МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МІКРОПРОЦЕСОРНІ ПРИСТРОЇ КЕРУВАННЯ ТА ОБРОБКИ ІНФОРМАЦІЇ

МЕТОДИЧНІ ВКАЗІВКИ ДО КОМП'ЮТЕРНОГО ПРАКТИКУМУ

ЧАСТИНА 2

Київ 2010

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МІКРОПРОЦЕСОРНІ ПРИСТРОЇ КЕРУВАННЯ ТА ОБРОБКИ ІНФОРМАЦІЇ

МЕТОДИЧНІ ВКАЗІВКИ ДО КОМП'ЮТЕРНОГО ПРАКТИКУМУ

ЧАСТИНА 2

для студентів спеціальностей 7.0912.01 «Акустичні засоби та системи», 7.0912.02 «Медичні акустичні та біоакустичні прилади і апарати», всіх форм навчання

Затверджено Методичною радою НТУУ «КПІ»

Київ 2010

Мікропроцесорні пристрої керування та обробки інформації. Методичні вказівки до комп'ютерного практикуму для студентів спеціальностей 7.0912.01 «Акустичні засоби та системи», 7.0912.02 «Медичні акустичні та біоакустичні прилади і апарати», всіх форм навчання. - К.: НТУУ "КПІ", 2010. – 68 с.

Навчальне видання

Мікропроцесорні пристрої керування та обробки інформації. Методичні вказівки до комп'ютерного практикуму для студентів спеціальностей 7.0912.01 «Акустичні засоби та системи», 7.0912.02 «Медичні акустичні та біоакустичні прилади і апарати», всіх форм навчання

Укладачі: Тодоренко Віктор Агафонович, доц., к.т.н.
Батрак Лариса Миколаївна, ст. викл.
Хоменко Олександр Васильович, ст. викл.

Відповідальний редактор: Жуйков В.Я., проф., д.т.н.

Рецензент: Михайлов С. Р., доц., к.т.н.

ЗМІСТ

стор.

ВСТУП

Сучасну електроніку важко представити без такої важливої складової, як мікроконтролери.

Використання мікроконтролерів в системах керування забезпечує досягнення високих показників ефективності при низькій вартості. Мікроконтролери є ефективним засобом автоматизації різноманітних об'єктів і процесів. Можна вважати що мікроконтролер – це комп'ютер, що розмістився в одній мікросхемі. Звідси і його основні привабливі якості: малі габарити; високі продуктивність, надійність і здатність бути адаптованим для виконання самих різних завдань.

Структурна організація, набір команд і апаратурно-програмні засоби введення/виведення інформації мікроконтролерів краще всього пристосовані для вирішення завдань керування і регулювання в приладах, пристроях і системах автоматики, а не для вирішення завдань обробки даних.

Метою другої частини комп’ютерного практикуму з дисципліни „Мікропроцесорні пристрої керування та обробки інформації” є закріплення теоретичних знань з апаратно-програмних засобів введення/виведення статичних, імпульсних та аналогових сигналів у мікропроцесорах сімейства MCS-51, дослідження роботи послідовного порту та набуття практичних навичок з програмування взаємодії мікроконтролера з клавіатурою та виведення даних на індикатори.

Матеріал може використовуватися при курсовому і дипломному проектуванні, а також може бути корисним для інженерів - схемотехників, що займаються розробкою і експлуатацією електронної апаратури.

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 1

ВИВЕДЕННЯ І ВВЕДЕННЯ СТАТИЧНИХ ТА ІМПУЛЬСНИХ ЦИФРОВИХ СИГНАЛІВ У МІКРОПРОЦЕСОРАХ СІМЕЙСТВА MCS-51

1. Мета практикуму

- Ознайомитися з методами виведення і введення статичних та імпульсних цифрових сигналів у мікропроцесорах сімейства MCS-51;
- Освоїти алгоритми виведення і введення цифрових сигналів.

2. Програма практикуму

2.1. Вивчити особливості виведення/введення цифрових сигналів;

2.2. У програмному середовищі μ-ViSiON/51 написати, асемблювати і відлагодити програми:

2.2.1. виведення статичних цифрових сигналів;

2.2.2. виведення імпульсних цифрових сигналів із заданою тривалістю та періодом повторення. Частоту генератора вибрати самостійно;

2.2.3. виведення в послідовному коді імпульсних цифрових сигналів з формуванням імпульсів синхронізації по кожному переданому біту. Частоту генератора вибрати самостійно;

2.2.4. введення статичних цифрових сигналів;

2.2.5. введення цифрових сигналів, сформованих з використанням контактної пари (із придушенням брязкоту контактів);

2.2.6. вимірювання тривалість цифрового сигналу;

2.3. З використанням програмного симулятора dScope-51 покроково виконати програми. Простежити за зміною даних у відповідних регістрах спеціального призначення (Special Function Registers - SFR).

3. Завдання до практикуму

У табл. 1.1 наведені вихідні дані, що відносяться до виведення, а в табл. 1.2 – до введення цифрових сигналів, де: "порт" - порт призначення; *T* - період

повторення сигналу; δ - тривалість сигналу; SDA - порт виведення даних; SCL - порт виведення синхросигналів; $T_{ДР}$ - час брязкоту контактів; $T_{И}$ - тривалість імпульсу.

Таблиця 1.1.

№	П.2.3.1		П.2.3.2			П.2.3.3		
	Порт	Рівні	Порт	T [мкс]	τ [мкс]	SDA	SCL	
						Порт	Порт	T [мкс]
1	P2	01011100b	P1.1	100	30	P1.2	P2.1	200
2	P3	0FFh	P2.2	250	10	P2.1	P1.5	250
3	P0	11100011b	P3.7	50	49	P0.7	P0.4	100
4	P2	01010101b	P2.2	300	20	P1.5	P3.1	70
5	P1.2	“L”	P0.3	500	40	P2.4	P2.0	75
6	P3	11001100b	P1.1	100	90	P1.7	P2.3	230
7	P2	00000001b	P2.5	1000	100	P1.3	P3.2	105
8	P3.1	“1”	P2.4	320	200	P2.2	P0.1	110
9	P1	10100000b	P1.1	200	100	P2.6	P1.2	211
10	P1	10D	P0.0	132	14	P1.5	P3.7	39
11	P3	01010000b	P1.3	190	121	P0.3	P2.2	244
12	P2	00000000b	P2.2	120	103	P2.4	P1.3	222

Таблиця 1.2

№	П.2.3.4	П.2.3.5		П.2.3.6		
	Порт	Порт	$T_{ДР}$ [мс]	$T_{И}$ [мкс]	Регістрація імпульсу	
					Початок	Закінчення
1	P2	P1.1	1,2		P2.2	P2.3
2	P3	P2.2	2,3	~100	INT0	INT1
3	P0	P3.7	3,5	~200	P2.3	P0.1
4	P2	P2.2	3,0	~230	T1	T1
5	P1.2	P0.3	5,0	<30	INT1	INT0
6	P3	P1.1	1,8	<203	T0	T0
7	P2	P2.5	3,5	<249	P3.2	P3.3
8	P3.1	P2.4	3,2	<212	P3.0	P3.2
9	P1	P1.1	2,0	<118	P3.2	P3.0
10	P1	P0.0	1,3	<192	INT1	INT0
11	P3.0	P1.3	1,9	~232	P0.0	P0.1
12	P2	P2.2	2,0	<187	INT0	INT1

4. Зміст звіту

- Титульний лист з назвою виконаної роботи і склад бригади;

- Текст програми з коментарями.

5. Контрольні питання

- Визначити основні типи цифрових сигналів.
- Які особливості організації виведення статичних сигналів?
- Як реалізується виведення періодичних імпульсних сигналів з використанням системи переривань?
- Які особливості виведення цифрових імпульсних сигналів у послідовному коді з формуванням імпульсів синхронізації?
- Які особливості організації введення статичних сигналів?

6. Теоретичні відомості

Мікропроцесор AT89C51 сімейства MCS-51 має чотири 8 - бітні паралельні двонаправлені порти P0 - P3, що дозволяють вводити і виводити цифрові сигнали. Кожен порт містить регістр-фіксатор і драйвер. Драйвери портів P1 - P3 виконані за схемою з вбудованим навантаженням. Їх вихідний стан визначається інформацією, що записана в регістр-фіксатор. Драйвер порту P0 виконаний за схемою з відкритим стоком. Якщо P0 використовується як порт загального призначення для введення/виведення цифрових сигналів, то до ліній порту необхідно підключити зовнішнє навантаження (між виходом порту і шиною +5В). Регістри - фіксатори портів P0 - P3 є регістрами SFR області з доступом як на байтовому, так і бітовому рівнях. Частина ліній одного порту може використовуватися як приймачі, у той час як інші лінії - як передавачі цифрових сигналів.

Вихідні драйвери портів P0 і P2, а також вхідний буфер порту P0 використовуються при звертанні до зовнішньої пам'яті (ЗП). При цьому через порт P0 в режимі тимчасового мультиплексування спочатку виводиться молодший байт адреси ЗП, а потім видається або приймається байт даних. Через порт P2 виводиться старший байт адреси в тих випадках, коли розрядність адреси дорівнює 16 біт.

Всі виводи порту P3 можуть бути використані для реалізації альтернативних функцій, перерахованих в табл. 1.3. Альтернативні функції можуть бути задіяні шляхом запису 1 у відповідні біти регістра-фіксатора (P3.0 - P3.7) порту 3.

Таблиця 1.3.

Символ	Позиція	Ім'я і призначення
RD	P3.7	Читання. Активний сигнал низького рівня формується апаратно при зверненні до ВПД
WR	P3.6	Запис. Активний сигнал низького рівня формується апаратно при зверненні до ВПД
T1	P3.5	Вхід таймера/лічильника 1 або тест-вхід
T0	P3.4	Вхід таймера/лічильника 0 або тест-вхід
INT1	P3.3	Вхід запиту переривання 1. Сприймає сигнал низького рівня або зріз
INT0	P3.2	Вхід запиту переривання 0. Сприймає сигнал низького рівня або зріз
TXD	P3.1	Вихід передавача послідовного порту. Вихід синхронізації в режимі регістра зсуву.
RXD	P3.0	Вхід приймача послідовного порту. Введення/вивід даних в режимі регістра зсуву.

Порт P0 є двонаправленим, а порти P1, P2 і P3 - квазідвонаправленими. Кожна лінія портів може бути використана незалежно для введення або виведення інформації. Для того, щоб деяка лінія порту використовувалася для введення, в D-триггер регістра-фіксатора порту повинна бути записана 1, яка закриває МОП-ТРАНЗИСТОР вихідного ланцюга.

По сигналу початкове установлення («RESET») в регістри-фіксатори всіх портів автоматично записуються одиниці, що настроюють їх тим самим на режим введення.

Всі порти можуть бути використані для організації введення/виводу інформації по двонаправлених лініях передавання. Проте порти P0 і P2 не можуть бути використані для цієї мети у випадку, якщо МК-система має зовнішню пам'ять, зв'язок з якою організовується через загальну шину адреси/данних, що працює в режимі часового мультиплексування.

ВИВЕДЕННЯ ЦИФРОВИХ СИГНАЛІВ

Особливістю мікропроцесорів сімейства MCS-51 є те, що не потрібно настроювання портів на режим передавання. При ініціалізації ліній портів, що працюють у режимі передавання, визначаються лише їх початкові вихідні рівні "0"/"1".

Виведення статичних цифрових сигналів

Для установлення окремих ліній порту в "1" або скидання в "0" – використовуються команди ассемблера

SETB bit, CLR bit; де "bit" - біт призначення.

Якщо весь порт використовується в режимі передавача, і одночасно змінюється біт, то доцільно використовувати команди завантаження байта.

```
MOV P1, # data; "data" ;безпосередні дані, або символічне ім'я, привласне-
                ;не константі.
```

Досить часто, при виведенні сигналів по ряду ліній порту, застосовують прийом маскування. Якщо необхідно скинути окремі лінії порту, то використовують логічну операцію "І"

ANL <port>, #data ; де "port" - порт призначення.

При виконанні такої операції ті лінії порту, яким у розрядах числа "data" відповідають нульові значення, скидаються.

Для установлення в "1" ліній порту застосовують операцію "АБО"

ORL <port>, #data.

При виконанні цієї команди лінії порту, яким у розрядах числа "data" відповідають одиничні значення, встановлюються в "1".

Приклад. *Завдання.* Завантажити константи безпосередньо на лінії порту (P3.1, P3.2) і на лінії портів, які мають символічні імена біт (OUT_1, OUT_2).

; Опис констант і змінних

```
OUT_1 bit P3.3; символічне ім'я третього вивода порту P3
OUT_2 bit P3.4; символічне ім'я четвертого виводу порту P3
```

```
Const_1 equ 00110011b; символічне ім'я константи і її значення
```

; Програма ініціалізації регістрів мікроконтролера

```
Org 0H ; адреса вектора рестарту після пуску процесора
        SJMP INIT ; перехід на блок ініціалізації мікропроцесора
ORG 10H                    ; початкова адреса блоку ініціалізації
INIT: SETB P3.1            ; лінія порту P3.1 встановлена в "1" стан
        CLR P3.2           ; лінія порту P3.2 скинута в "0" стан
        SETB OUT_1         ; лінія порту P3.3 встановлена в "1" стан
        CLR OUT_2          ; лінія порту P3.4 скинута в "0" стан
        MOV P1, #0Fah      ; передавання в порт P1 числа 0Fh
        MOV P2, #const_1   ; передавання в порт P2 числа з ім'ям const_1
        MOV P3, #00001111b ; завантаження числа у порт P3
        ANL P3, #11000011b  ;результат операції "I" - 00000011b
        MOV P3, #00001111b ; завантаження числа у порт P3
        ORL P3, #11000011b  ; результат операції "АБО" - 11001111b
END                        ;директива ассемблера про закінчення програми
```

Виведення імпульсних періодичних сигналів

При виведенні імпульсних періодичних сигналів доцільно використовувати таймери/лічильники і систему переривань. Один з таймерів/лічильників організує формування періоду повторення сигналу, а другий - його тривалість.

Таймер/лічильник, що забезпечує підтримку періоду сигналу, працює безупинно. При його переповненні викликається підпрограма переривань, що запускає другий таймер/лічильник. У цій же підпрограмі встановлюється вихідний сигнал. При переповненні другого таймера/лічильника, відповідно, викликається друга підпрограма переривань, де скидається рівень вихідного сигналу і виключається другий таймер/лічильник. Для підтримки незмінного періоду повторення і тривалості вихідного сигналу необхідно в підпрограмах переривань перезавантажувати рахункові регістри TH0(1), TL0(1), або використовувати режим роботи таймерів/лічильників з автоперезавантаженням (режим

2). При визначенні констант завантаження рахункових регістрів таймерів/лічильників виходять з того, що рахункові імпульси надходять на вхід таймера з частотою f_{OSC} /12.

Приклад. *Завдання.* Вивести на вивід мікроконтролера імпульсний періодичний сигнал тривалістю 50 мкс і періодом повторення 100 мкс при частоті роботи генератора мікропроцесора f_{BQ} = 12 Мгц.

```
; Опис констант і змінних
ini_TMOD equ 00100010b ; 2 режим роботи T/C0 і T/C1.
ini_TC0 equ (0FFh - 100); задання періоду в нульовому таймері
ini_TC1 equ (0FFh - 50) ; задання тривалості імпульса в першому таймері
OUT bit P1.1 ; лінія порту на якому формується сигнал

; Програма
org   0H              ; адреса вектора рестарту після пуску процесора
      sjmp INIT       ; перехід на блок ініціалізації мікропроцесора
org   0BH             ; адреса вектора переривань таймера/лічильника TC0
      sjmp INT_TC0    ; перехід на підпрограму переривань від TC0
org   1BH             ; адреса вектора переривань таймера/лічильника TC1
      sjmp  INT_TC1   ; перехід на підпрограму переривань від TC1
org   20H             ; початкова адреса блоку ініціалізації
INIT: CLR OUT         ; скидання вихідного сигналу
      MOV TMOD, #ini_TMOD ; ініціалізація таймерів/лічильників
      MOV TH0, #ini_TC0 ;завантаження рахункового регістра таймера T/C0
      MOV TL0, #ini_TC0 ;завантаження регістра перевантаження таймера T/C0
      MOV TH1, #ini_TC1 ;завантаження рахункового регістра таймера T/C1
      MOV TL1, #ini_TC1 ;завантаження регістра перевантаження таймера T/C1
      SETBET0   ;дозвіл переривань від T/C0
      SETBET1   ;дозвіл переривань від T/C1
```

```
SETB EA     ;загальний дозвіл переривань
SETB TR0    ;включення таймеру/лічильник T/C0

;Головна програма
MAI: SJMP MAI                ; зациклення головної програми
;Підпрограма переривань від таймера - генератора періоду
INT_TC0: SETB OUT ; установка в "1" стан вихідного імпульсу
         SETB TR1 ; включення нульового таймера
         RETI     ; повернення із переривання

;Підпрограма переривань від першого таймера
INT_TC1: CLR OUT  ; скидання вихідного імпульсу
         CLR TR1  ; вимикання першого таймера
         RETI     ; повернення із переривання
END                  ; директива ассемблера закінчення програми
```

Виведення у послідовному коді імпульсних цифрових сигналів з формуванням імпульсів синхронізації по кожному переданому біту

При виведення імпульсних цифрових сигналів, які супроводжуються імпульсами синхронізації, необхідно забезпечити:

- початковий і кінцевий рівні цифрових сигналів у лінії;
- задану швидкість передавання окремих біт;
- порядок передавання біт (з якого біта починається передавання - старшого чи молодшого);
- часові співвідношення між сигналами даних і синхронізації.

Для передавання сигналів у таких випадках застосовують дві лінії SDA (Serial Data) і SCL (Serial Clock). Типові часові діаграми наведені на рис. 1, де прийняті такі позначення: T - період повторення синхроімпульсів; δ - тривалість синхроімпульсів; t_1 - затримка між моментом установлення сигналу по лінії SDA і фронтом імпульсу SCL.

У наведеному прикладі по лінії передається байт 01011101b (починаючи з молодшого біта). Для завдання швидкості обміну можуть використовуватися таймер/лічильник разом із системою переривань. Малі часові затримки між сигналами SDA і SCL можуть формуватися програмним шляхом, з використанням порожньої операції (команда NOP) або підпрограм затримки.

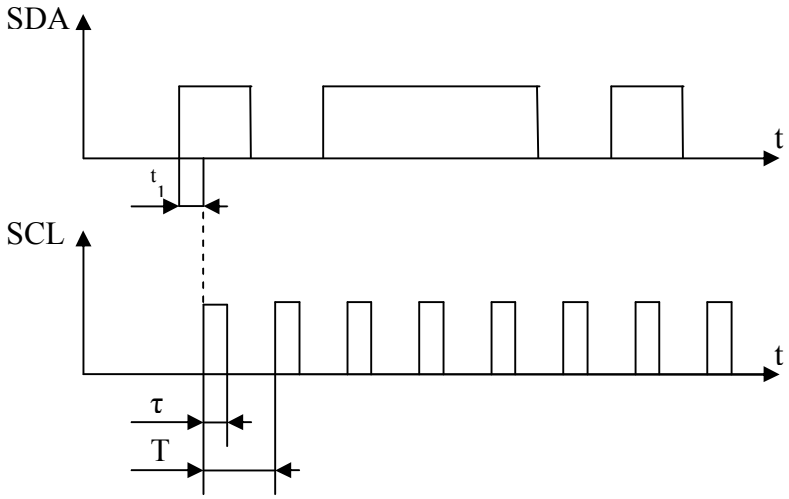


Рис. 1.1

Приклад. *Завдання.* Передати байти, починаючи з молодшого біта, з періодом передавання повторення сигналу 100 мкс. Частота генератора процесора 12 Мгц. Тривалість імпульсу синхросигналу складає 1 мкс.

```
; Опис констант і змінних
ini_TMOD equ 00000010b ; організація другого режиму роботи T/C0 .
ini_TC0 equ (0FFh - 100) ; задання періоду SCL в нульовому таймері
SDA bit P1.1            ; лінія порту на якому формується сигнал SDA
SCL bit P1.2            ; лінія порту на якому формується сигнал SCL
SERIAL equ 25           ;значення переданого числа

; Програма
org 0H                  ;адреса вектора рестарту після пуску процесора
```

```
        sjmp INITIAL       ; перехід на блок ініціалізації мікропроцесора
org 0BH                    ;адреса вектора переривань таймера/лічильника TC0
        sjmp INTER_TC0 ;перехід на підпрограму переривань від TC0
org20H                     ; початкова адреса блоку ініціалізації
INITIAL: CLR SDA ;скидання вихідного сигналу SDA
            CLR SCL  ;скидання вихідного сигналу SCL
            MOV A, #SERIAL ;завантаження в акумулятор переданого числа
            MOV COUNTER, #9;завантаження лічильника переданих біт
            MOV TMOD,#ini_TMOD ;ініціалізація таймерів/лічильників
            MOV TH0,#ini_TC0;завантаження старшого регістра T/C0
            MOV TL0,#ini_TC0 ;завантаження молодшого регістра T/C0
            SETB ET0 ; дозвіл переривань від T/C0
            SETB EA ; загальний дозвіл переривань
            SETB TR0 ; включення таймеру/лічильника T/C0
```

;Головна програма

```
MAI:    SJMP MAI ; зациклення головної програми
```

;Підпрограма переривань від таймера - генератора періоду

```
INTER_TC0:  DEC COUNTER     ;модифікація лічильника переданих біт
            JNZ INTER_TXD   ; розгалуження на передачу
            CLR TR0         ;якщо всі біти передані - вимикання таймера
            CLR SDA         ;скидання вихідного сигналу
            SJMP INTER_END; перехід на вихід із підпрограми
INTER_TXD: RRC A            ; виділення молодшого біта
            MOV SDA, C      ; передавання його у вихідний порт SDA
            SETB SCL        ; формування синхроімпульсу
            CLR SCL
INTER_END: RETI             ; повернення із переривання
END                         ; директива закінчення програми
```

ВВЕДЕННЯ ЦИФРОВИХ СИГНАЛІВ

Особливістю мікропроцесорів сімейства MCS-51 є необхідність настроювання портів на режим приймання. При ініціалізації ліній портів, що працюють у режимі приймання, у відповідні біти SFR регістрів портів записується "1".

Наприклад, ініціалізацію лінії порту P3.1 на режим приймача можна виконати таким чином: SETB P3.1.

Побудова алгоритмів введення цифрових сигналів залежить від їхнього типу (статичні та імпульсні), а також від наявності флуктуацій форми сигналу. Розрізняють алгоритми введення сигналів зі стабільною формою (наприклад, від конфігураційного поля джамперів), а також сигналів, що характеризуються флуктуаціями. Характерним прикладом у цьому випадку може бути введення сигналу від контактної пари. У цьому випадку в сигналі існує інтервал часу брязкоту контактів, де логічний рівень може кілька разів змінюватися.

При введенні імпульсних сигналів характерними є два типи задач:

- Реєстрація числа сигналів, що надійшли;
- Вимірювання основних часових характеристик, періоду або тривалості.

Введення статичних цифрових сигналів

Для забезпечення введення статичних цифрових сигналів необхідно ініціалізувати порт на режим приймання і потім передати інформацію з порту за адресою призначення.

Якщо відбувається приймання інформації з однієї лінії порту, то може використовуватися команда передавання в бітовий акумулятор "C": MOV C, P3.1. При введенні даних із усього порту може використовуватися інструкція передавання в акумулятор або регістр внутрішньої RAM області мікропроцесора: MOV A,P3; MOV 22,P2.

Мікроконтролери сімейства MCS-51 допускають виконання ряду логічних інструкцій, команд розгалуження з використанням SFR регістрів портів і, у ряді випадків, окремих бітів портів.

Наприклад, JB bit, rel, де rel – відносна адреса переходу; JNB P3.2,METKA_1; INC port; DEC P3; ANL port, data; ORL P2,#1Fh.

Введення цифрових сигналів, зформованих з використанням контактної пари (із усуненням брязкоту контактів)

Приклад. *Завдання.* Організувати затримку на 2,3 мс, яка б забезпечувала усунення брязкоту контактів при введенні цифрових сигналів, що зформовані з використанням контактної пари. Частота генератора процесора 12 Мгц.

```
; Опис констант і змінних
ini_TMOD equ 00000001b ; задаємо 1-й режим таймера/лічильника T/C0
ini_TC0 equ (0FFFFh - 2300) ; константа ініціалізації T/C0
; Програма
Org 0H                  ; адреса вектора рестарту після пуску процесора
    SJMP INITIAL   ; перехід на блок ініціалізації
Org 0BH                 ; адреса вектора розгалуження
    SJMP INT_TC0 ;перехід на підпрограму переривання TC0
org20H                  ; початкова адреса блоку ініціалізації
INITIAL: MOV TMOD, #ini_TMOD ;ініціалізація T/C0
         MOV TH0, #HIGH(ini _TC0) ; ініціалізація роботи старшого біта T/C0
         MOV TL0, #LOW(ini_TC0) ; ініціалізація роботи молодшого біта T/C0
         SETB P2.2 ; ініціалізація P2.2 на приймання даних
         SETB ET0; дозвіл переривання від T/C0
         SETB EA ;загальний дозвіл переривань
; Головна програма
Main:  JNB   P2.2, Main ;зациклення програми до приходу імпульсу
       SETB TR0         ;включення T/C
M1:    SJMP M1 ;зациклення програми до переповнення лічильника і виклик
                ; переривання
INT_TC0: MOV C, P2.2 ;перенесення даних з P2.2 у біт переносу
END                     ; директива закінчення програми
```

Вимірювання тривалості імпульсних цифрових сигналів

Для вимірювання тривалості імпульсних цифрових сигналів використовується таймер і система переривань. При реєстрації початку імпульсу таймер запускається, а по завершенню імпульсу - зупиняється. Тривалість імпульсу визначається по величині часу одного циклу мікропроцесора і по підрахованому таймером числу циклів.

Приклад. *Завдання.* Виміряти тривалість імпульсних цифрових сигналів. Для реєстрації початку і кінця імпульсу використовувати входи INTO, INT1, по спадаючому фронту сигналів. Для забезпечення спрацьовування різних переривань, по наростаючому і спадаючому фронтам, використовувати інвертор.

```
; Опис констант і змінних
ini_TMOD equ 00000010b ; задання 2-го режимуТ/С0
ini TCO equ 0          ; константа ініціалізації Т/С0
; Програма
org 0H                 ;адреса вектора початку програми
    sjmp INIT          ;перехід на блок ініціалізації
org 20H                ;адреса блоку ініціалізації
INIT: mov TMOD,#ini_TMOD  ;завантаження константи в таймер-лічильник
    mov TLO, #ini_TCO     ;початкове значення таймера-лічильника
    setb INTO             ;установка одиниці на INTO
; Головна програма
main:  Jb  INT0, main ;очікування приходу "1" на INTO (початок імпульсу)
    setb  TR0         ; включення таймера-лічильника
m1:  jnb INTl,ml      ;очікуння приходу "1" на INT1 (закінчення імпульсу)
    clr TR0           ; вимикання таймера-лічильника
    mov A, TL0    ;перенесення в акумулятор кількості рахункових імпульсів
    mov TLO, #ini_TC0  ; установлення початкового значення таймера
    sjmp main         ; перехід на мітку головної програми
End                ; директива закінчення програми.
```

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 2

ДОСЛІДЖЕННЯ UART-КОНТРОЛЕРА МІКРОПРОЦЕСОРА AT89C51

1. Мета практикуму

- Ознайомитися з основними властивостями UART-контролера мікропроцесора AT89C51;
- Освоїти програмування UART-контролера.

2. Програма практикуму

2.1. Вивчити алгоритмічні основи програмування UART-контролера;

2.2. У програмному середовищі μ-VISION/51 написати програми обміну інформацією з застосуванням UART-контролера з використанням і без застосування системи переривань. Провести асемблювання програм. Відкоригувати помилки;

2.3. З використанням програмного симулятора dScope-51 покроково виконати програму. Простежити за зміною даних у відповідних регістрах SFR.

3. Завдання до практикуму

У табл. 2.1 наведені вихідні дані для виконання практикуму.

Таблиця 2.1

№	$f_{ВО}$ [МГц]	Приймання				Передавання				Додаткові умови	
		A/C	BR [кбод]	P	N	A/C	BR [кбод]	P	N	Кількість байт, що передаються	Кількість байт, що приймаються
1	12	A	9,6	-	-	A	4,8	P		2	4
2	6	A	2,4	-	-	A	4,8		N	4	6
3	11,059	A	4,8	-	-	A	9,6	P		5	2
4	4	C	*	-	-	A	2,4		N	4	2
5	10	A	2,4	P	-	C	*	-	-	2	3
6	12	A	9,6	-	N	C	*	-	-	3	6
7	8	C	*	-	-	A	2,4		N	3	2
8	7	A	2,4	-	-	A	9,6	P		3	7
9	15	A	19,2	-	-	A	4,8	P		4	5
10	13	A	9,6	P	-	C	*	-	-	2	3
11	14,5	A	4,8	-	N	C	*	-	-	6	4
12	12	A	9,6	-	-	A	2,4	P		5	3

де: A/C – режим обміну асинхронний (A), синхронний (C); * – швидкість розрахувати самостійно; BR – швидкість обміну; f_{BQ} - частота генератора мікропроцесора; P - контроль парності, N - контроль непарності.

4. Зміст звіту

4.1. Титульний лист з назвою практикуму і складом бригади;

4.2. Текст програми з коментарями.

5. Контрольні питання

- Які основні режими роботи UART-контролера?
- Які властивості має протокол обміну?
- Які регістри SFR використовуються для керування UART-контролером?
- Яка особливість використання прапорців RI, TI?

6. Теоретичні відомості

Контролер послідовного інтерфейсу UART (Universal Asynchronous Receiver Transceiver) або він ще називається універсальним асинхроним приймо-передавачем (УАПП) призначений для організації введення-виведення послідовних даних. Зазвичай його використовують для підключення до МК периферійного обладнання, що вимагає послідовного зв'язку (модем, радіотехнічні пристрої, системи зв'язку з іншими МК або комп'ютерами на великій відстані - системи комп'ютерних мереж). Головне призначення УАПП - перетворення паралельної інформації, яка циркулює всередині МК, в послідовну інформацію на спеціальному зовнішньому виводі МК і навпаки.

Дозволяє вести асинхронний/синхронний обмін даними як у повному дуплексному (одночасне приймання і передавання даних), так і симплексному (або приймання, або передавання даних) режимах.

Для організації обміну між двома UART-контролерами необхідно узгодити режими їхнього використання – швидкість обміну, тип представлення даних у посилках та тип синхронізації передавача і приймача.

Структура контролера

Спрощена структурна схема контролера наведена на рис. 2.1. До його складу входять приймальний і передавальний регистри зсуву, буферні регістри приймача і передавача, схема керування інтерфейсом та регістри керування. Зсувні регістри передавача і приймача забезпечують формування послідовних даних на зовнішніх шинах контролера і їхнє зворотне перетворення з послідовного в паралельній вид.

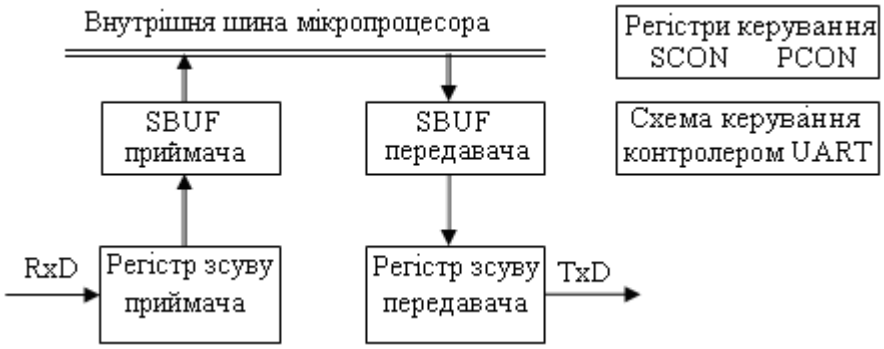


Рис. 2.1.

Буферні регістри приймача і передавача мають загальне ім'я SBUF та одну адресу в карті SFR і в той же час це різні регістри SFR області. Вони забезпечують побайтовий обмін інформацією між внутрішньою шиною даних мікропроцесора і регістрами зсуву інтерфейсу. Схема керування інтерфейсом забезпечує формування сигналів керування контролера в різних режимах роботи. Для керування режимами роботи UART-контролера використовуються два регістри SFR області - SCON і PCON.

При передаванні байт, що передається по внутрішній шині мікропроцесора записується у буфер передавача. Відповідно до заданого режиму роботи,

дані у послідовному двійковому коді, починаючи з молодшого значущого розряду, передаються на вивід TxD мікроконтролера.

При прийманні, під впливом схеми керування контролера, дані, що надходять на вхід RxD мікропроцесора, у послідовному двійковому коді, починаючи з молодшого розряду, заповнюють регістр зсуву приймача. Потім прийнятий байт передається в буфер приймача. Послідовний порт може приймати черговий байт, навіть якщо прийнята до цього інформація не була прочитана з регістра приймача. Однак, якщо до закінчення приймання, байт, що знаходиться в регістрі приймача не буде прочитаний, прийнятий байт втрачається.

Програмний доступ до регістрів приймача і передавача здійснюється звертанням до регістру спеціальних функцій SBUF. При записі в SBUF байт завантажується в регістр передавача, а при читанні SBUF байт читається з регістра приймача.

Регістри керування контролером

Регістр керування SCON (Serial Control) є біт-адресованим регістром SFR, доступним для запису та читання. Структура регістру наведена у табл. 2.2.

Відповідність між станом бітів SMO, SM1 і режимами роботи UART наведена у табл. 2.3.

Регістр керування PCON (Power Control) є байт адресованим регістром SFR, доступним для запису та читання. Його бітова структура наведена в табл. 2.4. Така структура відповідає мікропроцесорам, виконаним за CMOS-технологією (87C51, AT89C51). У мікропроцесорах NMOS-типу (8051) існує тільки один біт керування – SMOD.

Для керування режимами роботи UART контролера використовується тільки біт SMOD. При SMOD = 1 швидкість обміну в режимах 1, 2, 3 подвоюється. Біти GFO, GF1 використовуються як прапорці загального призначення, а PD, IDL - для керування режимами холостого ходу та зниженого споживання енергії.

Таблиця 2.2

Біт	№ біту	Функція
SMO	SCON.7	Біт задання режиму роботи UART-контролера
SM1	SCON.6	Біт задання режиму роботи UART-контролера
SM2	SCON.5	Біт дозвілу багатопроцесорної роботи. У нульовому режимі SM2 = "0". У другому та третьому режимах при SM2 = 1 прапорець RI = "0", якщо дев'ятий прийнятий біт даних дорівнює "0". У режимі 1 при SM2 = 1 прапорець RI = "0", якщо прийнятий стоп-біт, рівний "1".
REN	SCON.4	Біт дозвілу приймання послідовних даних. Встановлюється і скидається програмою.
TB8	SCON.3	Дев'ятий біт переданих даних у режимах 2 і 3. Встановлюється і скидається програмою.
RB8	SCON.2	Дев'ятий біт прийнятих даних у режимах 2 і 3. У режимі 1 при SM2 = 0 RB8 є прийнятим стоп-бітом. У режимі 0 не використовується.
TI	SCON.1	Прапорець переривання передавача. Скидається програмно. Встановлюється апаратно наприкінці часу видачі 8-го біта в нульовому режимі і на початку стоп-біту в інших режимах.
RI	SCON.0	Прапорець переривання приймача. Скидається програмно. Встановлюється апаратно наприкінці часу приймання 8-го біту в режимі 0, через половину інтервалу стоп-біту в режимах 1, 2, 3 при SM2 = 0. При SM2 = 1 робота цього прапорця наведено в описі біту SM2.

Таблиця 2.3

SMO	SM1	Режим	Назва		Швидкість передавання
0	0	0	Синхронний регістр зсуву		f_{BQ} /12
0	1	1	асинхронний	8-бітовий приймач-передавач	Змінна і задається таймером
1	0	2		9-бітовий приймач-передавач	f_{BQ} /64 або f_{BQ} /32
1	1	3		9-бітовий приймач-передавач	Змінна і задається таймером

Таблиця 2.4

SMOD	-	-	-	GF1	GFO	PD	1DL

Режими роботи контролера

Контролер UART може використовуватися в 4 режимах роботи "0 - 3", які відрізняються типом обміну (асинхронний/синхронний), а також протоколом обміну.

Протоколом обміну називається зведена характеристика, що визначає швидкість обміну і спосіб упакування переданих байт.

Швидкість обміну

Швидкість обміну може бути фіксованою (режими 0, 2), або змінною (режими 1, 3). У режимах роботи з фіксованою швидкістю частота посилки біт визначається частотою роботи генератора мікропроцесора і станом біта подвоєння швидкості SMOD (режим 2). Одиницею виміру швидкості є число переданих біт за секунду [біт/с] = [бод].

У режимі 0 швидкість обміну BR (Baud Rate) фіксована

$$BR = f_{BQ} /12 \text{ (біт/с)}.$$

У режимі 2 швидкість обміну BR також фіксована

$$BR = (2^{SMOD}/64) * f_{BQ} \text{ (біт/с)}.$$

У режимах 1, 3 швидкість обміну BR змінна, задається відповідним настроюванням таймера/лічильника T/C1 і станом біта SMOD

$$BR = (2^{SMOD}/32) * Fovf \text{ T/C1 (біт/с)}$$

Fovr T/C1 – частота переповнень T/C1.

Зазвичай для синхронізації послідовного порту таймер T/C1 включається в режим автозавантаження (режим 2).

У цьому випадку швидкість послідовного обміну визначається так:

$$BR = (2^{SMOD} * f_{BQ})/(32 * 12 * [256 - TH1)]) \text{ [біт/с]},$$

де TH1 – константа завантаження регістра TH1 таймера T/C1.

У табл. 2.5 наведені приклади реалізації деяких протоколів обміну.

Таблиця 2.5

Режим UART	BR [кбод]	f_{BQ} [МГц]	SMOD	Таймер Т/С1		
				Режим	С/Т	TH1
1,3	1,2	11,059	0	2	0	E8H
1,3	2,4	11,059	0	2	0	F4H
1,3	19,2	11,059	1	2	0	FDH
1,3	0,11	12	0	1	0	FEEBH

Протоколи обміну

Протоколи пакування біт у посилці

У *режимі 0* (синхроний обмін) UART-контролер працює як восьмиразрядний регістр зсуву. При цьому 8 біт інформації в послідовному коді приймаються і передаються через вивід RxD молодшим бітом уперед. На виводі TxD при цьому формуються сигнали синхронізації зсуву.

У *режимі 1* (асинхронного обміну) інформація приймається через порт RxD і передається через TxD.

Посилка пакується таким чином:

10 біт посилки = 1 стартовий біт + 8 біт даних (LSB перший) + 1 стоп-біт. При прийманні стоп-біт заноситься в біт RB8 регістра SCON.

У *режимах 2 і 3* (асинхронні обміни) інформація приймається через порт RxD і передається через TxD.

Протоколи упакування біт ідентичні:

11 біт посилки = 1 стартовий біт + 8 біт даних (LSB перший) + 1 біт контролю + 1 стоп-біт.

При передаванні біт контролю використовується біт TB8 регістра SCON. При прийманні біт контролю передається в біт RB8 регістра SCON.

В якості біт контролю може використовуватися біт парності/непарності переданих даних, що дозволяє підвищити захист каналу зв'язку.

Приклад програмування UART контролера

Завдання: Запрограмувати UART-контролер на асинхронний, цілком дуплексний обмін посилками без контролю парності/непарності зі швидкістю 2,4 кбод при частоті роботи генератора мікропроцесора f_{BQ} = 11,059 Мгц.

Так як обмін асинхронний, то нульовий режим не підходить для реалізації поставленної задачі. Зіставлення значень швидкості обміну і частоти роботи генератора мікропроцесора виключає використання 2 режиму, бо це обмін з фіксованою швидкістю. Для реалізації асинхронного режиму, зі змінною швидкістю можна використовувати перший та третій режими UART контролера. Але, так як обмін відбувається без контролю парності/непарності переданих даних, то доцільно використовувати 1 режим роботи контролера.

У даному режимі роботи для завдання швидкості обміну використовується таймер/лічильник Т/С1. Його доцільно використовувати в другому режимі, з автоперезавантаженням регістру TH1. Константу завантаження даного регістра можна визначити за формулою, наведеною вище або використовувати значення з табл. 1.5.

ТЕКСТ ПРОГРАМИ

;Опис констант і змінних

```
ini_P3 equ 00000011b ; константа ініціалізації порту Р3, задані
                     ; альтернативні функції по лініях TxD, RxD
ini_TMOD equ 00100000b ; константа ініціалізації таймера-лічильника,
                       ; заданий 2 режим роботи Т/С1
ini_SCON equ 00100000b  ; константа ініціалізації UART контролера,
                        ; заданий 1 режим роботи
ini_TC1 equ OF4h; константа завантаження рахункових регістрів таймера/
                ; лічильника Т/С1, яка впливає на швидкість UART
ini_PCON equ 0 ; константа ініціалізації регістра PCON.
```

; Програма ініціалізації регістрів мікроконтролера

```
org 0H ; адреса вектора рестарту після пуску процесора
        sjmp INIT ; перехід на блок ініціалізації мікропроцесора
org 10H       ; початкова адреса блоку ініціалізації
INIT: mov P3, #ini_P3 ; ініціалізація порту P3.
        mov TMOD,#ini_TMOD;задання режиму роботи таймера/лічильника T/C1
        mov TH1, #ini_TC1 ;завантаження рахункового регістра таймера T/C1
        mov TL1, #ini_TC1 ;завантаження регістра перевантаження таймера T/C1
        mov SCON, #ini_SCON ; задання режиму роботи UART контролера
        mov PCON, #ini_PCON ; скидання біта подвоєння швидкості обміну.
;(при скиданні процесора біт SMOD = 0 і останню команду можна виключити)
        setb TR1 ; включення таймера/лічильник T/C1
        setb REN ; дозвіл роботи приймача UART-контролера.
```

;Головна програма MAI:

```
MAI_RxD: jnb RI, MAI_ RxD; перевірка прапора завершення приймання
            call UART_RxD; при завершенні приймання, виклик підпрограми
обробки
MAI_TxD: jnb      Tl, MAI       ; перевірка прапора завершення передавання
            call UART_TXD     ; при завершенні - виклик підпрограми обробки
            sjmp MAI ; зациклення головної програми
```

;Підпрограма обробки завершення прийому

```
UART_RxD: clr RI ; скидання прапора приймача
             ret ; повернення з підпрограми
```

;Підпрограма обробки завершення передавання

```
UART_TxD: clr Tl; скидання прапора передавача
             ret ; повернення з підпрограми
            End    ; директива ассемблера про закінчення програми.
```

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 3

ВИВЕДЕННЯ І ВВЕДЕННЯ АНАЛОГОВИХ СИГНАЛІВ

1. Мета практикуму

• Ознайомитися з методами виведення і введення аналогових сигналів у мікроконтролерах з мікропроцесором AT89C51 і зовнішніми цифро-аналоговими та аналогово-цифровими перетворювачами;

• Освоїти алгоритми виведення і введення аналогових сигналів;

• Освоїти алгоритми згладжування флуктуацій аналогових сигналів, що вводяться.

2. Програма практикуму

2.1. Вивчити особливості виведення/введення аналогових сигналів;

2.2. У програмному середовищі μ-ViSiON/51 написати, ассемблировать і налагодити програми:

2.2.1. виведення статичного аналогового сигналу в мікроконтролері з мікропроцесором AT89C51 і зовнішнім цифро-аналоговим перетворювачем (ЦАП). Вибрати рівень опорної напруги. Визначити розрядність ЦАП. Зробити розрахунок виведеного коду;

2.2.2. виведення аналогового сигналу, що періодично змінюється і заданої форми. Частоту генератора вибрати самостійно;

2.2.3. введення аналогового сигналу в мікроконтролері з мікропроцесором AT89C51 і зовнішнім аналогово-цифровим перетворювачем (АЦП). Вибрати рівень опорної напруги. Визначити розрядність АЦП;

2.2.4. цифрової фільтрації флуктуацій введеного аналогового сигналу.

2.3. З використанням програмного симулятора dScope-51 пошагово виконати програми (п.п. 2.2.1, 2.2.3). Простежити за зміною даних у відповідних регістрах SFR.

3. Завдання до практикуму

У табл. 3.1 наведені вихідні дані для виведення аналогових сигналів, а в табл. 3.2 ─ для введення, де: U_{out} - вихідна напруга ЦАП; U_{in} - вхідна напруга АЦП; P8, I^2C - типи шин ЦАП/АЦП; δ - допустима похибка при виведенні/введенні аналогових сигналів; MAV, δ/N - типи алгоритмів цифрової фільтрації даних АЦП; N - число вимірів (див. теоретичні відомості).

Таблиця 3.1

№	П.2.2.1			П.2.2.2							
	Шина ЦАП	Порт	δ [%]	U_{out} [В]	Форма рис.	U_1 [В]	T_1 [мкс]	U_2 [В]	T_2 [мкс]	U_3 [В]	T_3 [мкс]
1	P8	P2	1,04	2,01	3.1.а	-	500	-	-	2,56	1000
2	I2C	P2	0,05	1,36	3.1.б	-	-	-	-	1,28	2000
3	I2C	P3	0,06	0,89	3.1.в	-	-	-	-	2,00	300
4	P8	P0	0,70	1,99	3.1.г	0,95	100	1,27	200	2,00	500
5	I2C	P2	0,50	2,34	3.1.д	-	-	1,00	200	2,00	300
6	P8	P1	0,62	4,20	3.1.е	1,00	1000	-	2000	2,00	3000
7	I2C	P3	0,80	2,72	3.1.ж	-	300	-	-	2,56	1000
8	I2C	P3	1,40	1,86	3.1.з	2,00	100	-	200	2,56	300
9	P8	P1	1,00	2,11	3.1.и	1,28	10000	-	20000	2,56	30000
10	P8	P1	2,45	4,66	3.1.к	2,00	500	-	1500	4,00	2000
11	I2C	P3	0,01	3,33	3.1.л	0,28	200	1,56	400	2,00	800
12	I2C	P2	0,02	2,67	3.1.м	0,34	1000	-	2500	2,30	3300

Таблица 3.2

№	П.2.2.3				П.2.2.4	
	Шина АЦП	Порт	U_{in} [В]	δ [%]	Алгоритм	N
1	I2C	P1	1,70	0,60	MAV	4
2	I2C	P2	1,86	0,10	Σ/N	64
3	P8	P1	2,31	1,00	MAV	8
4	P8	P2	4,16	0,45	Σ/N	16
5	I2C	P3	3,13	0,01	Σ/N	4
6	I2C	P2	2,37	0,02	Σ/N	2
7	P8	P2	2,01	0,50	MAV	16
8	I2C	P2	1,30	0,02	MAV	2
9	I2C	P1	0,99	0,80	Σ/N	8
10	P8	P0	1,59	1,40	Σ/N	16
11	I2C	P1	2,34	0,04	MAV	32

12	P8	P3	3,20	2,00	Σ/N	32

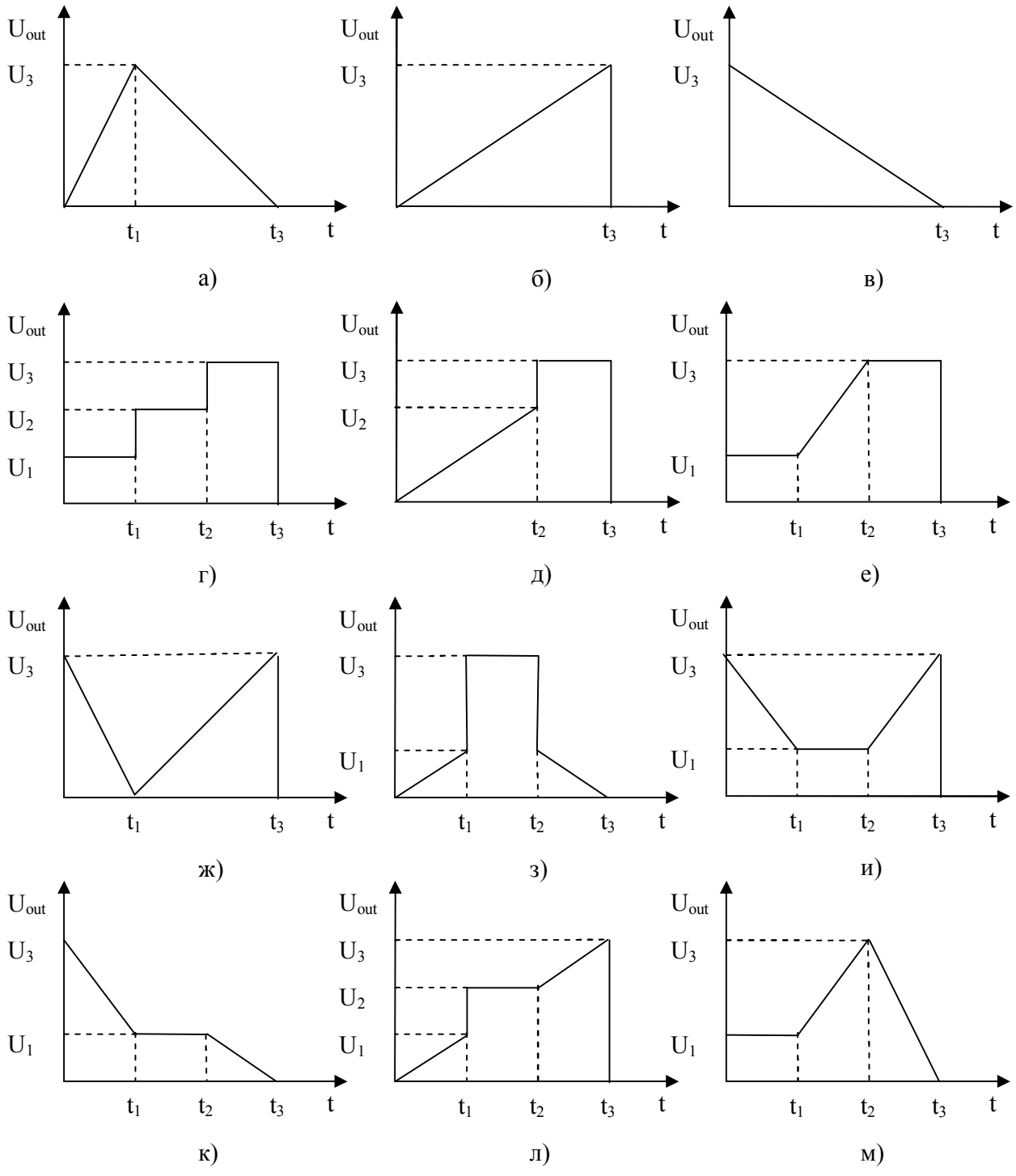


Рис. 3.1

4. Зміст звіту

- Титульний лист з назвою виконаної роботи і склад бригади;
- Текст програми з коментарями.

5. Контрольні питання

- Як реалізується виведення аналогових сигналів?
- Як реалізується виведення статичних аналогових сигналів?
- Як реалізується введення аналогових сигналів?

6. Теоретичні відомості

Мікроконтролер AT89C51 сімейства MCS-51 не має вбудованих цифро-аналогових і аналогово-цифрових перетворювачів. При побудові мікропроцесорних систем керування, у яких необхідно реалізувати такі функції перетворення, треба застосовувати зовнішні пристрої.

ВИВЕДЕННЯ АНАЛОГОВИХ СИГНАЛІВ

Для виведення аналогових сигналів використовуються наступні пристрої:

- Широтно-імпульсний модулятор з демодулятором;
- Цифровий потенціометр;
- Цифро-аналоговий перетворювач.

Широтно-імпульсний модулятор з демодулятором використовується в системах, де точність перетворення не є досить важливої, а важливими є компактність і мала вартість пристрою. Цифровий потенціометр можна представити як змінний резистор, у якого положення середнього виводу керується за допомогою цифрових сигналів. Цифрові потенціометри, що випускаються, мають від 32 до 256 стабільних станів. Відповідно, їх максимальна точність не перевищує 0,4%. Цифро-аналогові перетворювачі є найбільш точними пристроями.

Драйвер цифро-аналогового перетворювача

При використанні в мікроконтролерній системі периферійних пристроїв, необхідно підготувати підпрограму зв'язку такого пристрою з мікропроцесором.

Така програма є драйвером пристрою. При її створенні враховується тип використовуваного інтерфейсу і часові характеристики сигналів. Для обміну параметрами між основною програмою і підпрограмою призначаються регістри RAM області процесора.

Драйвер цифро-аналогового перетворювача із шиною P8

Структурна схема фрагменту мікроконтролерної системи наведена на рис.3.2, де MCU - мікропроцесор, DAC - цифро-аналоговий перетворювач. У даній схемі використовуються наступні лінії шини P8:

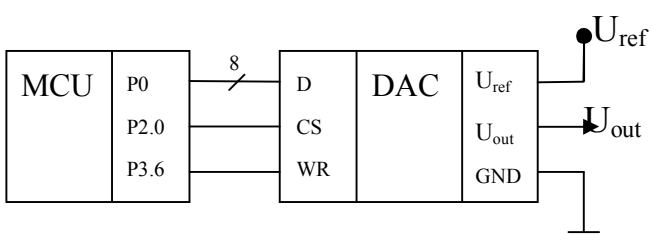


Рис.3.2

- 8 - розрядна шина даних (D0-D7);
- лінія передавання стробу запису даних (WR);
- лінія вибірки DAC (CS).

Для передавання інформації із шини P8 у мікропроцесорах сімейства MCS-51 можуть використовуватися команди MOVX @Ri, A і MOVX @DPTR, A, де Ri - регістри RAM - R0, R1. Перша команда дозволяє звертатися до зовнішньої пам'яті даних за 8-бітною, а друга - за 16-бітною адресою.

При використанні 16-бітної адресації молодший байт адреси встановлюється на шині даних. Для його фіксації в периферійному пристрої (регістрі) процесор генерує строб ALE (виведення процесора ALE). Старший байт адреси надходить на порт P2. Потім шина даних використовується для обміну даними з периферійним пристроєм. При передаванні даних із процесора генерується строб запису даних - WR. Часові діаграми, що відповідають передаванню інформації з мікропроцесора, показані на рис. 3.3.

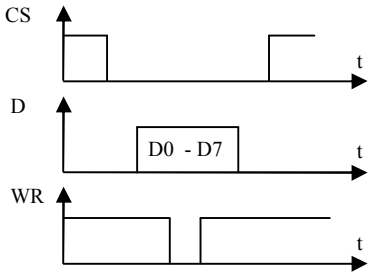


Рис. 3.3

Якщо число адресованих периферійних пристроїв не перевищує 8, то для їх адресації доцільно використовувати старший байт адреси. При цьому кожен розряд цього байту відповідає своєму периферійному пристрою. У даній схемі для адресації DAC використовується порт P2.0. У цей порт виводиться вміст регістра DPH покажчика адреси DPTR.

Вибірка DAC відбувається при надходженні цифрового сигналу CS низького рівня (рис. 3.3). Відповідно, для адресації DAC у регістр DPH необхідно записати константу 11111110B. Оскільки в структурній схемі відсутній фіксатор молодшого байта адреси, вміст регістру DPL можна не визначати.

Приклад підпрограми драйвера DAC із шиною P8

; Опис констант і змінних (заноситься в описову частину програми)

```
DAC_ADRES equ 11111110b; адреса вибірки DAC
DAC_DATA data 4H; регістр передавання даних у підпрограму
```

; Підпрограма

; В основній програмі, перед звертанням до даної підпрограми необхідно передати параметри в буферний регістр DAC_DATA

```
DAC_DRV_P8: MOV DPH, #DAC_ADRES; установлення адреси периферійного
                                ; пристрою
            MOV A, DAC_DATA     ; збереження переданих даних
            MOVX @DPTR, A       ; передавання даних у DAC
            RET                 ; повернення з підпрограми
```

Драйвер цифро-аналогового перетворювача із шиною I^2C

Структурна схема фрагменту мікроконтролерної системи наведена на рис.3.4. У даній схемі використовуються наступні елементи шини I^2C:

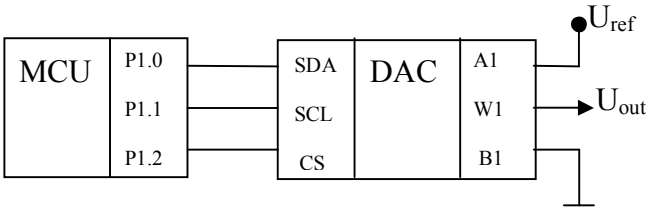


Рис.3.4

- SDA - лінія передавання даних;
- SCL - лінія передавання синхроімпульсів;
- CS - лінія вибірки DAC.

Часові діаграми основних сигналів шини I^2C наведені на рис. 3.5.

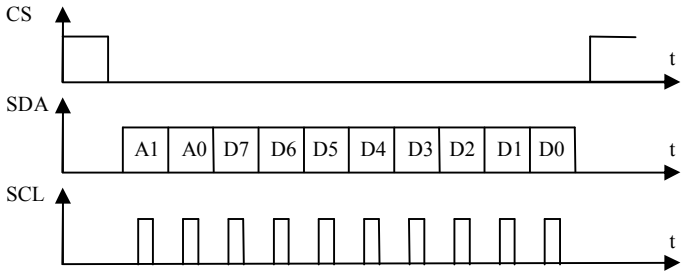


Рис. 3.5

Формат кадру даних (SDA) включає адресні біти (A1, A0) і біти даних (D7 - D0). Для різних типів мікросхем формат кадру може змінюватися. Адресні біти передаються в тому випадку, коли в одній мікросхемі знаходиться кілька однотипних пристроїв. Наприклад, мікросхема AD8404 містить 4 цифрових потенціометри. Дані можуть передаватися в різному упакуванні - починаючи з молодшого (LSB) або старшого (MSB) біта. У даному випадку обмін даними відбувається, починаючи зі старшого біта.

Приклад підпрограми драйвера DAC із шиною I^2C

; Опис констант і змінних (заноситься в описову частину програми)

```
SDA bit P1.0            ; лінія шини I2C
SCL bit P1.1            ; лінія шини I2C
CS_DAC bit P1.2         ; лінія вибірки DAC - "L"-активна
INI_P1 equ 00000100B ;константа ініціалізації порту P1,
                        ; CS_DAC = 1, SDA = 0, SCL = 0
DAC_DATA data 4H    ; регістр передавання даних у підпрограму
SDA_COUN data 5H    ; лічильник переданих біт
```

; Даний фрагмент заноситься в блок ініціалізації основної програми

MOV P1, #INI_P1 ; ініціалізація початкових станів ліній інтерфейсу

; Підпрограма - драйвер DAC із шиною I_2C

; В основній програмі, перед звертанням до даної підпрограми необхідно передати параметри в буферний регістр DAC_DATA

```
DAC_DRV_I2C: MOVA, DAC_DATA; завантаження переданих даних
             CLR CS_DAC ; установлення сигналу вибірки DAC
DAC_A1:      CLRSDA ; передавання старшого біта адреси
             CALL SCL_STROBE; виклик підпрограми формування стробу
DAC_A0:      CLR SDA; передавання молодшого біта адреси
             CALL SCL_STROBE; виклик підпрограми формування строба
             MOV SDA_COUN, #8;завантаження лічильника переданих біт
DAC_SDA: RLC A ; передавання старшого біту переданого байта в біт "C"
         MOV SDA, C ; передавання цього біту на лінію SDA
         DJNZ SDA_COUNT, DAC_SDA; перевірка закінчення передавання
          SETB CS_DAC; скидання сигналу вибірки DAC
          RET; повернення із підпрограми
```

; Підпрограма формування строба на лінії SCL (розміщається в області підпрограм, за межами головного циклу)

```
SCL_STROBE: SETB SCL; установлення сигналу строба
            NOP; затримка, яка формує тривалість стробу
            CLR SCL; скидання сигналу стробу
            RET; повернення із підпрограми
```

Виведення статичних аналогових сигналів

Для виведення аналогових сигналів за допомогою ЦАП необхідно вирішити наступні задачі:

- визначити рівень опорної напруги ЦАП;
- визначити розрядність перетворювача, виходячи з заданої точності;
- зробити розрахунок коду.

Цифро-аналоговий перетворювач формує вихідну напругу (U_{out}) у частках від опорної напруги (U_{ref}). Вихідна напруга ЦАП визначається за формулою:

$$U_{out} = (U_{ref} * K) / 2^n, \qquad (3.1)$$

де K – цифровий код, подаваний на ЦАП, n – розрядність перетворювача. Величина опорної напруги повинна перевищувати вихідну напругу. Звичайно величину опорної напруги вибирають кратними 2^n (1,28В, 2,56В, 1,024В).

Розрядність ЦАП вибирається виходячи з заданої точності перетворення. Звичайно зіставляють вагу молодшого розряду ЦАП (виражену у відсотках) з необхідною величиною точності (δ%). Так, вага молодшого розряду 8-розрядного перетворювача складає

$$P_{LSB} = 100 / 2^n = 100 / 256 = 0{,}39\%.$$

У технічних характеристиках перетворювачів звичайно наводять дані по нелінійності, виражені в *LSB* - одиницях - сумарному числу ваг молодшого розряду. Звичайно помилка знаходиться в межах 0,5 - 1 *LSB*.

Для розрахунку коду по відомим величинам розрядності ЦАП, опорній і вихідній напругах можна використовувати вираз (3.1).

Приклад програми формування аналогової напруги

Завдання. Сформувати аналогову напругу U_{out} = 2В, із джерелом опорної напруги U_{ref} = 2,56В, з точністю не гірше 1%.

Для досягнення цього можна застосувати 7-розрядний ЦАП. З урахуванням розрядності доцільно використати 8-розрядний перетворювач, наприклад MAX7224, що має шину P8. Вихідний код у даному випадку

$$K = 2{,}00*256/2{,}56 = 200.$$

У програму варто включити опис констант і змінних, блок ініціалізації і підпрограму драйвера I^2C.

;Опис констант і змінних (заноситься в описову частину програми)

```
CODE equ 200 ; значення вихідного коду
; Даний фрагмент передавання даних заноситься в основну програму
MOV DAC_DATA, #CODE ;завантаження коду в регістр передавання
                    ; даних у драйвер DAC
CALL DAC_DRV_P8 ; передавання коду в DAC
```

Виведення аналогових сигналів, що періодично змінюються

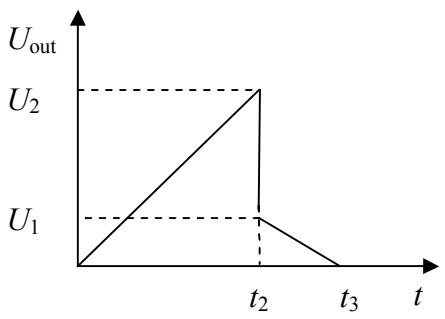
Для виведення періодичних аналогових сигналів необхідно вирішити наступні задачі:

- визначити рівень опорної напруги ЦАП;
- визначити розрядність перетворювача по заданій точності;
- визначити тривалість інтервалів незмінної аналогової напруги;
- зробити розрахунок кодів на цих інтервалах.

Для визначення рівня опорної напруги і розрядності перетворювача можна використовувати рекомендації, наведені вище.

При виведення аналогових сигналів, що періодично змінюються, для завдання інтервалів незмінності вихідної напруги ЦАП, доцільно використовувати таймери/лічильники і систему переривань.

Приклад. *Завдання.* Зформувати на виході мікроконтролера напругу, форма якої наведена на рис. Параметри напруги: U_1 = 1В; U_2 = 2В, t_2 = 20мс, t_3 = 30мс, частота генератора мікропроцесора 12Мгц, похибка $\delta < 1{,}0\%$.



Відповідно до рекомендацій, наведених вище, для формування напруги даної форми доцільно використовувати 8-розрядний ЦАП із джерелом опорної напруги U_{ref} = 2,56В.

На першому інтервалі часу 0 - 20мс вихідна напруга змінюється за 200 кроків у діапазоні 0 - 2 В. Тривалість інтервалу дискретизації складає

$$\Delta t_{02} = 20000\text{мкс}/200 = 100 \text{ мкс}.$$

На другому інтервалі 20 - 30 мс напруга змінюється від 1 до 0В за 100 кроків. Тривалість інтервалу дискретизації

$$\Delta t_{23} = 10000\text{мкс}/100 = 100 \text{ мкс}.$$

У даному випадку на першому і другому інтервалах можна генерувати переривання від таймера з однаковою тривалістю (t = 100 мкс.). Підпрограма переривань повинна забезпечити виведення вихідної напруги.

У зв'язку з тим, що мінімальна тривалість інтервалу дискретизації досить велика, у порівнянні з тривалістю циклу процесора, можна використовувати ЦАП, як із шиною Р8, так і I^2C. Тривалості циклу досить для організації виведення в обидвах випадках. Виберемо ЦАП із шиною I^2C.

Приклад селекції інтервалів вихідної напруги

В наведеному приладі використаний прийом прапорової маніпуляції. На першому інтервалі активізується прапор F_T_{02}, а на другому – F_T_{23}. У дану програму необхідно підключити фрагменти опису констант і змінних, блок ініціалізації і підпрограму драйвера I^2C, наведені в прикладі виведення статичного аналогового сигналу

```
; Опис констант і змінних
INI_TMOD equ 00000010b ; задається 2 режим роботи T/C0
INI_TC0 equ (0FFH - 100); задається інтервал дискретизації 100 мкс
F_T02 bit 20H.0 ; прапор інтервалу t0 - t2
F_T23 bit 20H.1 ; прапор інтервалу t2 - t3
U_OUT data 2H ; регістр вихідної напруги
U_0 equ 0 ; початкова напруга інтервалу T_02
U_1 equ 100 ; початкова напруга інтервалу T_23
U_3 equ (200 + 1) ; кінцева напруга інтервалу T_02
```

; Програма

```
ORG 0H              ; адреса вектора рестарту після пуску процесора
SJMP INI            ; перехід на блок ініціалізації мікропроцесора
ORG 0BH             ; адреса вектора переривань таймера/лічильника TC0
SJMP INT_TC0      ; перехід на підпрограму переривань від TC0
ORG 20H             ; початкова адреса блоку ініціалізації
INI: MOV TMOD, #INI_TMOD ; ініціалізація таймерів лічильників
MOV TH0, #INI_TC0 ;завантаження рахункового регістра таймера T/C0
MOV TL0, #INI_TC0 ;завантаження регістра перезавантаження T/C0
MOV U_OUT, #0       ; початкове скидання регістра вихідної напруги
SETB F_T23          ; ініціалізація прапора інтервалу,формування
                    ; сигналу почнеться з інтервалу t0 - t2
SETB ET0                : дозвіл переривань від T/C0
SETB EA                 ; загальний дозвіл переривань
SETB TR0                ; включення таймера-лічильника T/C0
```

;Головна програма

```
MAI: SJMP MAI           ; зациклення головної програми
```

;Підпрограма переривань від таймера - генератора інтервалу дискретизації

```
INT_TC0: JB F_T02, INT_T23 ;аналіз прапорів інтервалів і перехід
         JB F_T23, INT_T02 ; на видачу вихідної напруги
         SJMP INT_END
INT_T02: CJNE R2, #U_3, INT_T02_1 ; порівняння поточної вихідної напруги
                                  ; з граничним на даному інтервалі
         CLR F_T02; зміна прапорів інтервалів
         SETB F_T23
         MOV U_OUT, #U_1 ;установлення початкової напруги інтервалу T_23
         SJMP INT_END   ; перехід на видачу вихідної напруги
INT_T02_1: INC U_OUT        ; модифікація вихідної напруги на інтервалі T_02
```

```
        SJMP INT_END   ; перехід на видачу вихідної напруги
INT_T23: DJNZ U_OUT, INT_END ; модифікація і порівняння поточної
                              ; вихідної напруги з граничним на інтервалі
        CLR F_T23 ; зміна прапорів інтервалів
        SETB F_T02
        MOV U_OUT, #U_0;установлення початкової напруги інтервалу T_02
INT_END: MOV DAC_DATA, U_OUT ; передавання вихідної напруги в
                              ; буферний регістр драйвера
        CALL DAC_DRV_I2C ; виведення даних у DAC
        RETI             ; поверненя із переривання
END                      ; директива ассемблера про закінчення програми
```

Введення аналогових сигналів. Для введення аналогових сигналів можуть використовуватися убудовані в мікропроцесор або зовнішні аналогово-цифрові перетворювачі (АЦП, ADC). Зовнішні АЦП розрізняються за наступними основними ознаками: розрядність; число каналів; час перетворення; інтерфейс; наявність або відсутність джерела опорної напруги; точність.

Драйвер аналогово-цифрового перетворювача. При використанні в мікроконтролерній системі зовнішнього АЦП необхідно підготувати драйверну підпрограму. При її створенні, як і для ЦАП, враховується тип інтерфейсу і часові характеристики сигналів. Для обміну параметрами між основною програмою і підпрограмою призначаються регістри RAM області процесора.

Відмінністю АЦП, у порівнянні з ЦАП, є наявність ліній запуску перетворення і готовності його результатів. Час перетворення АЦП залежить від рівня вимірюваної напруги. Читання результатів організують або в циклі основної програми, або через заданий (максимальний для обраного АЦП) час після запуску перетворення, або використовують сигнал готовності для генерації переривання мікропроцесора. Останній варіант дозволяє досягти мінімального часу одержання результатів перетворення.

Драйвер АЦП із шиною P8

Структурна схема фрагмента мікроконтролерної системи наведена на рис.3.7, де MCU - мікропроцесор, ADC - аналогово-цифровий перетворювач. У даній схемі використовуються наступні лінії шини:

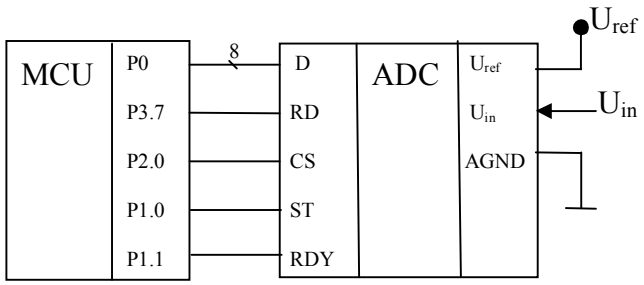


Рис.3.7

- 8-розрядна шина даних (D0 - D7);
- лінія передавання строба читання даних (RD);
- лінія вибірки ADC (CS);
- лінія початку перетворення (ST);
- лінія готовності даних (RDY).

Для читання інформації із шини P8 у мікропроцесорах сімейства MCS-51 можуть використовуватися команди MOVX A, @Ri і MOVX A, @DPTR, де Ri - регістри RAM - R0, R1. Часові діаграми такої шини наведені на рис. 3.8. Цикл перетворення ініціюється сигналом ST. Після завершення перетворення АЦП формує сигнал готовності даних RDY. Читання даних з АЦП відбувається з використанням ліній шини P8.

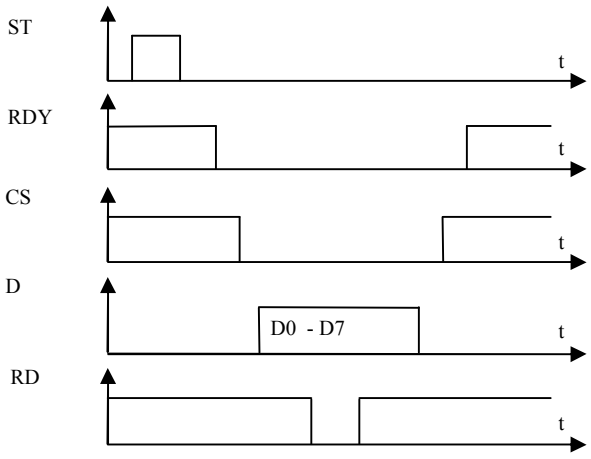


Рис. 3.8

Приклад підпрограми драйвера ADC із шиною P8

; Опис констант і змінних (заносится в описову частину програми)

ADC_ADRES equ 11111110b ; адреса вибірки DAC

```
ADC_DATA data 4H          ; регістр передавання даних з підпрограми
ST bit P1.0               ; лінія сигналу запуску
RDY bit P1.1              ; лінія готовності даних
```

; Підпрограма

```
; В основній програмі читання даних відбувається з регістра ADC_DATA
ADC_DRV_P8: MOV DPH, #ADC_ADRES ; установлення адреси ADC
            SETB ST             ; запуск перетворення
            CLR ST
ADC_LOOP:   JB RDY, ADC_LOOP    ; очікування завершення перетворення
            MOVX A, @DPTR       ; читання даних з ADC по шині P8
            MOV DAC_DATA, A     ; передавання даних у буферний регістр
            RET                 ; повернення із підпрограми
```

Драйвер АЦП із шиною I^2C

Структурна схема мікроконтролерної системи наведена на рис. 3.9.

Після установки адреси АЦП (CS) формується сигнал запуску перетворення (ST). Після надходження сигналу готовності даних (RDY) виробляється читання. У даній системі інформація з АЦП надходить у мікропроцесор по лініях шини I^2C.

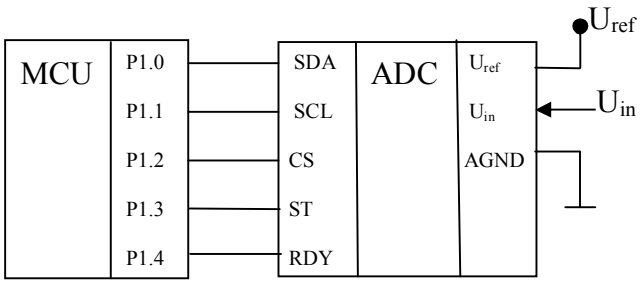


Рис.3.9

Приклад підпрограми-драйвера ADC з шиною I^2C

; Опис констант і змінних (заноситься в описову частину програми)

```
SDA         bit     P1.0
```

```
SCL         bit     P1.1
CS          bit     P1.2
ST          bit     P1.3
RDY         bit     P1.4
ADC_DATA    data    4H     ; регістр передавання даних з підпрограми
```

; Підпрограма

; В основній програмі читання даних ввдбувається з буферного регістру ADC_DATA

```
ADC_DRV_I2C: CLR CS ; установлення адреси ADC
             SETB ST; запуск перетворення
             CLR ST
ADC_LOOP:    JB RDY, ADC_LOOP; очікування завершення перетворення
             CALL I2C_READ; приймання даних по шині I2C, передавання
                          ; їх у регістр ADC_DATA
             SETB CS; скидання адреси ADC
             RET; повернення із підпрограми
```

Цифрова фільтрація введеного аналогового сигналу

Наявність у вимірюваній напрузі пульсацій та високочастотних завад призводять до того, що результати перетворення АЦП можуть змінюватися. Боротьбу з цим явищем проводять з використанням як схемотехнічних, так і програмних засобів. Високочастотні завади фільтруються за допомогою схемотехнічних засобів – завадоперешкоджаючих фільтрів. Для боротьби з низькочастотними пульсаціями використовують як схемотехнічні засоби (низькочастотні фільтри), так і програмні.

Серед програмних методів низькочастотної фільтрації найбільш часто вживають просте усереднювання та ковзаюче усереднювання.

Просте усереднювання передбачає проведення на періоді визначення значення напруги серії вимірювань. По результатам вимірювань визначається се-

реднє арифметичне. Недоліком методу простого усереднювання є те, що результат можливо визначити лише після проведення повної серії вимірювань. Це приводить до суттєвої затримки у часі.

Ковзаюче усереднювання (MAV – Moving Average) широко вживається для фільтрації як у техніці, так і у фінансах, економіці. Найбільш часто вживають наступні типи MAV фільтрів:

- Просте ковзаюче середнє;
- Зважене ковзаюче середнє;
- Експоненціальне ковзаюче середнє;

Термін "ковзаюче середнє" визначає, що масив значень, які усереднюються безперервно рухається у часі.

Ковзаюче усереднювання передбачає проведення двох етапів вимірювання. На попередньому етапі на періоді визначення значення напруги проводиться серія вимірювань. Масив значень запам’ятовується. На другому етапі, при визначенні наступних значень напруги, проводиться не серія, а лише одне вимірювання. Результат цього вимірювання розміщується в масиві замість найстарішого результату. Значення напруги визначається як середнє арифметичне записаних у масив даних

Такий підхід використовується у MAV фільтрі з простим ковзаючим середнім. У випадках, коли необхідно підсилити вагу результатів останніх вимірювань визначення проводять з ваговими коефіцієнтами складових масиву. Величини вагових коефіцієнтів обирають за різними методами, наприклад змінюючи їх за експоненціальним законом. У цих випадках отримують MAV фільтри із зваженим, або експоненціальним ковзаючим середнім.

Використання методів фільтрації із ваговими коефіцієнтами дає змогу поліпшити динамічні характеристики MAV фільтрів, зробити їх більш чутливими на останні зміни у вимірюваній напрузі.

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 4

КЛАВІАТУРА МІКРОКОНТРОЛЛЕРА

1. Мета практикуму

- Ознайомитися з схемотехнічними прийомами побудови клавіатур;
- Освоїти принципи побудови драйверів клавіатури.

2. Програма практикуму

2.1. вивчити схемотехнічні прийоми побудови клавіатур;

2.2. вивчити алгоритми побудови оброблювачів натискання кнопки;

2.3. розробити схему клавіатури з заданною кількістю кнопок;

2.4. розробити драйвер клавіатури, що забезпечує введення інформації в мікропроцесор і задані реакції на натискання кнопок;

2.5. В програмному середовищі μ-VISION/51 відладити програму драйверу клавіатури.

3. Завдання до практикуму

В табл. 4.1 наведені вихідні дані і прийняті наступні скорочення:

Таблиця 4.1

№	Число кнопок	Алгоритм прискорення брязкоту	Фіксація стану натискання	Цикл опитування [сек]	Час брязкоту [мсек]	Реакції на натискання кнопок
1	10	А1	ФН	1	1	Погоджується з викладачем
2	3	А2	ФН	0,5	2	
3	4	А3	ФНО	0,7	3	
4	5	А2	ФН	2	2	
5	2	А3	ФНО	0,3	1,3	
6	6	А1	ФН	0,8	1,8	
7	7	А1	ФН	1	3,3	
8	5	А2	ФНО	0,4	2,1	
9	4	А1	ФНО	0,2	2,7	
10	3	А3	ФН	0,9	3	
11	8	А2	ФН	1,4	2	
12	6	А1	ФНО	1,2	1	

А1 - алгоритм з двума опитуваннями і затримкою на час брязкоту контактів; А2 – з багатократними опитуваннями і мажоритарним принципом розпізнавання; А3 - з заданим числом співпадаючих значень, при багатократних опитуваннях; ФН – фіксація стану при натисканні кнопки; ФНО - фіксація стану при натисканні і відпускання кнопки.

4. Зміст звіту

- Титульний лист з назвою виконаного практикуму і склад бригади;
- Схема клавіатури с заданим числом кнопок;
- Текст програми з коментарями.

5. Контрольні питання

5.1. Приведіть класифікацію типів клавіатур.

5.2. Опишіть схемотехнічні прийоми використання окремих кнопок.

5.3. Визначіть схемотехнічні прийоми побудови клавіатур.

5.4. Опишіть алгоритми ідентифікації натисненого стану кнопок.

5.5. Які алгоритми фіксації натискання кнопок використовуються в драйверах клавіатур.

5.6. Які вихідні дані враховуються при побудові драйвера клавіатури і підпрограм реакцій на натискання кнопок.

6. Теоретичні відомості

Класифікація клавіатур

При побудові клавіатур, залежно від їх функціонального призначення, використовують різне число кнопок. Розрізняють клавіатури:

- для введення та редагування тексту (персональні комп'ютери);
- для введення переважно цифрової інформації, з можливістю введення буквеної інформації (телефони);
- для введення тільки цифрової інформації (калькулятори);

• для епізодичного настроювання параметрів мікроконтролерних пристроїв (побутова техніка, мікроконтролерні пристрої промислового призначення).

В клавіатурах з інтенсивним використанням число кнопок підвищене. Наприклад, розширена клавіатура персонального комп'ютера містить більш ніж 100 кнопок. Зазвичай, у міру зниження інтенсивності використання клавіатури, число кнопок зменшується. При цьому для збереження функціональних можливостей клавіатури кнопкам призначаються альтернативні функції, а також застосовуються алгоритми перепризначення таких функцій. Типова клавіатура мікроконтролера, що використовується епізодично для настроювання параметрів, містить 4 - 6 кнопок (СВЧ-печі, пральні машини, цифрові фотоапарати).

Схемотехнічні прийоми використання окремих кнопок

При формуванні принципової схеми клавіатури вирішуються дві задачі:

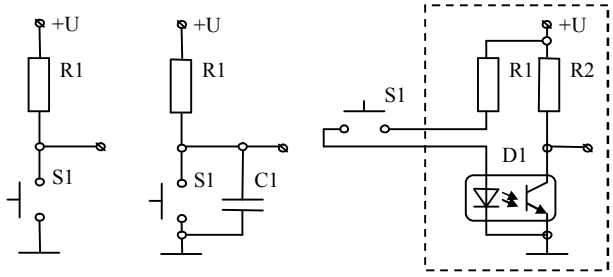
• забезпечення введення інформації від кнопки;

• узгодження клавіатури заданого обсягу з мікропроцесором.

Кнопка клавіатури містить комутуючу контактну пару. При встановленні режимів роботи контактної пари звичайно враховують величину мінімального струму. Для тактових кнопок, які використовуються при побудові клавіатур, цей струм складає 0,3 – 1 мА. Встановлення струму меншої величини веде до відмов функціонування кнопок.

Достатньо часто для формування напруги низького і високого рівней і забезпечення допустимого мінімального струму контактів використовується схема, наведена на рис.4.1,а.

На вибір схемотехнического рішення великий вплив також роблять відстань кнопки від мікроконтролера та рівень перешкод. При низькому рівні перешкод і відстані від кнопки до портів введення мікроконтролера порядку 0,1 - 0,2 м використається схема, наведена на рис.4.1,а. При підвищеному рівні високочастотних перешкод застосовують додатковий конденсатор (рис.4.1, б).



а) б) в)

Рис. 4.1

Дану схему використають на відстані до 2 - 5м від портів мікроконтролера. Зі збільшенням рівня перешкод підвищують струм контактної пари (до 3 - 10 мА) і величину ємності завадоподавляючого конденсатора. При необхідності збільшення довжини ліній зв'язку між кнопкою і мікроконтролером понад 5 м доцільно використати струмові петлі (10 - 100мА) і оптичні розв'язки. Така схема наведена на рис. 4.1, в. Канал оптичної розв'язки дозволяє усунути вплив спадання напруги на лінії зв'язку на рівні напруг, що сформовані, при включеному і виключеному станах кнопки.

Схемотехнічні прийоми побудови клавіатур

Розробку електричної схеми клавіатури провадять по заданому числу кнопок.

При малій кількості кнопок (1 - 10 шт) звичайно використовують безпосереднє підключення до ліній портів. Такий спосіб побудови клавіатури наведений на рис. 4.2.

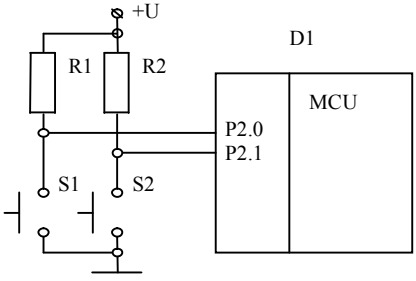


Рис. 4.2

При використанні мікроконтролера AT89C51, що має чотири 8-бітні порти введення-виведення інформації, максимальне число кнопок у клавіатурі досягає 32. У реальних мікроконтролерних системах це число зменшується, за рахунок необхідності реалізації інших функцій, наприклад індикації.

Якщо число кнопок клавіатури перебуває в діапазоні від 10 до 64 шт, то для введення інформації доцільно використати додаткові регістри. Електрична схема такої клавіатури наведена на рис. 4.3. У даній схемі для читання інформації з регістрів використана паралельна шина, сформована на базі порту P1 мікроконтролера. Для стробування регістрів використовують лінії порту P2. Дане рішення досить просто дозволяє за допомогою двох портів і 8 додаткових регістрів опитати клавіатуру, що містить 64 кнопки.

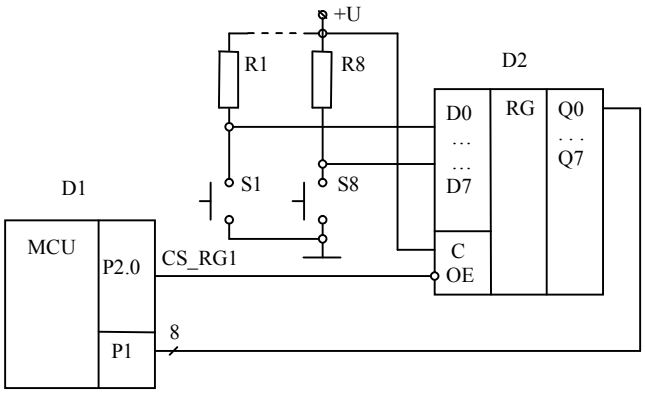


Рис. 4.3

При розробці клавіатур із числом кнопок, що перевищує 30 шт, звичайно використовують схеми сканованого опитування. На рис. 4.4 наведена електрична схема, що ілюструє можливість побудови такої клавіатури.

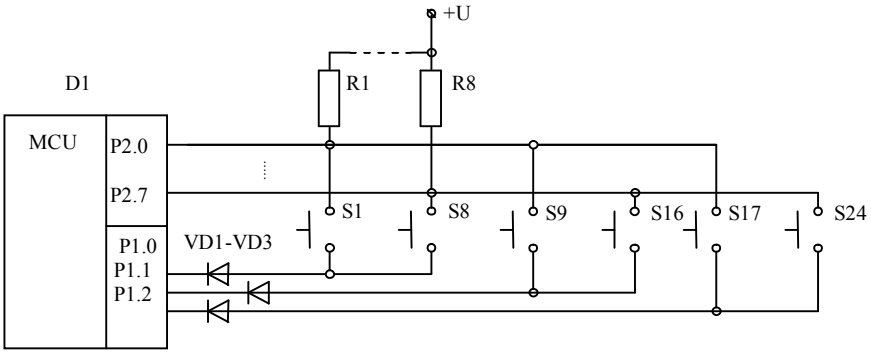


Рис. 4.4

Схема містить три групи по 8 кнопок S1 - S8, S9 - S16, S17 - S24, підключені до загальної паралельної шини введення даних (порт P2). Групи кнопок через відсікаючі діоди VD1-VD3, підключені до ліній сканування P1.0 - P1.2. Відсікаючі діоди виключають можливість короткого замикання по лініях порту P1 при одночасному натисканні кнопок різних груп.

На лініях сканування по черзі встановлюється “0”- стан і проводиться опитування стану кнопок активізованої групи. При використанні двох портів, максимальне число кнопок такої клавіатури досягає 64. Дана схема є більш простою, у порівнянні з попереднім варіантом.

Алгоритми ідентифікації натискання

Комутація контактної пари характеризується брязкотом контактів - багаторазовими перемиканнями із частотою механічного резонансу. В основному брязкот контактів характерний для замикання контактів. Час брязкоту залежить від типу контактів і може становити від 1 до 20 мс. У мікроконтролерних системах часто використовуються тактові кнопки з часом брязкоту 1 - 3мс. Для виключення багаторазового урахування перемикань контактів на інтервалі брязкоту, алгоритми ідентифікації натискання кнопки необхідно формувати з урахуванням даного явища.

Часто використаються три типи алгоритмів ідентифікації:

- з двома опитуваннями та затримкою на час брязкота контактів;
- з багаторазовими опитуваннями та мажоритарним принципом розпізнавання стану;
- із заданим числом співпадаючих значень, при багаторазових опитуваннях стану.

При використанні першого алгоритму виконується періодичне опитування кнопки клавіатури. При реєстрації натиснутого стану кнопки, для виключення впливу брязкоту контактів, організовується затримка на час брязкоту, потім повторне опитування. Результат повторного опитування відповідає дійсному

стану кнопки. Найбільше часто в драйверах клавіатури використається саме цей алгоритм.

У другому алгоритмі виконуються множинні опитування стану кнопки. Визначаються числа результатів експериментів, що відповідають різним станам. На підставі зіставлення цих чисел, за мажоритарним принципом, приймається рішення про поточний стан кнопки. Для виключення впливу брязкоту контактів, число опитувань вибирається таким чином, щоб час, витрачений на множинні опитування, перевищував час брязкоту.

При реалізації третього алгоритму виконуються задане число опитувань. Якщо отримані різні дані про стан кнопки, то результат опитувань анулюється і серія повторюється. Як й у попередньому алгоритмі число опитувань вибирається відповідно до часу брязкоту контактної пари.

Драйвер клавіатури

При підготовці драйвера клавіатури задаються:

- число кнопок;
- час брязкоту контактів і тип алгоритму його усунення;
- час циклу опитування стану кнопок;
- спосіб реєстрації натискання.

Час циклу опитування стану кнопок визначає швидкодія клавіатури. Введення періоду циклу опитування необхідне для узгодження швидкості роботи контролернії системи та часу реакції оператора. Типовий час реакції становить 0,3 - 1сек. Для забезпечення заданої циклічності опитування клавіатури, залежно від особливостей функціональних властивостей мікроконтролерів, використовують два прийоми. Якщо в мікроконтролерній системі часові затримки в 0,3 - 1сек при виконанні головного циклу програми є несуттєвими, то зазначену затримку реалізують у головному циклі. У протилежному випадку циклічне опитування клавіатури забезпечують за рахунок використання переривань від таймера лічильника. Застосування другого способу зменшує час виконання головного циклу.

При реєстрації натискання кнопок використають два алгоритми:

- подію реєструють у натиснутому стані кнопки;
- подію реєструють, якщо кнопка була натиснута і потім відпущена.

При використанні драйверів, сформованих на основі першого алгоритму, тривале утримання кнопки в натиснутому стані призводить до циклічного виконання відповідної підпрограми реакції. Таку реакцію драйвера доцільно застосовувати при необхідності зміни значення змінної.

Якщо необхідно забезпечити більше чітку зміну параметрів, із твердою фіксацією їхніх значень по кожному натисканню кнопок - тоді доцільно використати другий алгоритм.

Приклад драйверу клавіатури

*Завдання***.** Запрограмувати роботу мікроконтролера з клавіатурою, яка має 3 кнопки, алгоритм усунення брязкоту з двума опитуваннями і затримкою на час брязкоту контактів в 3 мс, спосіб опитування за допомогою підпрограми опитування, час циклічного опитування стану кнопок реалізовані за допомогою затримки не меньше 0,5 с (узгодження з оператором), яка знаходиться в головному циклі. Спосіб регістрації стану кнопки при натисканні.

```
; Опис констант і змінних
    INI_P2 EQU 00000111b ; константа ініціалізації порту кнопок
    UP BIT P2.0 ; кнопка збільшення параметру
    DOWN BIT P2.1 ; кнопка зменшення параметру
    MODE BIT P2.2 ; кнопка зміни режиму роботи мікроконтролера
ORG 0H ; адреса вектора рестарту після пуску процесора
    SJMP INIT ; перехід на блок ініціалізації
ORG 30H ; адреса блоку ініціалізації
INIT:  MOV P2, #INI_P2 ; ініціалізація порту P2
```

; Основна програма

```
MAIN: CALL KEY ; підпрограма опитування клавіатури
      CALL DELAY_05s ; затримка для узгодження з оператором
      SJMP MAIN ; зациклення головної програми
```

; Підпрограма опитування клавіатури

```
KEY_1:    JNB UP, KEY_DEL ; перша перевірка натискання кнопок
          JNB DOWN, KEY_DEL; якщо якась кнопка натиснена, то
          JNB MODE, KEY_DEL ; перехід на організацію затримки, якщо
          SJMP KEY_END ; жодна не натиснена, то вихід із підпрограми
KEY_DEL: CALL DELAY_3ms ; затримка на час брязкоту контактів
KEY_2:    JNB UP, KEY_UP ; друга перевірка натискання кнопок і перехід
          JNB DOWN, KEY_DOWN; на відповідні підпрограми обробки
          JNB MODE, KEY_MODE
          SJMP KEY_END      ; перехід на вихід із підпрограми
KEY_UP:   CALL PP_KEY_UP   ; виклик підпрограми натискання кнопки “UP”
          SJMP KEY_END      ; перехід на вихід із підпрограми
KEY_DOWN: CALL PP_KEY_DOWN ; виклик підпрограми обробки
                           ; натискання кнопки “DOWN”
          SJMP KEY_END ; вихід із підпрограми опитування клавіатури
KEY_MODE: CALL PP_KEY_MODE; виклик підпрограми обробки
                           ;натискання “MODE”
KEY_END: RET; повернення із підпрограми
END                        ; директива асемблеру про закінчення програми
```

Формування реакцій на натискання кнопок

Зміст підпрограм реакцій визначається функціональними особливостями реалізованої клавіатури. Розглянемо приклад, у якому можлива селекція двох режимів роботи. У кожному з режимів може бути модифікація однієї із двох змінних із заданим кроком і діапазоном зміни.

У наведеному прикладі драйвера використано три кнопки “MODE”, “UP”, “DOWN”. Кнопку “MODE” використовується для зміни режиму роботи, а “UP”, “DOWN” служать для зменшення або збільшення значень змінних. В регістрі FLAG використані два прапорці, що відповідають роботі в першому (F_MODE1) або другому (F_MODE2) режимах. Активним є одиничний стан біта. Константа ініціалізації регістру (INI_FLAG) забезпечує настроювання на перший режим роботи. Для зберігання значень змінних використано регістри PEREM_1 і PEREM_2. Робочі регістри PEREM, MIN_PEREM, MAX_PEREM використовуються для обміну даними між підпрограмами. При призначенні комірки для змінної PEREM доцільно використати регістри з адресами 0H - 7H. Це пов'язане з необхідністю застосування команди порівняння CJNE.

; Опис констант і змінних (заносится в описову частину програми)

```
FLAG DATA 20H ; регістр прапорців визначається в бітовій області RAM
F_MODE1 BIT FLAG.0 ; прапорець першого режиму
F_MODE2 BIT FLAG.1 ; прапорець другого режиму
INI_FLAG EQU 00000001b ; константа ініціалізації регистру прапорців
PEREM DATA 2H ; визначення регістра зберігання змінної PEREM
MIN_PEREM DATA 10H; визначення регістра зберігання MIN_PEREM
MAX_PEREM DATA 11H ; визначення регістра зберігання MAX_PEREM
PEREM_1 DATA 12H ; визначення регістра зберігання змінної PEREM_1
PEREM_2 DATA 13H ; визначення регістра зберігання змінної PEREM_2
INI_PEREM_1 EQU 30 ;дистрибутивне значення змінної PEREM_1
MIN_PEREM_1 EQU 15 ;мінімальне значення змінної PEREM_1
MAX_PEREM_1 EQU 50 ;максимальне значення змінної PEREM_1
DELTA_1 EQU 5 ;крок зміни змінної PEREM_1
INI_PEREM_2 EQU 30 ;дистрибутивне значення змінної PEREM_2
MIN_PEREM_2 EQU 10 ;мінімальне значення змінної PEREM_2
MAX_PEREM_2 EQU 100 ;максимальне значення змінної PEREM_2
DELTA_2 EQU 10 ;крок зміни змінної PEREM_2
```

Блок ініціалізації

; В цьому блоці провадиться завантаження дистрибутивних значень змінних, а також активізація першого режиму роботи.

```
INIT: MOV PEREM_1, #INI_PEREM_1 ; завантаження дистрибутивних значень
      MOV PEREM_2, #INI_PEREM_2 ; змінних PEREM_1, PEREM_2 і
      MOV FLAG, #INI_FLAG       ; регистра прапорців.
      MOV PEREM, PEREM_1        ;завантаження  значення змінної PEREM_1
      MOV MIN_PEREM, #MIN_PEREM_1 ;значення мінімальної границі
      MOV MAX_PEREM, #MAX_PEREM_1 ; значення максимальної границі,
      MOV DELTA, #DELTA_1 ; значення кроку зміни
```

; Підпрограма реакції на звернення до кнопки MODE

; В підпрограмі реалізуються розгалуження у відповідності з поточним режимом роботи. Потім відбувається переключення прапорців на наступний режим роботи, передавання змінної із робочого регістру (PEREM) в регістр зберігання (PEREM_1/2) і завантаження в робочі регістри (PEREM, MIN_PEREM, MAX_PEREM, DELTA,) параметрів нового режиму роботи.

```
PP_KEY_MODE: JB F_MODE1, MODE_1; розгалуження у відповідності з
             JB F_MODE2, MODE_2;  поточним режимом
             SYMP MODE_END; вихід при невизначеному стані прапорців
MODE_1:      CLR F_MODE1 ; зміна режимів з MODE_1 на MODE_2
             SETB F_MODE2
             MOV PEREM_1, PEREM ;запис значення PEREM_1
             MOV PEREM, PEREM_2 ; завантаження в робочі регістри
             MOV MIN_PEREM, #MIN_PEREM_2 ; параметрів PEREM_2
             MOV MAX_PEREM, #MAX_PEREM_2
             MOV DELTA, # DELTA_2
             SJMP MODE_END ; перехід на вихід із підпрограми
MODE_2:      CLR F_MODE2 ; зміна режиму з MODE_2 на MODE_1
             SETB F_MODE1
```

```
                MOV PEREM_2, PEREM; запис значення змінної PEREM_2
                MOV PEREM, PEREM_1 ; завантаження змінної PEREM_1
                MOV MIN_PEREM, #MIN_PEREM_1 ; PEREM_1
                MOV MAX_PEREM, #MAX_PEREM_1
                MOV DELTA, #DELTA_1
MODE_END:       RET; повернення із підпрограми
```

Підпрограми реакції на натискання кнопок “UP” і “DOWN” використовують загальний алгоритм. Для виключення збійних ситуацій при невизначеному стані прапорців проводиться їх аналіз з можливістю аварійного виходу із підпрограми. Потім проводиться перевірка досягнення межі діапазону зміни змінної. Якщо змінна не відповідає граничному значенню, відбувається модифікація її значення.

; Підпрограма реакції на звернення до кнопки UP

```
PP_KEY_UP: JNB F_MODE1, UP_END ; вихід із підпрограми при
           JNB F_MODE2, UP_END ; невизначеному стані прапорців
           CJNE R2, MAX_PEREM, UP_ADD; порівняння з максимальним
           SYMP UP_END; перехід на вихід із підпрограми
UP_ADD:    ADD R2, DELTA ; збільшення значення змінної
UP_END:    RET; повернення із підпрограми
```

; Підпрограма реакції на звернення до кнопки DOWN

```
PP_KEY_DOWN: JNB F_MODE1, DOWN_END ; вихід із підпрограми при
             JNB F_MODE2, DOWN_END ; невизначеному стані прапорців
             CJNE R2, MIN_PEREM, DOWN_SUBB; порівняння з мінімальним
              SYMP DOWN_END; перехід на вихід із підпрограми
DOWN_SUBB: CLC C; зменшення значення змінної
             SUBB R2, DELTA
DOWN_END:  RET; повернення із підпрограми.
```

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 5
ВИВЕДЕННЯ ДАНИХ НА ІНДИКАТОРИ

1. Мета практикуму

- Ознайомитися з методами виведення інформації на індикатор;
- Освоїти метод статичної індикації;
- Освоїти метод динамічної індикації.

2. Програма практикуму

2.1. Вивчити різновиди індикаторів, а також особливості побудови електричних схем і алгоритмів програм виведення інформації.

2.2. Для методу статичної індикації розробити структурну схему дисплея і програму, що забезпечує виведення заданих символів на багаторозрядний семисегментний індикатор.

2.3. Для методу динамічної індикації розробити структурну схему дісплея і програму, що забезпечує виведення символів:

2.3.1. на багаторозрядний семисегментний індикатор;

2.3.2. на мозаїчний індикатор;

2.4.У програмному середовищі μ-VISION/51 налагодити програми статичної індикації.

3. Завдання до практикуму

У табл. 5.1 наведені вихідні дані для п.п. 2.2, 2.3.1. Виведені числа вибрати самостійно, у відповідності із заданою розрядністю числа. У таблиці заданий спосіб підключення сегментів індикаторів "*a* – *h*" до ліній порту і використовуються такі позначення: *N* – розрядність одного індикатора; *K* – число індикаторів розрядністю *N*.

4. Зміст звіту

- Титульний лист з назвою виконаного практикуму і склад бригади;
- Текст програми з коментарями.

Таблиця 5.1

№	*K*	*N*	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*
1	4	1	P1.0	P3.1	P1.2	P0.3	P1.4	P1.5	P1.6	P1.7
2	1	3	P1.4	P1.5	P1.6	P2.7	P1.0	P1.1	P1.2	P1.3
3	2	2	P2.0	P2.1	P2.2	P2.3	P1.4	P1.5	P1.6	P1.7
4	2	2	P3.4	P1.5	P3.6	P1.7	P2.0	P2.1	P3.2	P3.3
5	4	2	P1.0	P1.1	P3.2	P1.7	P2.3	P1.4	P2.2	P1.5
6	2	3	P1.2	P1.4	P0.0	P0.1	P2.2	P2.7	P1.5	P1.7
7	1	3	P3.2	P1.5	P0.1	P1.1	P2.1	P2.0	P1.2	P0.7
8	2	3	P1.3	P2.0	P1.1	P2.3	P2.2	P2.5	P1.7	P0.7
9	1	2	P2.2	P2.4	P2.0	P2.1	P3.2	P3.7	P1.5	P1.1
10	2	3	P3.0	P1.3	P3.1	P1.2	P2.5	P2.4	P2.0	P2.7
11	3	2	P1.1	P2.1	P3.1	P0.1	P2.3	P0.4	P2.5	P1.6
12	5	1	P3.0	P3.1	P3.2	P1.1	P2.2	P1.3	P2.4	P1.5

5. Контрольні питання

- Як класифікуються індикатори?
- Як реалізується метод статичної індикації?
- Як реалізується метод динамічної індикації?

6. Теоретичні відомості

Класифікація індикаторів. По типу фізичних процесів розрізняють індикатори: лампи накалювання; вакуумні; люминисцентні; світлодіодні (LED); рідкокристалічні (LCD); плазмові панелі; активні і пасивні матриці.

У мікроконтролерних системах керування та обробки інформації в даний час найбільше поширення одержали LED і LCD індикатори.

Індикатори LED - типу. Даний тип індикаторів використовується при побудові дисплеїв, працюючих у широкому діапазоні температур (- 60 … + 100°С).

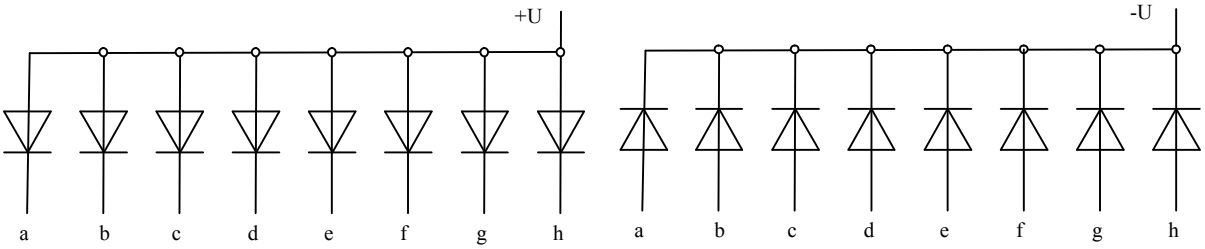
Індикатори характеризуються високою надійністю, яскравістю світіння (в окремих екземплярів до 35-50 мкд), низькою вартістю. Фірмами Kingbright і Paralight випускаються індикатори з семи основними кольорами світіння.

У LED-індикаторах містяться, як правило, тільки елементи індикації, а інтерфейсні вузли відсутні.

Випускають індикатори наступних конструкцій: одиночні індикатори циліндричної форми, з діаметрами 1, 3, 5, 7, 10 мм; мнемонічні індикатори (прямокутної, трикутної, символьної форми); світлові табло прямокутної і круглої форми; графічні лінійки; одиночні семисегментні індикатори і їхні набори (2 - 10 десяткових розрядів); мозаїчні індикатори.

Одиночні індикатори, як правило, містять один світлодіод. Індикатори з великою площею світної поверхні (табло, семисегментные індикатори великого розміру) можуть містити декілька послідовно включених світлодіодів.

У дисплеях для відображення цифр застосовуються семисегментні індикатори. По виду електричної схеми розрізняють два типи таких індикаторів – із загальним анодом (рис.5.1,а) і з загальним катодом (рис. 5.1,б).



а) б)

Рис.5.1

Вибір конкретного типу індикатора визначається особливостями принципової схеми пристрою і допустимими рівнями вихідних струмів (високого і низького рівня) буферних підсилювачів.

Сегменти індикатора позначаються літерами "*a* – *h*". Розташування сегментів в індикаторі звичайно відповідає варіанту, наведеному на рис. 5.2. Для побудови багаторозрядних дисплеїв можуть використовуватися одиночні індикатори, або набори індикаторів. За електричною схемою розрізняють два типи наборів, призначених для статичної і динамічної індикації.

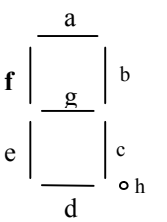


Рис. 5.2

Набори семисегментних індикаторів статичної індикації містять повний набір ліній керування для кожного розряду виведеного числа (рис. 5.1). У набо-

рах, призначених для динамічної індикації (рис.5.3), лінії керування “*a* - *h*” усіх розрядів поєднуються. Розділяються лише загальні виводи розрядних груп (+ *U* або – *U*).

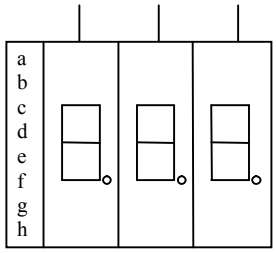


Рис.5.3

Мозаїчні індикатори використовуються при необхідності відображення символів алфавіту. Такі індикатори містять матрицю світлодіодів. Розрізняють два типи індикаторів. Перший тип призначений для побудови великих панелей. Розмір таких індикаторів 4 х 4, 8 х 8, 10 х 10. Розмір другого типу індикаторів 5 х 7 відповідає піксельному набору, використовуємого для відображення шрифтів CGA формату. Електрична схема мозаїчного індикатора розміром 2 х 2 представлена на рис. 5.4. Індикатори такого типу можуть застосовуватися тільки в дисплеях з динамічною індикацією.

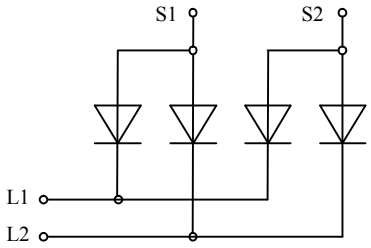


Рис. 5.4

Метод статичної індикації. У дисплеях, що використовує статичний метод індикації, через світлодіоди індикатора протікає безупинний робочий струм. Для забезпечення такого режиму роботи кожен одиночний семисегментний індикатор повинний підключатися через індивідуальний буферний підсилювач. Як такі підсилювачі часто використовують рівнобіжні регістри. Передавання інформації на регістри здійснюється за допомогою шини мікроконтролера P8, або формується програмний рівнобіжний інтерфейс.

На рис.5.5 наведена принципова схема двохрозрядного індикатора, у якому використана програмна паралельна шина. У схемі використано два буферних регістри D2, D3, підключених до загальної паралельної шини, сформованої за допомогою порту P1. Для активізації виходів регістрів на лінії OE (OUT ENABLE – включення виходу) подані активні рівні логічних сигналів. Вибірка відповідного регістра здійснюється за допомогою ліній P2.0, P2.1. Резистори *R*1 – *R*16 використовуються для обмеження струму, що протікає через світлодіоди індикаторів *H*1, *H*2.

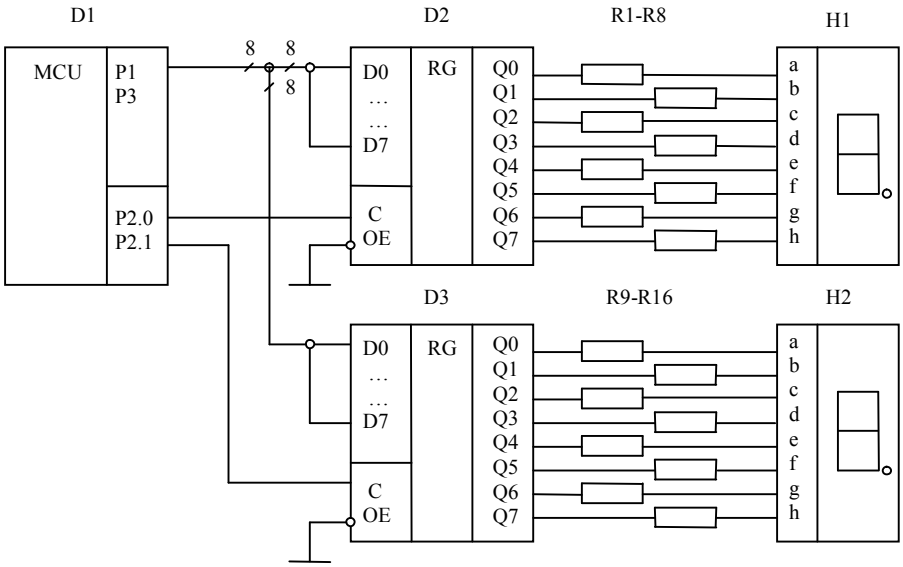


Рис. 5.5

Досить часто при розробці друкованої плати контролера виникає необхідність перенесення ряду ліній паралельної шини з одного порту на інший, або зміни їхнього порядку проходження в межах одного порту. У цьому випадку застосовують перекодування ліній паралельної шини. Таке перекодування може застосовуватися як для шини в цілому, так і для кожного розряду індикатора. Нижче розглянутий приклад формування драйвера пристрою індикації, приведеного на рис.5.5.

Драйвер дворозрядного індикатора з програмно формованою рівнобіжною шиною і статичною індикацією. До паралельної шини регістри підключені однаково, а спосіб комутації ліній порту P1 представлено в табл. 5.3.

Таблиця 5.3

Сегменти індикатора	*h*	*g*	*f*	*e*	*d*	*c*	*b*	*a*
Вихідне пакування	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
Код “1”	1	1	1	1	1	0	0	1
Код “5”	1	0	0	1	0	0	1	0
Розряди портів	P1.0	P1.5	P3.4	P1.3	P3.7	P1.2	P1.6	P1.1

Регістр *D*2 з'єднаний з індикатором десятків *H*1, а *D*3 з індикатором одиниць – *H*2. Для перекодування ліній паралельної шини з упакування кодування символів використовується допоміжна підпрограма.

У схемі використані регістри зі статичним керуванням. Запис інформації в регістр здійснюється при подачі на вхід синхронізації (C) високого рівня напруги.

Для передавання даних у підпрограму перекодування паралельної шини необхідно використовувати біт доступний регістр. Коди виведених символів визначалися для семисегментного індикатора з загальним катодом (L - активні рівні керування), для схеми розташування сегментів, приведеної на рис.5.2. У таблиці 5.3 представлені коди двох виведених цифр (“1” і “5”).

; Опис констант і змінних (заноситься в описову частину програми)

```
INI_P1      EQU  0      ; константа ініціалізації порту P1
INI_P3      EQU  0      ; константа ініціалізації порту P3
INI_P2      EQU  0      ; константа ініціалізації порту керування
CS_TEN      BIT  P2.0   ; лінія вибірки регістра десятків
CS_ONE      BIT  P2.1   ; лінія вибірки регістра одиниць
RG_CODE DATA 20H ; регістр передавання даних у підпрограму перекодування
CODE_TEN EQU 11111001bB ; код виведеного розряду десятків “1”
CODE_ONE EQU 10010010b  ; код виведеного розряду одиниць “5”
    LINE0           BIT  P1.1       ; лінія шини LINE0
    LINE1           BIT  P1.6       ; лінія шини LINE1
    LINE2           BIT  P1.2       ; лінія шини LINE2
    LINE3           BIT  P3.7       ; лінія шини LINE3
    LINE4           BIT  P1.3       ; лінія шини LINE4
    LINE5           BIT  P3.4       ; лінія шини LINE5
    LINE6           BIT  P1.5       ; лінія шини LINE6
    LINE7           BIT  P1.0       ; лінія шини LINE7
```

; Фрагмент блоку ініціалізації

```
        MOV P1, #INI_P1 ; ініціалізація першого порта
        MOV P2, #INI_P2 ; ініціалізація другого порта
        MOV P3, #INI_P3 ; ініціалізація третього порта
```

; Підпрограма перекодування паралельної шини

; Вихідні дані заносяться перед викликом підпрограми в буферний регістр RG_CODE. Результат перекодування на відповідних лініях паралельної шини.

```
CODE_0: MOV C, RG_CODE.0 ; Передавання “0” біта буферного регістра
            MOV LINE0, C          ; в лінію шини LINE0
CODE_1: MOV C, RG_CODE.1 ; Передавання “1” біта буферного регістра
            MOV LINE1, C          ; в лінію шини LINE1
CODE_2: MOV C, RG_CODE.2 ; Передавання “2” біти буферного регістра
            MOV LINE2, C         ; в лінію шини LINE2
CODE_3: MOV C, RG_CODE.3 ; Передавання “3” біти буферного регістра
            MOV LINE3, C          ; в лінію шини LINE3
CODE_4: MOV C, RG_CODE.4 ; Передавання “4” біти буферного регістра
           MOV LINE4, C           ; в лінію шини LINE4
CODE_5: MOV C, RG_CODE.5 ; Передавання “5” біта буферного регістра
           MOV LINE5, C            ; в лінію шини LINE5
CODE_6: MOV C, RG_CODE.6 ; Передавання “6” біта буферного регістра
           MOV LINE6, C          ; в лінію шини LINE6
CODE_7: MOVC, RG_CODE.7   ; Передавання “7” біта буферного регістра
           MOV LINE7, C          ; в лінію шини LINE7
RET; повернення із підпрограми
```

; Драйвер двохрозрядного індикатора з програмною рівнобіжною шиною

```
DIS_TEN: MOV RG_CODE, # CODE_TEN ; завантаження коду десятків
             CALL CODE; перекодування – виведення на шину
             SETB CS_TEN ; строб запису шини данних в регістр десятків
             CLR CS_TEN
DIS_ONE: MOV RG_CODE, #CODE_ONE; завантаження коду одиниць
```

```
CALL CODE ; перекодування – виведення на шину
SETB CS_ONE ; строб запису шини данних в регістр одиниць
CLR CS_ONE
 RET ; повернення із підпрограми
```

Метод динамічної індикації. У дисплеях, що використовують динамічний метод індикації, через світлодіоди індикатора протікає імпульсний робочий струм. Використовується загальний буферний підсилювач для живлення сегментів індикатора "*a* - *h*". Об'єднані виводи розрядів індикатора по черзі підключаються до шини живлення. Для виключення оптичних ефектів частота комутації розрядів повинна перевищувати 25 Гц. Світлодіоди індикаторів допускають підвищення частоти до 2 - 4 кГц. Для забезпечення нормальної яскравості світіння сегментів необхідно збільшувати імпульсний струм світлодіодов так щоб середнє за період індикації значення струму відповідало номінальному статичному струму. Так, у трьохрозрядному індикаторі необхідно збільшувати імпульсний струм у три рази, у порівнянні з індикатором статичного типу. Динамічний метод індикації схемотехнично реалізується простіше, у порівнянні зі статичним, однак програмний продукт – більш складний.

На рис.5.6 наведена принципова схема двохрозрядного індикатора, у якому використаний метод динамічної індикації.

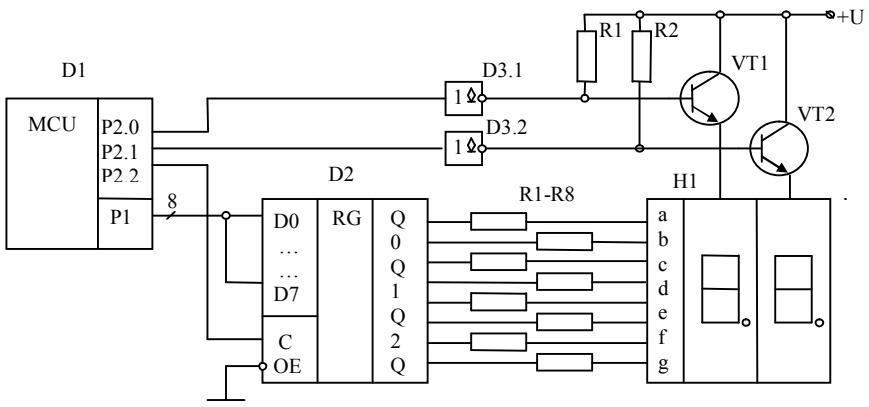


Рис. 5.6

У схемі використаний один буферний регістра D2, підключений до паралельної шини мі-

кроконтролера (порт P1). Для активізації виходів регістра на лінію OE поданий нульовий логічний рівень напруги. Вибірка відповідного розряду індикатора здійснюється за допомогою ліній P2.0, P2.1. Резистори *R*1 – *R*8 використовуються для обмеження струму, що протікає через світлодіоди індикатора *H*1.

Для стабілізації напруги, прикладеної до світлодіодів індикатора, використані підсилювачі напруги *D*3.1, *D*3.2. Транзистори *VT*1, *VT*2 застосовуються для комутації розрядів індикатора.

Драйвер двохрозрядного індикатора з програмно формованою рівнобіжною шиною і динамічною індикацією. Паралельна шина процесора формується тільки портом P1. Упакування ліній порту наведено в табл. 5.4. Для забезпечення стабільної частоти вибірки розрядів індикатора використовується таймер-лічильник і система переривань. Частота генератора мікропроцесора складає 12 Мгц. Період комутації розряду індикатора складає 100 мкс (1 Кгц).

Таблиця 5.4

Сегменти індикатора	*h*	*g*	*f*	*e*	*d*	*c*	*b*	*a*
Упакування ліній шини	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

```
; Опис констант і змінних
INI_P1 EQU 0       ; константа ініціалізації порту шини даних
INI_P2 EQU 0       ; константа ініціалізації порту керування
CS_TEN BIT P2.0 ; лінія вибірки розряду десятків
CS_ONE BIT P2.1 ; лінія вибірки розряду одиниць
CS_RG BIT P2.2
CODE_TEN EQU 11111001b ; код виведеного розряду десятків  “1”
CODE_ONE EQU 10010010b ; код виведеного розряду одиниць “5”
INI_TMOD EQU 00000010b   ; 2 режим роботи T/C0
INI_TC0 EQU (0FF - 100) ; константа завантаження нульового таймера
; Програма;
```

```
ORG 0H                    ; адреса вектора рестарту після пуску процесора
      SJMP INIT           ; перехід на блок ініціалізації мікропроцесора
ORG 0BH                   ; адреса вектора переривань таймера/лічильника TC0
      SJMP INTER_TC0 ; перехід на підпрограму переривань від TC0
ORG 20H                   ; початкова адреса блоку ініціалізації
INIT: MOV P1, #INI_P1  ; ініціалізація порту P1
      MOV P2, #INI_P2  ; ініціалізація порту P2
      MOV TMOD, #INI_TMOD ;ініціалізація таймерів/лічильників T/C0, T/C1
      MOV TH0, #INI_TC0 ;завантаження рахункового регістра таймера T/C0
      MOV TL0, #INI_TC0 ;завантаження регістра перевантаження T/C0
      SETB ET0            ; дозвіл переривань від T/C0
      SETB EA             ; загальний дозвіл переривань
      SETB TR0            ; включення таймера-лічильника T/C0
      ;Головна програма
MAIN:     SJMP MAIN       ; зациклення головної програми
;Підпрограма переривань від таймера, що генерує інтервал дискретизації
INT_TC0: JB CS_ONE, INT_TEN ;аналіз лінії вибірки розряду десятків
         JB CS_TEN, INT_ONE  ;аналіз лінії вибірки розряду одиниць
         SJMP INT_END ; перехід на кінець підпрограми
INT_ONE: CLR CS_TEN ;зміна прапорців інтервалів
         MOV P1, # CODE_ONE ;передавання в порт коду індукуємого розряду
         MOV P1, RG_ONE ;передавання коду розряду при BCD-перетворені
         SETB CS_RG; фіксація коду в буферному регістрі
         CLR CS_RG
         SETB CS_ONE ; активізація індукуємого розряду
         SJMP INT_END ; вихід із процедури
INT_TEN:  CLR CS_ONE
         MOV P1, # CODE_TEN ;передавання в порт коду індукуємого розряду
         MOV P1, RG_TEN; передавання коду розряду при BCD-перетворені
           SETB CS_RG
```

```
        CLR CS_RG
        SETB CS_TEN
INT_END: RETI
END                     ; директива асемблера про закінчення програми
```

У наведеному прикладі на дисплей виводяться заздалегідь визначені цифри «1» і «5». Зазвичайно виникає необхідність передавання на дисплей числа, у якого значення розрядів заздалегідь невідомо. У таких випадках застосовують двійково-десяткове перетворення з наступним перекодуванням символів розрядів. Передавання параметрів у приведеній вище програмі може здійснюватися за допомогою регістрів RAM (RG_TEN, RG_ONE). У даному прикладі цей варіант використаний як коментар. Дані в підпрограму передаються через регістр RG_CODE, а з підпрограми - RG_TEN, RG_ONE.

Виділення десяткових розрядів здійснюється послідовним діленням вихідного числа на 100 (число сотень), потім залишку від цього ділення на 10 (число десятків) і виділенням останнього залишку (числа одиниць).

Для перекодування отриманих чисел у коди індикації використовується табличний метод. У пам'яті програм (наприкінці програмного продукту) розташовують таблиц з послідовним розташуванням кодів від 0 до 9. У наведеній нижче підпрограмі об'єднані BCD-перетворення і перекодування. Підпрограма забезпечує перетворення однобайтних чисел.

; Опис констант і змінних (заноситься в описову частину програми)

```
RG_CODE  DATA      5H; регістр передавання даних у підпрограму BCD
RG_HUND DATA       6H; регістр передавання даних з підпрограми BCD
RG_TEN    DATA     7H; регістр передавання даних з підпрограми BCD
RG_ONE    DATA     8H; регістр передавання даних з підпрограми BCD
K0    EQU          ; код індикації “0”
K1    EQU  11111001B ; код індикації “1”
K2    EQU          ; код індикації “2”
K3    EQU          ; код індикації “3”
K4    EQU          ; код індикації “4”
```

```
K5    EQU  10010010b  ; код індикації “5”
K6    EQU             ; код індикації “6”
K7    EQU             ; код індикації “7”
K8    EQU             ; код індикації “8”
K9    EQU             ; код індикації “9”
```

; Фрагмент блоку ініціалізації

```
MOV DPTR, # TAB_CODE
```

; Підпрограма BCD - перетворення

; Вихідні дані заносяться перед викликом підпрограми в буферний регістр RG_CODE. Результат перекодування - це виведені коди розрядів у регістрах RG_HUND, RG_TEN, RG_ONE.

```
BCD_HUND: MOV A, RG_CODE
          MOV B, #100
          DIV AB
          MOVC A, @A+DPTR
          MOV RG_HUND, A
BCD_TEN:  MOV A, B
          MOV B, #10
          DIV AB
          MOVC A, @A+DPTR
          MOV RG_TEN, A
BCD_ONE:  MOV A, B
          MOVC A, @A+DPTR
          MOV RG_ONE, A
           RET; повернення із підпрограми
```

;Таблиця кодів. Розміщується наприкінці програми, перед директивою “END”.

```
ORG 200H
          TAB_CODE: DB K0, K1, K2, K3, K4, K5, K6, K7, K8, K9
```

ЛІТЕРАТУРА

1. Григорьев В.Л. Программирование однокристальных микропроцессоров. – М.: Энергоатомиздат, 1987. – 288 с.

2. Сташин В.В. Урусов А.В. Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах: – М.: Энергоатомиздат, 1990. – 221 с.

3. Липовецкий Г.П. и др. Однокристальные микроЭВМ. Семейство МК48, Семейство МК51. Техническое описание и руководство по применению. - М.: МП "Бином", 1992. – 339 с.

4. Боборыкин А.В., Липовецкий Г.П., Литвинский Г.В., Оксинь О.Н., Прохорчик С.В., Проценко Л.В., Петренко Н.В., Сергеев А.А., Сивобород П.В. Однокристальные микроЭВМ. -М.: МИКАП, 1994. – 400с.

5. Микропроцессоры. В 3-х кн. Кн.1. Архитектура и проектирование микро-ЭВМ. Организация вычислительных процессов. Учебник для ВТУЗов. /П.В.Нестеров, В.Ф. Шаньгин, В.Л. Горбунов и др. Под ред. Л.Н. Преснухина. - М.: Высш. шк., 1986, – 495с.

6. Фрунзе А.В. Микроконтроллеры? Это же просто!; В 3-х т. - М.; ООО «ИД СКИМЕН», 2002.

7. Каспер Э. Программирование на языке Ассемблера для микроконтроллеров семейства i8051. – М.: Горячая линия – Телеком, 2004. – 191 с.

8. Схемотехника электронных систем. Микропроцессоры и микроконтроллеры: Учебник / В.И.Бойко, А.Н.Гуржий, В.Я. Жуйков и др. - СПб.: БХВ-Петербург, 2004. - 464 с.

9. Мікропроцесорна техніка. 2е видання, перероблене. / Ю.І.Якименко, Т.О.Терещенко, Є. І.Сокол, В.Я.Жуйков, Ю.С.Петергеря. – Київ 2004. – 440 с.