "mas2"

mov @R0,A ; в "mas" значення А

inc DPTR ; наступний елемент "mas2"

INC R0 ; наступний елемент "mas"

DJNZ R1,next\_element ; кінець массиву?

ret

END

Принципи організації динамічної індикації продемонстровано в прикладі Hello\_asm.

;====================================================================

;

; Author : Korotenko S.A. VA-12

;

; File : Hallo.a51

;

; Hardware : MCS-51

;

; Description : Output of word "HELLO" on the Indicators

;

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

;Use 8051 predefined Symbols

N equ 5

dseg at 30h

mas: ds N

cseg at 00000h

org 00000h

jmp main

mas2: DB 10101101b, 1011101b, 11100b, 11100b,11111100b

; H E L L O

Main:

call mas\_to\_memory ;пересылка массива из

;памяти кода в память данных

mov R0,#mas

mov DPTR,#8000h

mov p1,#4

mov R1,#N

cycl: mov A,@R0 ;загрузка в А элемента массива

movx @DPTR,A ;вывод на индикатор

inc R0 ;увеличение адреса массива

mov A,#0 ;

movx @DPTR,A ;гасим индикаторы

dec p1 ;выбирам следующий индикатор

djnz r1,cycl ;все ли символы выведены?

jmp main

mas\_to\_memory: ;(подпрограмма перенесения из "mas2" в "mas" )

mov DPTR, #mas2 ;загрузка указателя на массив с данными

mov R0, #mas ; загрузка указателя на массив в памяти данных

mov R1,#N ; загрузка кол-ва элементов массива

next\_element:

mov A,#0 ; обнуление А

movc A,@A+DPTR ; загрузка в А i-го значения "mas2"

mov @R0,A ; загрузка в "mas" значение А

inc DPTR ; следующий элемент "mas2"

INC R0 ; следующий элемент "mas1"

DJNZ R1,next\_element ; конец массива?

ret

end

Нижче наведено приклад програми, де використовуються всі пристрої віртуального макету. Після натискання на кнопку INT1 на термінал виводиться запит на введення логіна через термінал (login = qwerty). При невірному вводі програма пропонує повторити спробу. Після введення правильного логіна на термінал виводить запит на введення пароля з клавіатури макету. Після вводу правильного пароля програма на кілька секунд виводить його на індикатор, а потім подає звуковий сигнал. Протягом всього виконання програми за допомогою таймера 0 блимає середній світлодіод.

#include <reg51.h> // Підключаємо необхідні бібліотеки

#include <stdlib.h>

#include "kscanf.h"

#include "pprintfn.h"

#define RELOAD\_HIGH 0xD4; // Визначаємо частоту переривання

#define RELOAD\_LOW 0xCD; // таймера 0

#define pass 2009 // Константа пароль = 2009

sbit beep = P3^5;

sbit LED = P3^4;

bit ok, flag;

unsigned int counter = 0;

char xdata str[MAX\_CHARS];

void init (void);

void main (void)

{

long password;

while (1)

{

unsigned char i, log\_in[6];

init(); //Викликаємо функцію ініціалізації

sprintf(log\_in, "qwerty");//Записуємо в масив log\_in

// послідовність ASCII-символів "qwerty"

ok = 0; // Обнуляємо бітові змінні

flag = 0;

while (!ok) {} // Чекаємо поки ok не дорівнює 1

start:

printf("\n Enter login: ");// Виводимо на термінал повідомлення

for (i = 0; i < MAX\_CHARS; i++)

{

str[i] = 0; // Очищуємо масив str[]

}

gets(str, sizeof(str)); // Зчитуємо масив з терміналу

for (i = 0; i < 6; i++) // Посимвольно перевіряємо логін

{

if (str[i] != log\_in[i])

{ // Якщо неправильний символ

flag = 1; // встановлюємо прапорець і

break; // виходимо з циклу перевірки

}

}

if (flag) // Якщо прапорець встановлено

{ // виводимо повідомлення

printf("\n Wrong login! Try again!");

flag = 0; // Обнуляємо прапорець і переходимо

goto start; // на мітку start для повторної спроби

}

else printf("\n Enter password from keyboard");

// Якщо flag = 0 то логін - правильний

// і виводимо запрошення ввести пароль

entpass: for (i = 0; i < MAX\_CHARS; i++)

{

str[i] = 0; // Очищаємо масив str[]

}

kscanf(&str); // Зчитуємо пароль з клавіатури

password = atoi(str); // Переводимо з ASCII в int

if (password == pass)// Перевіряємо пароль

{ // Якщо пароль правильний, виводимо

pprintf(str); // його на індикатор на кілька секунд

ok = 0; // Обнуляємо бітову змінну ok

while (!ok) // Поки ok = 0 виводимо звуковий сигнал

{

beep = 0;

for (i = 0; i < 0x80; i++) {}

beep = 1;

for (i = 0; i < 0x80; i++) {}

}

}

else

{

sprintf(str, "FALLS");// Якщо пароль невірний

pprintf(str); // виводимо на індикатор повідомлення

goto entpass; // Переходимо на мітку entpass для

} // повторної спроби

}

}

//-----------------------------------------------

void init (void) // Функція ініціалізації

{

SCON = 0x52; // Налаштовуємо регістр послідовного порту

TMOD |= 0x21; // Таймер0 в режим 1, таймер1 в режим 2

TH1 = 0xFA; // Визначаємо бітрейт 4800

TH0 = RELOAD\_HIGH; // Переривання таймера кожні 11059

TL0 = RELOAD\_LOW; // машинних циклів

TCON |= 0x04; // Переривання за зрізом

EA = 1; // Розблоковуємо всі переривання

ET0 = 1; // Розблоковуємо переривання таймера 0

EX1 = 1; // Розблоковуємо переривання INT1

TR0 = 1; // Запускаємо таймер 0

TR1 = 1; // Запускаємо таймер 1

}

//-----------------------------------------------

void button (void) interrupt 2 // Обробка переривання INT1

{

ok = 1; // Встановлюємо бітову змінну ok

IE1 = 0; // Скидаємо прапорець переривання

}

//-----------------------------------------------

void timer0 (void) interrupt 1{ // Обробка переривання таймера 0

TR0 = 0; // Зупиняємо таймер 0

TH0 = RELOAD\_HIGH; // Перезавантажуємо регістр таймера

TL0 = RELOAD\_LOW;

TR0 = 1; // Запускаємо таймер

if (counter < 10)

{ // Кожні перші 10 переривань таймера

LED = 1; // світлодіод горить

}

else

{ // Наступні 30 переривань таймера

LED = 0; // світлодіод не горить

if (counter > 40) // Після сорока циклів

{ // обнуляємо лічильник

counter = 0;

}

}

counter++; // Інкрементуємо лічильник

}

###### 4 Список рекомендованої літератури

1. Спеціалізовані мікроконтролерні системи. Теорія і практика : Підручник / Є. І. Сокол, І. Ф. Домнін, О. М. Рисований та ін. – Харків: НТУ “ХПІ”, 2007. – 252 с.

2. Мікропроцесорні та мікроконтролерні системи : підручник. У 2 ч. Ч. 1. Мікропроцесорні системи [Електронний ресурс] / А. О. Новацький. – Електронні текстові дані (1 файл: 43,8 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2019. – 367 с. : ил.

3. Мікропроцесорні та мікроконтролерні системи: Ч.2 «Проектування мікропроцесорних систем» [Електронний ресурс] / А.О. Новацький ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 20,3 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 460 с.

4. Мікропроцесорна техніка: Підручник / Якименко Ю.І., Терещенко Т.О., Сокол Є.І., Жуйков Ю.С., Петергеря Ю.С.; за ред.. Т.О. Терещенко. – 2-ге вид. переробл. та допов. –К.: ІВЦ “Видавництво ”Політехніка”; “Кондор”, 2004. 440 с.

5. Проектування мікропроцесорних систем: Проектування мікропроцесорних систем на базі мікроконтролерів сімейства MCS-51: Периферійні модулі мікроконтролерів сімейства MCS-51 / А. О. Новацький. – Київ : НТУУ «КПІ», 2016. – 399 с.: ил.

6. Поджаренко В.О., Кучерук В.Ю., Севастьянов В.М. Основи мікропроцесорної техніки. Навчальний посібник. - Вінниця: ВНТУ, 2006. - 226 с.

7. Схемотехніка електронних систем: У 3 кн. Кн.3. Мікропроцесори та мікроконтролери : Підручник / В.І. Бойко, А.М. Гуржій, В.Я. Жуйков та ін.- К.: Вища Шк.,2004.- 399с.

8. Р.П. Шевчук // Навчальний посібник з дисципліни „Мікропрограмування”, для студентів напрямку „Програмна інженерія”. – Тернопіль, 2011. - 121 с.

9. Програмування вбудованих систем : метод. вказівки до виконання лабораторних робіт для студентів денної та заочної форми навчання за спеціальністю 123 “Комп’ютерна інженерія ” / уклад. Дрєєва Г.М., Дрєєв О.М., Денисенко О.О., Коноплицька-Слободенюк О.К. — Кропивницький: ЦНТУ, 2018. — 90 с.