НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

з курсу „Мікропроцесорна техніка”

Мікроконтролери сімейства STMicroelectronics

Київ 2008

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ
з курсу „Мікропроцесорна техніка”
Мікроконтролери сімейства STMicroelectronics

ДЛЯ СТУДЕНТІВ СПЕЦІАЛЬНОСТІ
7.09.08.03 – "ЕЛЕКТРОННІ СИСТЕМИ"
УСІХ ФОРМ НАВЧАННЯ

Затверджено Радою факультету
електроніки, протокол № 02/08
від 25.02.2008 р.

Київ НТУУ “КПІ” 2008

Методичні вказівки до виконання лабораторних робіт з курсу „Мікропроцесорна техніка” „Мікроконтролери сімейства STMicroelectronics”для студентів спеціальності 7.09.08.03 – "Електронні системи" всіх форм навчання. - К.: НТУУ “КПІ”, 2008. – 51 с.

Навчальне видання

Методичні вказівки до виконання курсових робіт з курсу „Мікропроцесорна техніка” „Мікроконтролери сімейства STMicroelectronics”для студентів спеціальності 7.09.08.03 – "Електронні системи" всіх форм навчання

Укладач Терещенко Тетяна Олександрівна
Петергеря Юлія Сергійовна
Хохлов Юрій Вітальович

Рецензенти: В.В. Рогаль, доц., канд.техн.наук
В.А. Тодоренко, доц., канд.техн.наук

Редактор

ЗМІСТ

ВСТУП

Метою виконання лабораторних робіт є закріплення теоретичних знань основних принципів побудови і функціонування мікропроцесорних пристроїв на базі однокристальних мікроконтролерів, та використання набутих практичних навиків написання програм на асемблері та вивчення програмних та паратних засобів відлагодження програм.

У даних методичних вказівках наведено варіанти завдань для п‘яти лабораторних робіт, що охоплюють відомості по основним функціональним блоках мікроконтролерів сімейства STMicroelectronics, а саме портам введення та виведення інформації, системи переривань, інтерфейсу SPI, таймеру та АЦП.

В кожній роботі наведено теоеретичні відомості, приклад виконання та контрольні запитання.

У Додатках докладно описано програмне середовище та процес програмного відлагодження програм, описаний макет для апаратного відлагодження програм та надана система команд мікроконтролера.

Матеріали даних методичних вказівок може бути використано і при дипломному проектуванні.

Лабораторна робота №1

Порти введення та виведення мікроконтролерів ST7

Завдання:
Ініціалізувати порти, змінювати режим світіння світлових діодів після кожного натискання на кнопку S5 згідно табл.1.2. Режим світіння світлових діодів задано в табл. 1.1.
При роботі у вказаних в таблиці режимах світіння додатково включати із заданою частотою *F* звуковий сигнал та електричний двигун. Плавне регулювання яскравості світлодіодів організувати використовуючи широтно-імпульсну модуляцію.

Порядок роботи з макетом:

- Увімкнути всі перемикачі в блоці перемикачів B1/S6.
- Увімкнути джампер W7.
- Якщо в лабораторній роботі використовується звуковий сигнал, то увімкнути джампер W4 та вимкнути світлодіод №3 за допомогою перемикача в B1/S6.
- Якщо в лабораторній роботі використовується електричний двигун, то увімкнути джампер W3.
- Всі інші джампери вимкнути.

Таблиця 1.1. Режими світіння*

Режим	**Опис**
1	Послідовно засвічувати світлові діоди так, щоб утворилася „крапка, що біжить”. Час світіння кожного світлодіода дорівнює T_1. Напрямок пересування: від малих номерів світлодіодів до великих.
2	Послідовно засвічувати світлові діоди так, щоб утворилася „крапка, що біжить”. Час світіння кожного світлодіода дорівнює T_1. Напрямок пересування: від великих номерів світлодіодів до малих.
3	Послідовно засвічувати та гасити кожнен парний та непарний світловий діод. Час світіння кожного світлодіода дорівнює T_1.
4	Циклічно засвічувати та гасити парні світлодіоди. Час світіння кожного світлодіода дорівнює T_1.
5	Циклічно засвічувати та гасити непарні світлодіоди. Час світіння кожного світлодіода дорівнює T_1.
6	Циклічно засвічувати та гасити світлодіоди 1, 2, 5, 6. Час світіння кожного світлодіода дорівнює T_1. Плавно змінювати яскравість світлодіодів від мінімальної до максимальної та навпаки за проміжок часу T_2.
7	Циклічно засвічувати та гасити світлодіоди 3, 4, 7, 8. Час світіння кожного світлодіода дорівнює T_1. Плавно змінювати яскравість світлодіодів від мінімальної до максимальної та навпаки за проміжок часу T_2.
8	Циклічно засвічувати та гасити світлодіоди 1, 2, 3, 4. Час світіння кожного світлодіода дорівнює T_1. Плавно змінювати яскравість світлодіодів від мінімальної до максимальної та навпаки за проміжок часу T_2.
9	Циклічно засвічувати та гасити світлодіоди 5, 6, 7, 8. Час світіння кожного світлодіода дорівнює T_1. Плавно змінювати яскравість світлодіодів від мінімальної до максимальної та навпаки за проміжок часу T_2.
10	Циклічно засвічувати та гасити всі світлодіоди. Час світіння кожного світлодіода дорівнює T_1. Плавно змінювати яскравість світлодіодів від мінімальної до максимальної та навпаки за проміжок часу T_2.

Режим	Опис
11	Послідовно (за проміжок часу T_2) засвічувати світлові діоди поступово збільшуючи яскравість кожного наступного світлодіода так, щоб утворилася „крапка, що біжить”. Яскравість світлодіода №1 мінімальна, №8 — максимальна. Час світіння кожного світлодіода дорівнює T_1. Напрямок пересування: від малих номерів світлодіодів до великих.
12	Послідовно (за проміжок часу T_2) засвічувати світлові діоди поступово збільшуючи яскравість кожного наступного світлодіода так, щоб утворилася „крапка, що біжить”. Яскравість світлодіода №1 мінімальна, №8 — максимальна. Час світіння кожного світлодіода дорівнює T_1. Напрямок пересування: від великих номерів світлодіодів до малих.

* - якщо в лабораторній роботі використовується звуковий сигнал, то засвічувати світлодіод №3 не потрібно.

Таблиця 1.2. Завдання для лабораторної роботи 1

№ варі-анту	Режими світіння	Увімкнути звуковий сигнал в наступних режимах:	Увімкнути електричний двигун в наступних режимах:	Час T_1, с	Час T_2, с	Час-тота F, Гц
1.	1, 3, 6, 11	1, 6		0,1	0,5	100
2.	2, 3, 7, 12		2, 7	0,25	0,4	1000
3.	1, 3, 8, 11	1, 8		0,5	0,3	500
4.	2, 4, 9, 12		4, 9	0,8	1,0	1500
5.	1, 4, 10, 11	4, 10		0,2	0,6	1000
6.	2, 4, 6, 12		4, 12	0,6	0,7	2000
7.	1, 5, 7, 11	5, 7		0,9	1,0	1500
8.	2, 5, 8, 12		2, 8	0,5	0,2	2500
9.	1, 5, 9, 11	1, 9		0,1	0,3	2000
10.	2, 3, 10, 11		3, 10	0,5	0,6	3000

Теоретичні відомості

Порти введення-виведення. Мікроконтролер ST7 має 15 ліній введення/виведення – 7 ліній порту А та вісім порту В.

Кожен вивід порту може бути запрограмованний на введення або на виведення інформації. До того ж, окремі виводи мають декілька інших функцій, як-то: зовнішнє переривання, дублювати сигнал введення/виведення для периферійного пристрою на кристалі або для аналогового введення.

Для керування лініями порт МК має по три регістри для кожного порту, табл. 1.3 - це регістр даних порта (PADR та PBDR), регістр напряму передачі даних (PADDR, PBDDR), регістр опцій (PAOR , PBOR).

Таблиця 1.3.- Призначення регістрів портів

Порт	Позначення регістрів		Призначення
Порт А	PADR	Port A Data Register	Регістр даних порта А
	PADDR	Port A Data Direction Register	Регістр напряму передачі даних порта А
	PAOR	Port A Option Register	Регістр опцій порта А
Порт В	PBDR	Port B Data Register	Регістр даних порта В
	PBDDR	Port B Data Direction Register	Регістр напряму передачі даних порта В
	PBOR	Port B Option Register	Регістр опцій порта В

Скидання DDRx біта в 0 вибирає режим введення. Установка DDRx біта в 1 визначає режим виведення. Дія бітів регістру опцій DORx в режимах введення та виведення відображена в табл 1.4.

Таблиця 1.4- Конфігурація ліній портів

Режим конфігурації		DDR	DOR
Введення	Високоімпедансний вхід * (без підтягувального резистора)	0	0
	Вхід з підтягувальним резистором	0	1
Виведення	За двотактною схемою (Push-Pull)	1	1
	За схемою з відкритим стіком (Open Drain)	1	0

* Початковий стан

При використанні лінії для зовнішнього переривання або для аналогового входу АЦП ii треба запрограмувати на режим введення.

Ініціалізація портів полягає у запису даних в регістри напрямку та опцій:

```
init_portA:   ld    A,#INITPADDR   ; 0 -  вхід,  1 -  вихід відповідного біту регістра DDR
              ld    PADDR,A
              ld    A,#INITPAOR    ; 0 – для вхідного виводу - високоімпедансний вхід; для вихідного виходу – вихід за схемою з відкритим колектором;
                                   1 – для вхідного виводу – вхід з підтягувальним резистором; для вихідного виводу – вихід за двотактною схемою
              ld    PAOR,A
              ret                  ; повернення до головної програми

init_portB:   ld    A,#INITPBDDR   ; 0 -  вхід,  1 -  вихід відповідного біту регістра DDR
              ld    PBDDR,A
              ld    A,#INITPBOR    ; 0 – для вхідного виводу - високоімпедансний вхід; для вихідного виводу – вихід за схемою з відкритим колектором;
                                   1 – для вхідного виводу  – вхід з підтягувальним резистором; для вихідного виводу – вихід за двотактною схемою
              ld    PBOR,A
              ret                  ; повернення до головної програми
```

Після цього можна виконувати введення та виведення через порти даних. Підпрограма запису в порт має вигляд:

```
write_portB:  ld    A,portB_TX     ; Завантаження значення вмісту комірки пам’яті portB_TX в акумулятор
              ld    PBDDR,A        ; завантаження вмісту регістра DR порту B з акумулятора
              ret                  ; повернення до головної програми
```

Підпрограма читання порту має вигляд:

```
read_portB:   ld    A,PBDR          ; завантаження вмісту регістра PBDR порту B в
                                      акумулятор
              ld    portB_RX,A      ; Завантаження вмісту акумулятора у комірку
                                      пам'яті portB_RX
              ret                   ; повернення до головної програми
```

Приклад виконання лабораторного завдання

Завдання:
Циклічно засвічувати та гасити світлодіод №8. Період переключення світлодіода 0,5 сек.

Текст програми:

```
;*******************************************************************
;
; ІНІЦІАЛІЗАЦІЯ ПОРТІВ ST7
;
;*******************************************************************
init_ST7:
      clr  MCCSR              ; нормальний режим
      ret

init_led_ports:
      push a                  ;
      ld   a, #%10000000      ; переключаємо лінію 7 порту A у режим
      ld   PADDR, a           ; виводу (push-pull)
      ld   a, #%10000000
      ld   PAOR, a
      pop  a                  ;
      ret

;*******************************************************************
;
; МІСЦЕ ДЛЯ ПІДПРОГРАМ
;
;*******************************************************************
write_portA:
      ld   A,portA_TX         ; Завантаження значення вмісту комірки
                              ; пам'яті portA_TX в акумулятор
      ld   PADR,A             ; завантаження вмісту регістра DR порту A
                              ; з акумулятора
      ret                     ; повернення

read_portA:
      ld   A,PADR             ; завантаження вмісту регістра PADR
                              ; порту A в акумулятор
      ld   portA_RX,A         ; Завантаження вмісту акумулятора у
                              ; комірку пам'яті portA_RX
      ret                     ; повернення

led_on:
      push a                  ;
      call read_portA
      ld   a, portA_RX
      or   a, #%10000000      ; ВМИКАЄМО світлодіод №8
      ld   portA_TX, a
      call write_portA
      pop  a                  ;
      ret

led_off:
      push a                  ;
      call read_portA
      ld   a, portA_RX
      and  a, #%01111111      ; ВИМИКАЄМО світлодіод №8
      ld   portA_TX, a
```

```
      call write_portA
      pop  a                ;
      ret

delay:                      ; Підпрограма затримки
                            ; 256*(256*(3+4)+3+3+2)+3+4+2+5+4+6 ≈ 0,5 сек
      push x
      push y
      ld   x,#$ff
dec_2:
      ld   y,#$ff
dec_1:
      dec  y
      JRNE dec_1
      dec  x
      JRNE dec_2
      pop  y
      pop  x
      ret

;*********************************************************************
;
; ГОЛОВНА ПРОГРАМА ST7
;
;*********************************************************************
main:
      RSP                   ; Скидаємо покажчик стеку
      sim                   ; Маскуємо переривання
      call init_ST7         ; Ініціалізація
      call init_led_ports
start:
      call led_on           ; ВМИКАЄМО світлодіод
      call delay            ; затримка
      call led_off          ; ВИМИКАЄМО світлодіод
      call delay            ; затримка
      JP   start            ; зациклюємо програму
      JP   main
```

Контрольні запитання

1. З якою метою створена *RISC*-архітектура?
2. Назвіть переваги *RISC*-архітектури над *CISC*-архітектурою.
3. Які недоліки має *RISC* –архітектура?
4. Дайте характеристику ОМК сімейства ST7
5. Як відбувається звернення до портів введення-виведення?
6. Які можливості конфігурації мають порти введення-виведення?
7. Назвіть режими введення
8. Назвіть режими виведення.
9. Які засоби програмування мають порти введення виведення?

Лабораторна робота №2

Система переривань мікроконтролерів ST7

Завдання:

Написати програму керування заданим світлодіодом та звуковим сигналом за допомогою кнопок S4 та S5 згідно заданому у варіанті способу (таблиці 2.1 та 2.2). Для цього ініціалізувати систему переривань для роботи у заданому режимі та організувати обробку відповідних переривань.
Модифікувати програму, що була розроблена в лабораторній роботі №1, організувавши обробку натискання кнопки S5 з використанням системи переривань.

Порядок роботи з макетом:

- Вимкнути всі перемикачі в блоці перемикачів B1/S6.
- Вимкнути джампери W4 та W6.
- Увімкнути джампери W3, W5 та W7.

Таблиця 2.1. - Способи керування світлодіодом:

Режим	Опис
1.	Засвічувати та гасити світлодіод заданою кнопкою по задньому фронту та низькому рівню. Періодично один раз на секунду засвічувати світлодіод на короткий час при утриманні кнопки в натиснутому стані більше 1 секунди.
2.	Засвічувати та гасити світлодіод заданою кнопкою по передньому фронту. Кожне третє натискання кнопки ігнорувати.
3.	Засвічувати та гасити світлодіод заданою кнопкою по задньому фронту. Після кожного третього натискання кнопки видавати короткий звуковий сигнал.
4.	Засвічувати та гасити світлодіод заданою кнопкою по передньому та задньому фронту. Кожне третє натискання кнопки ігнорувати.

Таблиця 2.2. Завдання для лабораторної роботи 2

№ варіанту	Спосіб керування	Кнопка
1.	1	S4
2.	2	S4
3.	3	S4
4.	4	S4
5.	1	S5
6.	2	S5
7.	3	S5
8.	4	S5
9.	1	S4
10.	2	S5

Теоретичні відомості

Переривання. МК ST7 має два різні типи переривань: масковані і не масковані. Не масковане програмне переривання активується командою TRAP і виконується незалежно

від стану біта I. Типи переривань і початкові адреси підпрограм їх обробки наведено в табл. 2.3.

Перед виконанням переривання в стек записується: адреса команди, на яку треба повернути після обробки переривання (зміст регістра PC), регістри X, A, CC. Зміну всіх параметрів, що зв'язані з перериваннями рекомендують робити при заборонених перериваннях. Програма обробки переривання повинна закінчитися командою IRET, яка відновлює значення збережених регістрів із стека.

Таблиця 2.3. - Джерела переривань

№	Джерело	Опис	Пріоритет	Адреси підпрограми (вектор)
1	RESET	Скидання	Високий пріоритет	FFFEH – FFFFH
2	TRAP	Програмне переривання		FFFC – FFFDH
3	AWU	Автопробудження від переривання		FFFA – FFFBH
4	ei0	Зовнішне переривання 0		FFF8H – FFF9H
5	ei1	Зовнішне переривання 1		FFF6H – FFF7H
6	ei2	Зовнішне переривання 2		FFF4H – FFF5H
7	ei3	Зовнішне переривання 3		FFF2H – FFF3H
8	LITE TIMER	LITE таймер RTC2 – переривання по переповненню		FFF0H – FFF1H
9		Не використовується		FFEEH – FFEFH
10	SI	AVD переривання		FFECH – FFEDH
11	AT TIMER	AT таймер, який порівнює виходне переривання з входним		FFEAH – FFEBH
12		AT таймер – переривання по переповненню		FFE8H – FFE9H
13	LITE TIMER	LITE таймер – захват вхідного переривання		FFE6H – FFE7H
14		LITE таймер RTC1– переривання по переповненню		FFE4H – FFE5H
15	SPI	SPI переривання введеня/виведення		FFE2H – FFE3H
16		Не використовується	Низький пріоритет	FFE0H-FFE1H

Як видно з табл. 2.3, МК може обробляти 4 зовнішні переривання, причому можна програмно задати як чутливість (тип сигналу) переривання, так і номер виводу ВІС МК. Тип сигналу задається за допомогою регістра керування зовнішніми перериваннями EICR (EXTERNAL INTERRUPT CONTROL REGISTER).Формат регістра EICR подано на рис.2.1.

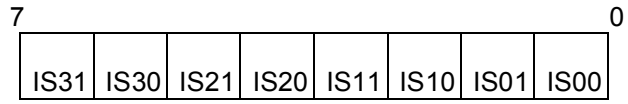


Рисунок 2.1 - Формат регістра EICR

Біти IS0[1:0] , IS1[1:0], IS2[1:0], IS3[1:0] , визначають чутливість зовнішнього переривання 0-3. відповідно згідно з табл. 2.4.

Таблиця 2.4. Чутливість зовнішнього переривання

Isx1	Isx0	Чутливість зовнішнього переривання
0	0	По спаду і низькому рівню
0	1	Тільки по фронту
1	0	Тільки по спаду
1	1	По спаду і по фронту

Номер виводу ВІС МК задається за допомогою регістру вибору зовнішнього переривання EISR (EXTERNAL INTERRUPT SELECTION REGISTER) Формат регістра EISR подано на рис.2.2.

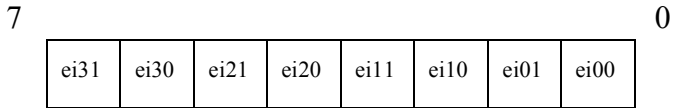


Рисунок 2.2. Формат регістра EI**S**R

Біти ei3[1:0] вибірають лінію введення/виведення для зовнішнього переривання 3 відповідно до табл. 2.5.

Таблиця 2.5. - Вибір зовнішнього переривання 3

ei31	ei30	Лінії введення/виведення
0	0	PB0*
0	1	PB1
1	0	PB2

* Стан скидання

Біти ei2[1:0] вибірають лінію введення/виведення для зовнішнього переривання 2 відповідно до табл. 2.6.

Таблиця 2.6. Вибір зовнішнього переривання 2

ei21	ei20	Введення/виведення
0	0	PB3*
0	1	PB4
1	0	PB5
1	1	PB6

* Стан скидання

Біти ei1[1:0] вибірають лінію введення/виведення для зовнішнього переривання 1 відповідно до табл. 2.7.

Таблиця 2.7 - Вибір зовнішнього переривання 1

ei11	ei10	Введення/виведення
0	0	PA4
0	1	PA5
1	0	PA6
1	1	PA7*

* Стан скидання

Біти ei0[1:0] вибирають лінію введення/виведення для зовнішнього переривання 0 відповідно до табл. 2.8.

Таблиця 2.8. Вибір зовнішнього переривання 0

ei01	ei00	Введення/виведення
0	0	PA0*
0	1	PA1
1	0	PA2
1	1	PA3

* Стан скидання

Приклад виконання лабораторного завдання

Завдання:
Циклічно засвічувати та гасити світлодіод №8. Період переключення світлодіода 0,5 сек.
При натисканні кнопки S4 передчасно гасити світлодіод №8.

Текст програми:

```
;*******************************************************************
;
; ІНІЦІАЛІЗАЦІЯ ПОРТІВ ТА ПЕРЕРИВАНЬ ST7
;
;*******************************************************************
init_ST7:
      clr  MCCSR            ; нормальний режим
      ret

init_led_ports:
      push a                ;
      ld   a, #%10000000    ; переключаємо лінію 7 порту A у режим
      ld   PADDR, a         ; виводу (push-pull) та лінію 3 порту A
      ld   a, #%10001000    ; у режим введення (floating/pull-up
                            ; interrupt)
      ld   PAOR, a
      pop  a                ;
      ret

init_interrupt:
      push a
      ld   a,#%00000010     ; настроюємо EICR на зовнішнє переривання ei0
                            ; (режим: Falling edge only)
      ld   EICR,a
      ld   a,#%00000011     ; настроюємо EISR на зовнішнє переривання ei0
                            ; по лінії PA3
      ld   EISR,a
      pop  a
      ret
```

```
;*******************************************************************
;
; ГОЛОВНА ПРОГРАМА ST7
;
;*******************************************************************
main:
      RSP                     ; Скидаємо покажчик стеку
      sim                     ; Маскуємо переривання
      call  init_ST7          ; Ініціалізація (див. приклад для ЛР1)
      call  init_led_ports
start:
      call  led_on            ; ВМИКАЄМО світлодіод (див. приклад для ЛР1)
      call  delay             ; затримка (див. приклад для ЛР1)
      call  led_off           ; ВИМИКАЄМО світлодіод (див. приклад для ЛР1)
      call  delay             ; затримка (див. приклад для ЛР1)
      JP    start             ; зациклюємо програму
      JP    main
```

Внести зміни у відповідні частини заготовки програми:

```
;*******************************************************************
;
; МІСЦЕ ДЛЯ ПІДПРОГРАМ ПЕРЕРИВАНЬ
;
;*******************************************************************

ext0_rt:
      sim
      push a
      push x
      push y
      call led_off
      call delay
      call delay
      call delay
      pop y
      pop x
      pop a
      rim
      IRET

;*******************************************************************
;
; ДЕКЛАРУВАННЯ ВЕКТОРІВ ПЕРЕРИВАННЯ
;
;*******************************************************************

ext0_it DC.W ext0_rt          ; Adresse FFF8-FFF9h
```

Контрольні запитання

1. Які особливості організації стека МК ST7?
2. Яка структура пам‘яті МК ST7?
3. Назвіть джерела тактових сигналів МК ST7
4. Назвіть випадки, коли МК ST7 входить в режим МК ST7 скидання.
5. На які види сигналів зовнішнього переривання реагує МК ST7?
6. Як відбувається перехід на підпрограму переривання?

Лабораторна робота №3

Інтерфейс SPI мікроконтролерів ST7

Завдання:

Ініціалізувати інтерфейс SPI, обравши для цього лінію порту PB1 для сигналу SCK, PB2 — для сигналу MISO, PB3 — для сигналу MOSI згідно завданням табл.. 3.1. та 3.2. Ініціалізувати контролер світлодіодного дисплею MAX7219. Вивести на світлодіодний дисплей інформацію та змінювати покази світлодіодного дисплею способом згідно свого варіанту по сигналу з кнопок.

Порядок роботи з макетом:

- Вимкнути всі перемикачі в блоці перемикачів B1/S6.
- Вимкнути джампери W4 та W6.
- Увімкнути джампери W3, W5 та W7.

Таблиця 3.1. - Способи керування світлодіодним дисплеєм:

Режим	**Опис**
1.	Забезпечити блимання одного з розрядів дисплею. Кнопкою *A* перемістити блимання на сусідній розряд праворуч. Кнопкою *B* збільшувати на одиницю число, що відображається розрядом, який блимає. При утриманні кнопки *B,* число в розряді змінювати зі швидкістю 5 чисел на секунду.
2.	Забезпечити блимання одного з розрядів дисплею. Кнопкою *A* перемістити блимання на сусідній розряд ліворуч. Кнопкою *B* зменшувати на одиницю число, що відображається розрядом, який блимає. При утриманні кнопки *B,* число в розряді змінювати зі швидкістю 3 чисел на секунду.
3.	Вивести на дисплей довільне число. Після натискання кнопки *A* почати на одиницю збільшувати число на дисплеї зі швидкістю *N* чисел на секунду. Після натискання кнопки *B* плавно змінити швидкість зміни чисел до 0. Після повної зупинки увімкнути на короткий час електричний двигун.
4.	Вивести на дисплей довільне число. При утриманні кнопки *A* збільшувати число, що відображається розрядами 1 та 2, а при утриманні кнопки *B* — розрядами 3 та 4. Числа змінювати зі швидкістю *N* чисел на секунду. Після відпускання кнопки плавно зупинити зміну числа у відповідних розрядах. Після повної зупинки увімкнути на короткий час електричний двигун.
5.	Організувати смугу прокрутки послідовності 16-значних цифр. Напрямок прокрутки змінювати кнопкою *A*. Швидкість прокрутки дискретно змінювати кнопкою *B* від 0 до *N* зсувів на один розряд на секунду із кроком *N*/5 по колу. При досяганні максимальної швидкості *N,* увімкнути на короткий час електричний двигун.
6.	Організувати смугу прокрутки послідовності 16-значних цифр. Напрямок прокрутки змінювати кнопкою *A*. Швидкість прокрутки плавно змінювати при утриманні кнопки *B* від 0 до *N* зсувів на один розряд на секунду. При досяганні максимальної швидкості *N*, увімкнути на короткий час електричний двигун, зачекати 0,5 секунди, скинути швидкість до 0, зачекати 0,2 секунди, почати знову збільшувати швидкість.

Режим	Опис
7.	Запрограмувати електронний таймер зворотного відліку. Кнопкою *A* таймер переводиться в режим програмування. У режимі налаштування один з розрядів повинен блимати, кнопкою *A* блимання переміщувати на сусідній розряд праворуч, а при утриманні кнопки *B,* число в розряді збільшувати зі швидкістю 5 чисел на секунду. Налаштування завершується після встановлення чисел у всі 4 розряди. Кнопкою *B* налаштований таймер запускається у роботу. Змінювати числа у 4-му розряді зі швидкістю *N* чисел на секунду. При досяганні нуля, увімкнути на короткий час електричний двигун.

Таблиця 3.2. Завдання для лабораторної роботи 3

№ варі-анту	Спосіб керування	Кнопка A	Кнопка B	Швидкість N
1.	1, 3	S4	S5	10000
2.	2, 4	S4	S5	100
3.	1, 5	S4	S5	25
4.	2, 6	S4	S5	25
5.	1, 7	S4	S5	5
6.	2, 3	S5	S4	6000
7.	1, 4	S5	S4	50
8.	2, 5	S5	S4	50
9.	1, 6	S5	S4	50
10.	2, 7	S5	S4	10

Теоретичні відомості

Послідовний периферійний інтерфейс SPI (Serial Peripheral Interface) призначенний для обміну даних у послідовному форматі між мікроконтролером і різноманітними периферійними пристроями або між декількома мікроконтролерами ST7.

Дані для передачі, а також прийняті дані записуються в регістр введення/виведення. даних SPIDR. При обміні даними по інтерфейсу SPI мікроконтролер може працювати як у режимі Master, так і в режимі Slave.

Схема підключення двох BIC МК по інтерфейсу SPI наведена на рис.3.1.

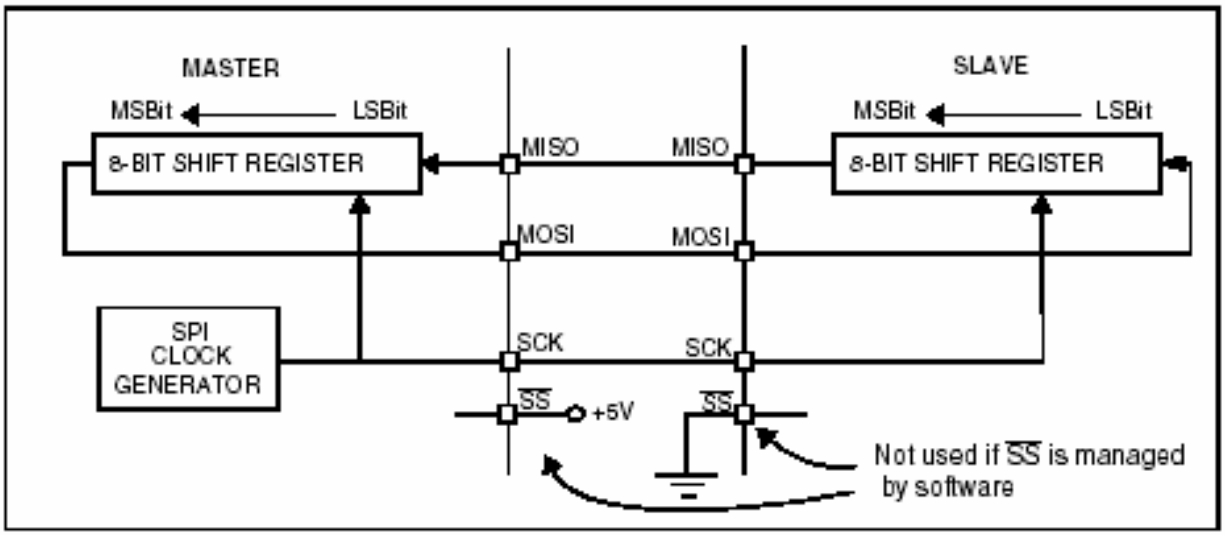


Рисунок 3.1 - Схема підключення двох BIC МК по інтерфейсу SPI

Обмін по протоколу SPI здійснюється за допомогою 4 виводів ВІС:
– MISO (Master In / Slave Out data) – вхід даних для ведучого Master пристрою та вихід для веденого Slave пристрою;
– MOSI (Master Out / Slave In data) - вихід даних для ведучого та вхід Master пристрою для веденого Slave пристрою;
– SCK (Serial Clock out by SPI masters and input by SPI slaves) - тактові імпульси (генеруються ведучим пристроєм і є входними для веденого пристрою)
– SS (Slave select) вибір пристрою : 0 для Slave і 1 для Master пристрою.

Регістр керування SPICR (рис. 3.2) задає режим *Master* /*Slave*, частоту послідовного обміну, фазу та полярність імпульсів, дозвіл переривання.

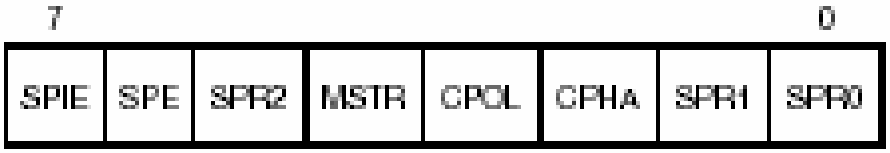


Рисунок 3.2. Формат регістра керування SPI SPICR

На рис. 3.2 позначено:
Біт **SPIE (***Serial Peripheral Interrupt Enable)*- дозвіл переривання SPI;
Біт **SPE (***Serial Peripheral Output Enable)* - дозвіл виходу SPI;
Біт **SPR2 (***Divider Enable)* – дозвіл ділення частоти (табл. 3.3);
Біт **MSTR (***Master Mode)*- режим *Master;*
Біт **CPOL (***Clock Polarity)* - полярність імпульсів*;*
Біт **CPHA (***Clock Phase)*- фаза імпульсів;
Біти **SPR[1:0] (***Serial Clock Frequency)*- завдання частоти (табл. 3.3).

Таблиця 3.3 – Завдання частоти роботи SPI

Serial Clock	SPR2	SPR1	SPR0
$f_{CPU}/4$	1	0	0
$f_{CPU}/8$	0	0	0
$f_{CPU}/16$	0	0	1
$f_{CPU}/32$	1	1	0
$f_{CPU}/64$	0	1	0
$f_{CPU}/128$	0	1	1

Діаграми обміну по інтерфейсу SPI (рис. 3.4) пояснюють вибір полярності та фази тактових імпульсів.

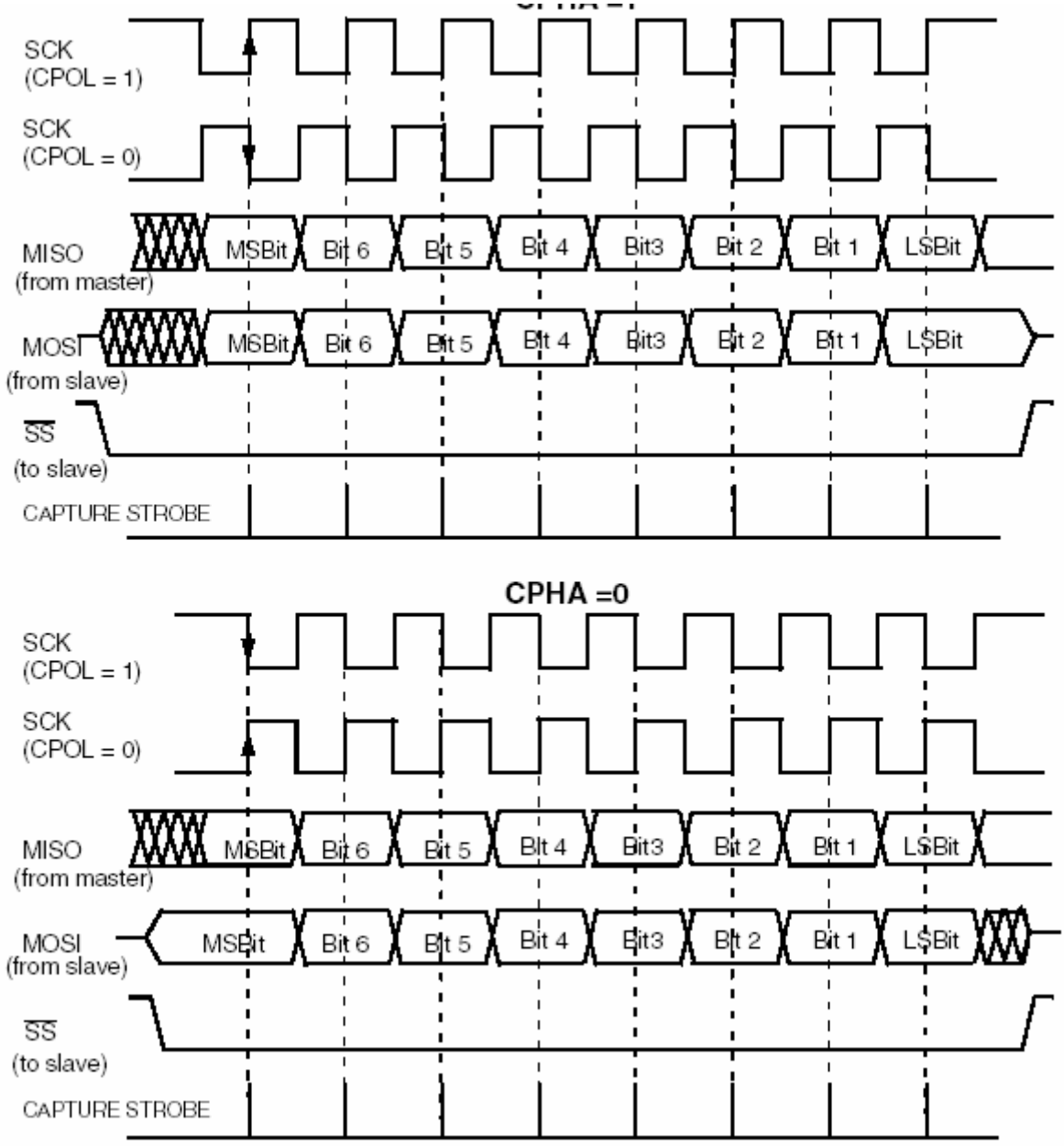


Рисунок 3.4 - Діаграми обміну по інтерфейсу SPI

Регістр керування/статусу SPICSR (рис. 3.5) містить наступні біти:

Рисунок 6.45. Формат регістра керування/ статусу SPICSR

Біт **SPIF (***Serial Peripheral Data Transfer Flag)* - прапорець завершення передачі*;*
Біт **WCOL (***Write Collision status)* - помилка запису - завантаження регістру даних SPI під час передавання;
Біт **OVR (**S*PI Overrun error*) – помилка переповнення;
Біт **MODF (***Mode Fault flag)* - прапорець аварійного режиму*;*
Біт **SOD (***SPI Output Disable)* – вихід SPI заборонений;
Біт **SSM (***SS Management)* - при 0 керування від зовнішнього пристрою*;*
Біт **SSI (***SS Internal Mode)* – при 0 вибір пристрою Slave.

Приклад виконання лабораторного завдання

Завдання:

Вивести на 4-розрядний семисегментний дисплей число «1980» застосувавши для цього SPI інтерфейс.

Текст програми:

```
;*********************************************************************
;
; ІНІЦІАЛІЗАЦІЯ ПОРТІВ ST7
;
;*********************************************************************
init_ST7:
      clr   MCCSR             ; нормальний режим
      ret

init_IO:
      ld    A,#%00000000      ; Настроювання регістру PADDR
                              ; (настройка порту A для введення
                              ; для всіх виводів)
      ld    PADDR,A

      ld a, #%00001000        ; PA3 в режимі input/interrupt
      ld    PAOR,A

      ld    A,#%00001110      ; Настроювання регістру PBDDR
                              ; (настройка порту B на виведення для
                              ; виводів 1,2,3)
      ld    PBDDR,A
      ld    A,#%00001110      ; Настроювання регістру PBOR
                              ; (setup port's B pins 1,2,3
                              ; for push-pull output)
      ld    PBOR,A
      ret

;*********************************************************************
;
; ІНІЦІАЛІЗАЦІЯ SPI
;
;*********************************************************************
init_SPI:
      ld    A,#%00000011      ; Настроювання регістру SPISR
                              ; (SPI status register)
      ld    SPISR,A           ; Вмикаємо режим Master для SPI (SSM=1, SSI=1)
      ld    A,#%01011100      ; Настроювання регістру SPICR
                              ; (SPI control register)
                              ; CPHA=1 -- SPI дані запам’ятовуються по
                              ; задньому фронту тактового імпульсу
                              ; CPOL=1 -- вивід SCK в стані очікування
                              ; буде "1"
                              ; MSTR=1 -- режим Master. Функція виводу SCK
                              ; змінюється з режиму введення в режим виведення
                              ; та функції виводів MISO та MOSI
                              ; є зарезервованими.
                              ; SPE=1 -- Serial Peripheral Output Enable
                              ; (альтернативні функції виводів SPI активовані)
      ld    SPICR,A           ;
      ret

;*********************************************************************
;
; ГОЛОВНА ПРОГРАМА ST7
;
;*********************************************************************
main:
      RSP                     ; Скидаємо покажчик стеку
      Sim                     ; Маскуємо переривання
      call init_ST7           ; Ініціалізація ST7
      call init_IO            ; Ініціалізація портів
      call init_SPI           ; Ініціалізація SPI
      call MAX7219_Init       ; Ініціалізація MAX7219
      call MAX7219_Clear      ; Очистка дисплею
```

```
start:
      ld   A,#1                      ; Вибір розряду №1
      ld   DisplayChar_Digit,A
      ld   A,#1                      ; Вивід цифри „1”
      ld   DisplayChar_Character,A
      call MAX7219_DisplayChar
      ld   A,#2                      ; Вибір розряду №2
      ld   DisplayChar_Digit,A
      ld   A,#9                      ; Вивід цифри „9”
      ld   DisplayChar_Character,A
      call MAX7219_DisplayChar
      ld   A,#3                      ; Вибір розряду №3
      ld   DisplayChar_Digit,A
      ld   A,#8                      ; Вивід цифри „8”
      ld   DisplayChar_Character,A
      call MAX7219_DisplayChar
      ld   A,#4                      ; Вибір розряду №4
      ld   DisplayChar_Digit,A
      ld   A,#0                      ; Вивід цифри „0”
      ld   DisplayChar_Character,A
      call MAX7219_DisplayChar

      JP  start              ; зациклюємо програму
      JP   main
```

Контрольні запитання

1. Назвіть режими енергоспоживання МК ST7 в порядку зменшення споживання.
2. В чому полягає відмінність режимів холостого ходу та мікроспоживання?
3. Чим відрізняється режим зберігання енергії від режиму мікроспоживання?
4. Чим відрізняється основний режим очікування від режиму мікроспоживання?
5. Яке призначення інтерфейсу SPI МК ST7?

Лабораторна робота №4

Таймер ART мікроконтролерів ST7

Завдання:
Ініціалізувати таймер. Використовуючи таймер забезпечити блимання першого світлодіода із заданою частотою *F* у двох режимах: дискретному та плавному (використовуючи *PWM* – широтно-імпульсну модуляцію). Кнопкою *A* змінювати режими блимання між собою. (табл. 4.1). Кнопкою *B* вмикати та вимикати блимання.
Модифікувати програму, що була розроблена в лабораторній роботі №3, організувавши необхідні затримки за допомогою таймеру.

Порядок роботи з макетом:

- Вимкнути всі перемикачі в блоці перемикачів B1/S6.
- Вимкнути джампери W4 та W6.
- Увімкнути джампери W3, W5 та W7.

Таблиця 4.1. Завдання для лабораторної роботи 4

№ варі-анту	Кнопка A	Кнопка B	Частота F, Гц
1.	S4	S5	1
2.	S5	S4	3
3.	S4	S5	5
4.	S5	S4	7
5.	S4	S5	10
6.	S5	S4	4
7.	S4	S5	2
8.	S5	S4	6
9.	S4	S5	12
10.	S5	S4	3

Теоретичні відомості

Таймер ART являє собою 12- бітний таймер з автоперезавантаженням
Призначений для:

- Реалізації часових затримок;
- Генерації переривань при переповненні, захопленні, порівнянні;
- Генерації 4 незалежних ШІМ сигналів.

Таймер заснований на автономному (несинхронізованному) 12-бітовому інкрементному лічильнику з вхідним автоперезавантажувальним регістром і 4-мя вихідними PWM каналами
Має 6 зовнішніх виводів:

- 4 виходи ШІМ;
- вхід ATIC (AT input capture) для функції захоплення введення;
- вхід BREAK для ШІМ (для припинення сигналу на PWM виводах).

Основні характеристики ART наступні:
- Частота лічби - 2КГц-4МГц при f_{CPU}= 8 МГц;
- Керування полярністю сигналів на входах і виходах;
- Масковані переривання при переповненні таймеру, при порівнянні та захопленні.

Реалізації часових затримок. При переповненні 12-бітового лічильнику CNTR встановлюється прапор переповнення OVF в регістрі контроля/статусу ARTCSR, що свідчить про закінчення певного інтервалу часу. Цей інтервал можна змінювати записом коду в регістр автоперезавантаження ATR та зміною частоти лічильних імпульсів. За одиничним станом прапора переповнення OVF лічильник переходить з стану FFFh в стан коду ATR (значення автоперезавантаження).

Захоплення події. За переднім або заднім фронтом на виводі ATIC мікроконтролера вміст 12-бітного лічильника CNTR запам‘ятовується у регістрі ATICR, при цьому встановлюється біт ICF та якщо переривання дозволено (біт ICIE встановлений) воно генерується. Біт ICF скидається при читанні ATICR регістра. Регістр ATICR доступний лише для читання і він завжди містить значення н інкрементуючого лічильника, яке відповідає останньому захопленню вхідних даних. Будь-яке подальше захоплення даних забороняється, поки біт ICF виставлений.

Режим порівняння. Щоб використовувати цю функцію необхідно завантажити 12-бітне значення в регістри DCRxH і DCRxL Коли лічильник (CNTR) досягне значення, яке зберігається в DCRxH і DCRxL регістрах, то CMPF біт в PWMxCSR встановиться в 1 і згенерує запит на переривання, якщо встановлений біт CMPIE дозволу переривання.

Генерація ШІМ сигналів. PWM режим дозволяє згенерувати до 4-х ШІМ сигналів. PWMx вихідні сигнали можуть бути дозволені або заборонені бітом OEx в регістрі PWMCR. Чотири PWM сигнали мають однакову частоту (f_{PWM}), яка задається частотою лічильника і значенням регістра ATR як:

$$f_{PWM} = f_{COUNTER} / (4096 – ATR) \qquad (4.1)$$

З формули (4.1) можна зробити наступні виводи:

- Якщо $f_{COUNTER}$=32 МГц, максимальне значення f_{PWM}=8МГц (значення регістру ATR = 4092), мінімальне значення - 8 КГц (значення регістру ATR = 0)
- Якщо $f_{COUNTER}$=4 МГц, максимальне значення f_{PWM}=2МГц (значення регістру ATR = 4094), мінімальне значення – 1 КГц (значення регістру ATR = 0)

Функція останову (Break) активується зовнішнім сигналом BREAK на одноіменому виводі (активний рівень - 0), рис. 4.1.

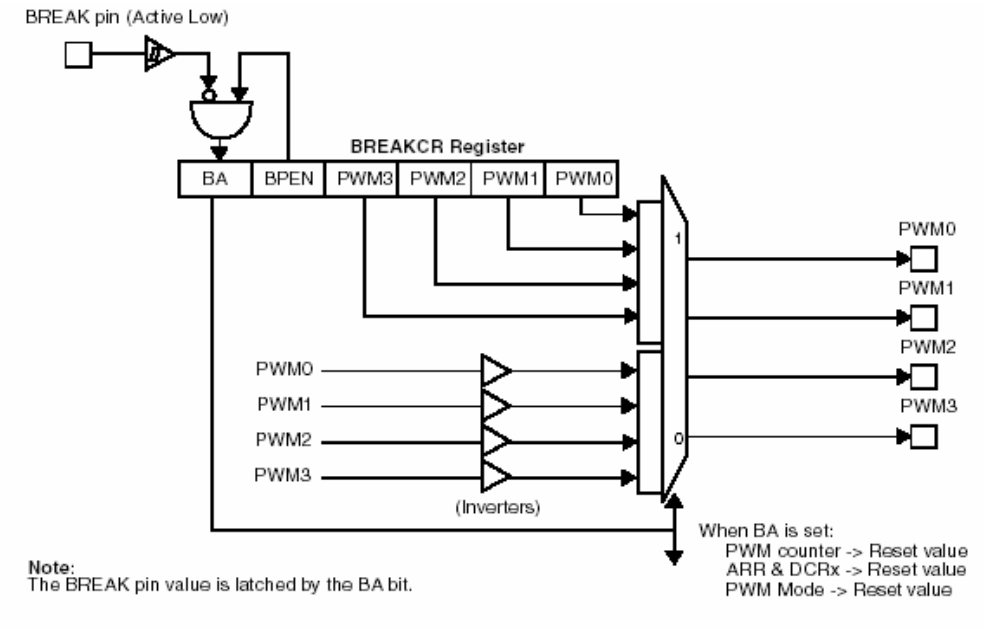


Рисунок 4.1 – Дія сигналу BREAK на ШІМ виходи

Для того, щоб використовувати BREAK вивід він повинен бути заздалегідь дозволений програмно, установкою BPEN бита в регістрі BREAKCR. Коли низький рівень визначений на BREAK виводі, то BA біт встановлюється в 1 і функція зупинки активується. Послідовність всіх 4 ШІМ припиняється, 12-бітний PWM лічильник, регістри ARR, PWMCR, DCRx і відповідні тіньові регістри встановлюються в свої початкові значення. при скиданні.

Опис регістрів таймеру.

Регістр керування/стану таймера ATCSR (AUTORELOAD TIMER CONTROL STATUS REGISTER)

Регістр допускає читання та запис. Значення скидання: 0x00 0000.

Формат регістра **ATCSR** подано на рис.4.2.

7 0

0	ICF	ICIE	CK1	CK0	OVF	OVFIE	CMPIE

Рисунок 4.2. Формат регістра ATCSR

На рис. 4.2 позначено:

Біт ICF (*Input Capture Flag*) - прапор вхідного захоплення. Цей біт встановлюється апаратно і скидається програмно при читанні ATICR регістра (дозвіл на читання з ATICRH або ATICRL скидатиме цей прапор). Запис в цей біт не міняє його значення. Якщо ICF= 1, то вхідне захоплення виконане.

Біт ICIE (*IC Interrupt Enable)* – дозвіл IC переривання. Цей біт встановлюється і скидається програмно. Якщо біт дорівнює 0, то переривання вхідного захоплення заборонено, якщо 1- то переривання вхідного захоплення дозволено.

Біти CK[1:0] (*Counter Clock Selection)* - вибір генератора лічильника. Ці біти встановлюються і скидаються програмно і скидаються апаратно після сигналу RESET. Вони визначають частоту генератора лічильника (табл.4.3). Зміна вступає в дію після переповнення.

Біт OVF (*Overflow Flag*) - прапор переповнення. Цей біт встановлюється апаратно і скидатися програмно при читанні TCSR регістра. Він відображає перехід стану лічильника з FFFh в ATR значення (автоперезавантаження). Якщо біт дорівнює 0, то переповнення лічильника не відбулося., якщо 1 – то відбулося.

Таблиця 4.3. Вибір генератора лічильника

Вибір генератора лічильника	CK1	CK0
Викл.	0	0
f_{LTIMER} (період1мс при f_{cpu}=8 Мгц)	0	1
f_{cpu}	1	0
32 МГц	1	1

Біт OVFIE (*Overflow Interrupt Enable*) – дозвіл переривання від переповнення. Цей біт встановлюється/скидається програмно і скидається апаратно після сигналу RESET. Якщо біт дорівнює 0, то OVF переривання відключене, якщо – 1, то OVF переривання дозволене.

Біт CMPIE (*Compare Interrupt Enable*) - дозвіл переривання захоплення. Цей біт встановлюється/скидається програмно і скидається апаратно після сигналу RESET. Якщо біт дорівнює 0, то OVF переривання заборонено, якщо – 1, то OVF переривання дозволено.

Старший регістр лічильника CNTRH (COUNTER REGISTER HIGH)

Регістр допускає тільки читання. Значення скидання: 0000 0000.

Формат регістра CNTRH подано на рис.4.4.

15 8

0	0	0	0	CNTR 11	CNTR 10	CNTR 9	CNTR 8

Рисунок 4.4 Формат регістра CNTRH

Молодший регістр лічильника CNTRL (COUNTER REGISTER LOW).
Регістр допускає тільки читання. Значення скидання: 0000 0000.
Формат регістра **CNTRL** подано на рис.4.5.

7 0

CNTR 7	CNTR 6	CNTR 5	CNTR 4	CNTR 3	CNTR 2	CNTR 1	CNTR 0

Рисунок 4.5. Формат регістра CNTRL

Біти CNTR [11:0] (*Counter Value)* – значення лічильника. Лічильник інкрементується з кожним тактовим імпульсрм, як тільки генератор лічильника вибраний (див. табл. 4.32). Для зчитування значення лічильника використовують дві послідовні операції читання. При переповненні лічильника в нього заноситься значення ATR регістра.

Автоперезаватажувальний регістр ATRH (*AUTORELOAD REGISTER HIGH)*
Регістр допускає читання та запис. Значення скидання: 0000 0000.
Формат регістра ATRH подано на рис.4.6.

Автоперезаватажувальний регістр ATRL (*AUTORELOAD REGISTER LOW)*
Регістр допускає читання та запис. Значення скидання: 0000 0000.
Формат регістра ATRL подано на рис.4.7.

15 8

0	0	0	0	ATR 11	ATR 10	ATR 9	ATR 8

Рисунок 4.6. Формат регістра ATRH

7 0

ATR 7	ATR 6	ATR 5	ATR 4	ATR 3	ATR 2	ATR 1	ATR 0

Рисунок 4.7 . Формат регістра ATRL

Біти ATR [11:0] (*Autoreload Register)* – значення автоперезаватажувального регістру. Значення ATR автоматично завантажується в інкрементуючий лічильник CNTR при його переповненні OVF. В режимі ШІМ значення регістра ATR використовується для завдання частоти вихідного сигналу з ШІМ (див. (4.1)).

Регістр керування виводом PWM PWMCR (PWM OUTPUT CONTROL REGISTER)
Регістр допускає читання та запис. Значення скидання: 0000 0000 .
Формат регістра PWMCR подано на рис.4.8.

7 0

0	OE3	0	OE2	0	OE1	0	OE0

Рисунок 4.8 Формат регістра PWMCR

Біти OE [3:0] (*PWMx output enable)* – дозвіл виводу PWMx - встановлюються і скидаються програмно, а так же скидаються апаратно після сигналу RESET. Якщо біт дорівнює 0, то вивід PWM заборонено, якщо – 1, то дозволено.

PWMx регістр керування/стану PWMx PWMxCSR (CONTROL STATUS REGISTER)

Регістр допускає читання та запис. Значення скидання: 0000 0000.

Формат регістра **PWMxCSR** подано на рис.4.9.

7							0
0	0	0	0	0	0	OPx	CMPFx

Рисунок 4.9 Формат регістра PWMxCSR

Біт OPx PWMx (*PWMx Output Polarity)* визначає полярність вихідного сигналу.

Біт **CMPFx** (PWMx *Compare Flag)*. Цей біт встановлюється апаратно і скидається програмно при читанні PWMxCSR регістра. Він дорівнює 1, якщо що значення інкрементуючого лічильника співпадає із значенням в регістрі DCRx.

Регістр керування зупинкою BREAKCR (BREAK CONTROL REGISTER)

Регістр допускає читання та запис. Значення скидання: 0000 0000.

Формат регістра **BREAKCR** подано на рис.4.10.

7							0
0	0	BA	BPEN	PWM3	PWM2	PWM1	PWM0

Рисунок 4.10. Формат регістра **BREAKCR**

На рис. 4.10 позначено:

Біт BA *(Break Active*) - зупинка дозволена. Цей біт встановлюється/скидається програмно, скидається апаратно після сигналу RESET і встановлюється апаратно, коли на виводі BREAK присутній нульовий рівень сигналу. Дію біта BA див. також на рис. 4.1.

Біт BPEN (*Break Pin Enable)* – дозвіл виводу BREAK. Цей біт встановлюється/скидається програмно і скидається апаратно після сигналу RESET. Якщо біт дорівнює 0, то вивід BREAK заборонено, якщо – 1, то дозволено.

Біти PWM [3:0] (*Break Pattern*) – BREAK шаблон. Ці біти встановлюються/скидаються програмно і скидаються апаратно після сигналу RESET. Вони використовуються для переводу 4-х PWMx вихідних сигналів в стабільний стан, коли функція BREAK активна.

Старший PWMx регістр робочого циклу PWMx DCRxH (DUTY CYCLE REGISTER HIGH)

Регістр допускає читання та запис. Значення скидання: 0000 0000.

Формат регістра DCRxH подано на рис.4.11.

15							8
0	0	0	0	DCR11	DCR10	DCR9	DCR8

Рисунок 4.11 Формат регістра DCRxH

Молодший PWMx регістр робочого циклу PWMx DCRxL (DUTY CYCLE REGISTER LOW)

Регістр допускає читання та запис. Значення скидання: 0000 0000.

Формат регістра DCRxL подано на рис.4.12

7							0
DCR7	DCR6	DCR5	DCR4	DCR3	DCR2	DCR1	DCR0

Рисунок 4.12. Формат регістра DCRxL

Біти DCR [11:0] *PWMx Duty Cycle Value)* задають значення робочого циклу. У PWM режимі (OEx = 1 в регістрі PWMCR) DCR [11:0] біти визначають скважність вихідного сигналу ШІМ. У режимі порівняння, вони визначають значення, яке порівнюватиметься з 12-бітним значенням інкрементуючого лічильника.

Формування імпульсів з ШІМ пояснюється рис. 4.13.

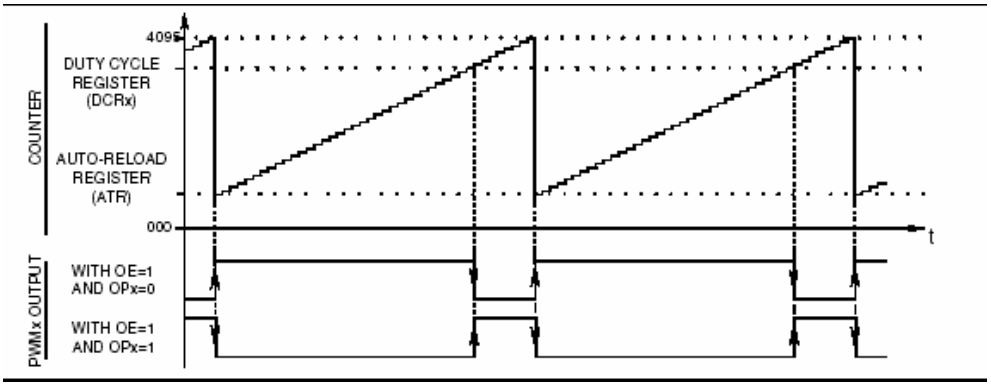


Рисунок 4.13. Формування ШІМ сигналів

Старший регістр вхідного захоплення ATICRH (INPUT CAPTURE REGISTER HIGH)

Регістр допускає тільки читання. Значення скидання: 0000 0000.

Формат регістра ATICRH подано на рис.4.14

15							8
	0	0	0	ICR11	ICR10	ICR9	ICR8

Рисунок 4.14 Формат регістра ATICRH

Молодший регістр вхідного захоплення ATICRL (INPUT CAPTURE REGISTER LOW)

Регістр допускає тільки читання. Значення скидання: 0000 0000.

Формат регістра ATICRL подано на рис.4.15.

7							0
ICR7	ICR6	ICR5	ICR4	ICR3	ICR2	ICR1	ICR0

Рисунок 4.15 Формат регістра ATICRL

Біти ICR [11:0] (*Input Capture Data)* - дані вхідного захоплення. Регістр ATICR містить захоплене значення 12-бітного CNTR регістра, тобто значення на момент появи переднього або заднього фронтів на виводі ATIC. Захоплення може бути виконано тільки тоді, коли прапор ICF скинутий.

Регістр керування передачею TRANCR (TRANSFER CONTROL REGISTER)

Регістр допускає читання та запис. Значення скидання: 0000 0001.

Формат регістра TRANCR подано на рис.4.16.

7							0
0	0	0	0	0	0	0	TRAN

Рисунок 4.16 Формат регістра TRANCR

Біт TRAN *(Transfer enable)* - дозвіл перезапису DCRx у тіньовий регістр після переповнення. Цей біт встановлюється/скидається програмно, скидається апаратно після кожної закінченої передачі і встановлюється апаратно після скидання RESET.

Приклад виконання лабораторного завдання

Завдання:

За допомогою ШІМ керувати швидкістю обертання ротора двигуна. Початкова швидкість дорівнює 0, швидкість збільшувати рівними кроками після кожного натискання кнопки S4.
Після кожного збільшення швидкості вводити затримку в 0,5 сек.

Текст програми:

```
;*******************************************************************
;
;  МІСЦЕ ДЛЯ ОГОЛОШЕННЯ ЗМІННИХ
;
;*******************************************************************

var   ds.b 1

;*******************************************************************
;
; ІНІЦІАЛІЗАЦІЯ ПОРТІВ ST7
;
;*******************************************************************
init_ST7:
      clr  MCCSR              ; нормальний режим
      ret

init_IO:
      ld   A,#%00001100       ; Настроювання регістру PADDR
                              ; (вивід PWM0 – вивід PA2)
      ld   PADDR,A

      ld   A,#%00001100       ; PA2 та PA3 в режимі виведення push-pull
      ld   PAOR,A

      ret

;*******************************************************************
;
;  PWM
;
;*******************************************************************
pwm_init:
      bset ATCSR,#0           ; активуємо CMPF переривання для виводу з
                              ; порівнянням (output-compare)
      clr  PWMCR              ; скидання регістру Output PWM
      bset PWMCR, #0          ; вибір вихідного сигналу PWM0
      ret

pwm:
      ld   A,#0
      ld   DCR0L,A            ; вміст регістру A записуємо в DCR0L
                              ; (нижній регістр таймеру каналу PWM0)
      ld   A,var              ; вміст змінної „var” записуємо в DCR0H
                              ; (верхній регістр таймеру каналу PWM0)
      srl  A                  ; Зсуваємо вліво 4 рази
      srl  A
```

```
      srl  A
      srl  A
      ld   DCR0H,A          ; Шпаруватість керується чотирма лівими бітами
                            ; регістру DCR0H (від 0 до $FF)

      ;bset ATCSR,#3        ; fCounterClock = fLTIMER (%00000011) 1ms@8MHz
      bset ATCSR,#4         ; fCounterClock = fCPU (8MHz)

      ld   A,#1             ; Завантажуємо число „1” до регістру A
      ld   TRANCR,A         ; вмикаємо автозаватаження
                            ; Інвертуємо полярність PWM0
      ld   A,#%00000001     ; шляхом встановлення біту 0 регістру TRANCR в 1
      ld   PWM0CSR,A        ; визиваємо переривання CMP для порівняння

      ret


; Підпрограма затримки
delay:
                            ; 256*(256*(3+4)+3+3+2)+3+4+2+5+4+6 ≈ 0,5 сек
      push x
      push y
      ld   x,#$ff
dec_2:
      ld   y,#$ff
dec_1:
      dec  y
      JRNE dec_1
      dec  x
      JRNE dec_2
      pop  y
      pop  x
      ret

;*******************************************************************
;
; ГОЛОВНА ПРОГРАМА ST7
;
;*******************************************************************
main:
      RSP                   ; Скидаємо покажчик стеку
      sim                   ; Маскуємо переривання
      call init_ST7         ; Ініціалізація ST7
      call init_IO          ; Ініціалізація портів
      call pwm_init         ; Ініціалізація таймеру, настройка PWM0 (ШІМ)

      ld   A, #%00000000    ; Задаємо стартову шпаруватість

start:
      ld   var, A

      call pwm              ; Налаштовуємо PWM0 на нову шпаруватість

      btjt PADR,#3, start   ; Якщо натиснута клавіша S4 не стрибати на start
      ld   A, var
      add  A, #%00001111    ; Збільшуємо шпаруватість на 0F
      ld   var, A           ; Зберігаємо нову шпаруватість
      call delay            ; Вводимо затримку на 0,5 сек

      JP  start             ; зациклюємо програму
      JP   main
```

Контрольні запитання

1. Назвіть призначення та склад ART таймеру
2. Як залучити до роботи сторожовий таймер МК ST7?
3. Поясніть функцію останову ШІМ сигналів.
4. Поясніть принцип формування ШІМ сигналів
5. Назвіть призначення та склад LITE таймеру.

Лабораторна робота №5

АЦП мікроконтролерів ST7

Завдання:

Ініціалізувати АЦП. Забезпечити вимірювання напруги на потенціометрі R20, що підключено до лінії порту PB0. Використовувати виміряне відносне значення напруги у спосіб, вказаний варіанті.

Порядок роботи з макетом:

- Увімкнути всі, крім 3-го, перемикачі в блоці перемикачів B1/S4.
- Вимкнути джампери W4 та W7.
- Увімкнути джампери W3, W5 та W4.

Таблиця 5.1. - Способи використання виміряного значення напруги:

Режим	**Опис**
1.	Виміряне значення напруги пропорційно перерахувати в ціле число від 1 до 8. Засвічувати світлодіод, номер якого відповідає розрахованому числу.
2.	Виміряне значення напруги пропорційно перерахувати в ціле число від 1 до 8. Засвічувати всі світлодіоди від першого до світлодіода, номер якого відповідає розрахованому числу включно.
3.	Забезпечити блимання одного з розрядів дисплею. Кнопкою *S5* переміщувати блимання на сусідній розряд праворуч та по колу. Виміряне значення напруги пропорційно перерахувати в ціле число від 0 до 9. Вивести розраховане число в розряд, що блимає.
4.	Керувати швидкістю обертання електричного двигуна. Задавати швидкість обертання на основі виміряного значення напруги.
5.	Виміряне значення напруги пропорційно перерахувати в число від 0.0 до 5.0 з точністю до десятих. Вивести розраховане число на світлодіодний дисплей.
6.	Вивести на світлодіодний дисплей довільне число. Після натискання кнопки *S5* почати на одиницю збільшувати число на дисплеї зі швидкістю, що залежить від виміряного значення напруги. Після повторного натискання кнопки *B* зупинити збільшення числа.
7.	Вивести на світлодіодний дисплей довільне число. Після натискання кнопки *S5* почати на одиницю змінювати число на дисплеї зі швидкістю, що залежить від виміряного значення напруги. Якщо виміряне значення напруги менше від половини максимально можливого значення, то число зменшувати, а якщо більше — збільшувати. Після повторного натискання кнопки *B* зупинити збільшення числа.
8.	Організувати смугу прокрутки послідовності 16-значних цифр. Напрямок прокрутки змінювати кнопкою *S5*. Швидкість прокрутки залежить від виміряного значення напруги.

Таблиця 5.1. Завдання для лабораторної роботи 5

№ варіанту	Спосіб керування
1.	1, 3
2.	2, 4
3.	1, 5
4.	2, 6
5.	1, 7
6.	2, 8
7.	1, 3
8.	2, 4
9.	1, 5
10.	2, 6

Теоретичні відомості

Аналого-цифровий перетворювач являє собою семиканальний 10-розрядний АЦП послідовного наближення. Структурна схема АЦП наведена на рис.5.1.

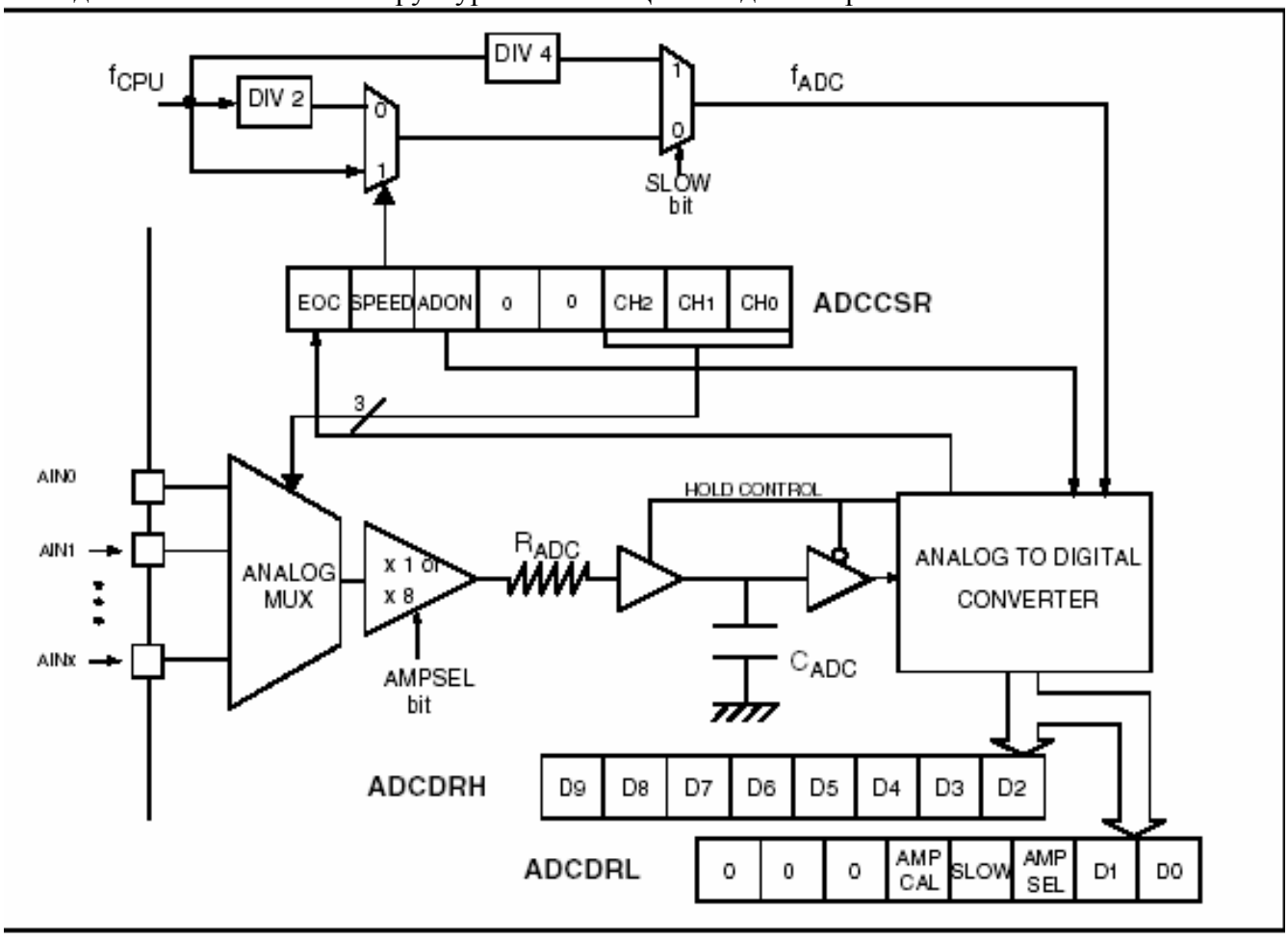


Рисунок 5.1 - Структурна схема АЦП

Схема містить: аналогово-цифровий перетворювач, аналоговий мультиплексор, вхідний підсилювач з регульованим коефіцієнтом підсилення (x1 або x8) та регістри даних ADCDRH, ADCDRL та керування/статусу ADCCSR.
Регістр ADCDRH зберігає старші біти результату аналогового перетворення D[9:2], регістр ADCDRL - два молодших біта D[1:0] та крім того, наступні керуючі біти:

AMPCAL Amplifier Calibration Bit – калібровка підсилювача. При 1 режим калібровки, при цьому вхідна напруга підсилювача встановлюється на нульовому рівні.
SLOW Slow mode – повільний режим.
AMPSEL Amplifier Selection Bit – біт виборки підсилювача (вибору коефіцієнту підсилення 1 або 8, див. рис. 4.46). При AMPSEL=1 діапазон вхідної напруги становить від 0 до Ucc/8, тобто при Ucc=5 В, від 0 до 430 мВ. При цьому роздільна здатність дорівнює 0,6 мВ (еквівалент 13-го розряду). При AMPSEL=0 діапазон вхідної напруги становить від 0 до Ucc.
Біти SLOW і біт SPEED регістру ADCCSR задають частоту роботи АЦП згідно табл.5.3.

Таблиця 5.3 – Вибор частоти АЦП перетворення

f_{ADC}	SLOW	SPEED
$f_{CPU}/2$	0	0
f_{CPU}	0	1
$f_{CPU}/4$	1	x

Регістр керування/статусу ADCCSR (див. рис. 3.1) містить наступні біти:
Біт *EOC (End of Conversion)* – встановлюється в 1 по закінченню аналогово-цифрового перетворення
Біт *SPEED (ADC clock selection)* – цей біт сумісно з бітом SLOW задають частоту роботи АЦП згідно табл.5.3.
Біт *ADON (A/D Converter on)* – запуск АЦП
Біти *CH[2:0] Channel Selection* – задають канал АЦП згідно з табл. 5.4.

Таблиця 5.4– Вибір каналу АЦП

Вивід* МК	CH2	CH1	CH0
AIN0	0	0	0
AIN1	0	0	1
AIN2	0	1	0
AIN3	0	1	1
AIN4	1	0	0
AIN5	1	0	1
AIN6	1	1	0

Приклад 5.1. Написати програму для ініціалізації АЦП для введення аналогового сигналу по лінії PB0
Для цього необхідно лінію порта PB0 запрограмувати для введення у високоімпеденсний стан (00) та обрати канал АЦП (000):

```
 Init_ADC:
                call    init_portB          ;ініціалізація порту B
                ld      A,#%00000000        ; Вибір AIN0 (PB0) скиданням CH0=CH1=CH2
                ld      ADCCSR,A
                ret
```

Приклад 5.2 Введення даних з АЦП (8-розрядного)

```
Read_ADC:
                bset    ADCCSR,#5           ; Запуск АЦП
 cont:                                      ;Очікування закінчення перетворення
                ld      A,ADCCSR
                and     A,#$80
                jreq    cont
                bres    ADCCSR,#5           ; Зупинка АЦП встановленням біта 5 (ADON)
                                            регістра ADCCSR
```

```
        ld      A,ADCDRH        ; Зчитування молодшого байту результату АЦП
        ld      var,A           ; запис у комірку var
        ret
```

Приклад виконання лабораторного завдання

Завдання:
Вимірювати напругу на потенціометрі R20. На основі виміряного значення напруги змінювати яскравість світлодіода №4. Керувати яскравістю за допомогою ШІМ.

Текст програми:

```
;*******************************************************************
;
;  МІСЦЕ ДЛЯ ОГОЛОШЕННЯ ЗМІННИХ
;
;*******************************************************************

var   ds.b 1

;*******************************************************************
;
; ІНІЦІАЛІЗАЦІЯ ПОРТІВ ST7
;
;*******************************************************************
init_ST7:
      clr   MCCSR             ; нормальний режим
      ret

init_IO:
      ld    A,#%00010000      ; Настроювання регістру PADDR
                              ; (вивід PWM2 – вивід PA4, світлодіод №4)
      ld    PADDR,A

      ld    A,#%00010000      ; PA4 в режимі виведення push-pull
      ld    PAOR,A
      ld    PBDDR,A           ; PB0 в режимі floating input
      ld    PBOR,A            ;

      ret

;*******************************************************************
;
;  PWM
;
;*******************************************************************
pwm_init:
      bset  ATCSR,#0          ; активуємо CMPF переривання для виводу з
                              ; порівнянням (output-compare)
      clr   PWMCR             ; скидання регістру Output PWM
      bset  PWMCR, #4         ; вибір вихідного сигналу PWM2
      ret

pwm:
      ld    A,#0
      ld    DCR2L,A           ; вміст регістру A записуємо в DCR2L
                              ; (нижній регістр таймеру каналу PWM2)
      ld    A,var             ; вміст змінної „var” записуємо в DCR2H
                              ; (верхній регістр таймеру каналу PWM2)
      srl   A                 ; Зсуваємо вліво 4 рази
      srl   A
      srl   A
      srl   A
      ld    DCR2H,A           ; Шпаруватість керується чотирма лівими бітами
                              ; регістру DCR2H (від 0 до $FF)

      ;bsetATCSR,#3           ; fCounterClock = fLTIMER (%00000011) 1ms@8MHz
      bset  ATCSR,#4          ; fCounterClock = fCPU (8MHz)

      ld    A,#1              ; Завантажуємо число „1” до регістру A
```

```
      ld   TRANCR,A          ; інвертуємо полярність PWM2
                             ; шляхом встановлення біту 0 регістру TRANCR в 1
      ld   A,#%00000001      ; визиваємо переривання CMP для порівняння
      ld    PWM2CSR,A

      ret

;*********************************************************************
;
;  ПІДПРОГРАМИ РОБОТИ З АЦП
;
;*********************************************************************
; - Підпрограма ініціалізації АЦП та налаштування регістру ADCSR
select_CH:
      ld   A,#%00000000      ; Вибір AIN0 (PB0) скиданням CH0=CH1=CH2
; Цей канал обрано тому, що до нього (PB0) під'єднано потенціометр
; Вивід PB0 повинен бути налаштований на високоімпедансний стан
; (floating input)
      ld   ADCCSR,A          ; завантажуємо %00000000 в регістр
      ret

; - Запуск процесу вимірювання аналогового сигналу з потенціометра
process_adc:
      bset ADCCSR,#5         ; Один раз запустити
                             ; аналогово-цифрове перетворення
cont:
      ld   A,ADCCSR          ; завантажити вміст ADCCSR до регістру "A"
      and  A,#$80            ; оператор "And" дозволяє перевірити:
                             ; Чи зацінчився процес перетворення?
      jreq cont              ; Якщо ні, то це означає, що біт «OEC» ще
                             ; не дорівнює 1
; Біт №7 регістру ADCCSR має назву "OEC"
      bres ADCCSR,#5         ; Зупинити АЦП шляхом встановлення біту №5
                             ; регістру ADCCSR в 1
; Біт №5 регістру ADCCSR має назву „ADON"
; Починаємо читати результат аналогово-цифрового перетворення
      ld   A,ADCDRH          ; Завантажуємо вміст старшого (верхнього)
                             ; регістру АЦП до регістру A
; "var" – це заздалегідь оголошена змінна, що зберігається в RAM0
      ld   var,A             ; змінна „var" тепер містить результат
                             ; аналогово-цифрового перетворення
      ret

;*********************************************************************
;
; ГОЛОВНА ПРОГРАМА ST7
;
;*********************************************************************
main:
      RSP                    ; Скидаємо покажчик стеку
      sim                    ; Маскуємо переривання
      call init_ST7          ; Ініціалізація ST7
      call init_IO           ; Ініціалізація портів
      call pwm_init          ; Ініціалізація таймеру, настройка PWM2 (ШІМ)
      call select_CH         ; Обираємо канал АЦП №0

start:
      call process_adc       ; Вимірюємо напругу на потенціометрі R20

      call pwm               ; Налаштовуємо PWM2 на нову шпаруватість

      JP  start              ; зациклюємо програму
      JP   main
```

Контрольні запитання

1. Дайте характеристику АЦП МК ST7.
2. Який принцип перетворення реалізовано в АЦП МК ST7
3. Скільки входів має АЦП? Як вони обираються?
4. Поясніть призначення підсилювача з регульованим коефіцієнтом підсилення?
5. Як визначається роздільна здатність АЦП у різних режимах підсилення?

Додаток А

Відлагодження програм у середовище ST7 Visual Develop

Середовище ST7 Visual Develop використовується для віртуальної відладки програм для мікроконтролерів ST7. Відладка відбувається з використанням спеціального емулятора мікро контролера.

Необхідні кроки підготовки до роботи:

1. Створити структури каталогів для зберігання файлів проектів.
2. Встановити програмне забезпечення STVD7.
3. Під'єднати та перевірити інтерфейс inDART-STX.
4. Запустити середовище STVD7 та створити новий робочий простір.
5. Створити новий проект.
6. Додати файли до проекту.

Детально опишемо кожен із кроків.

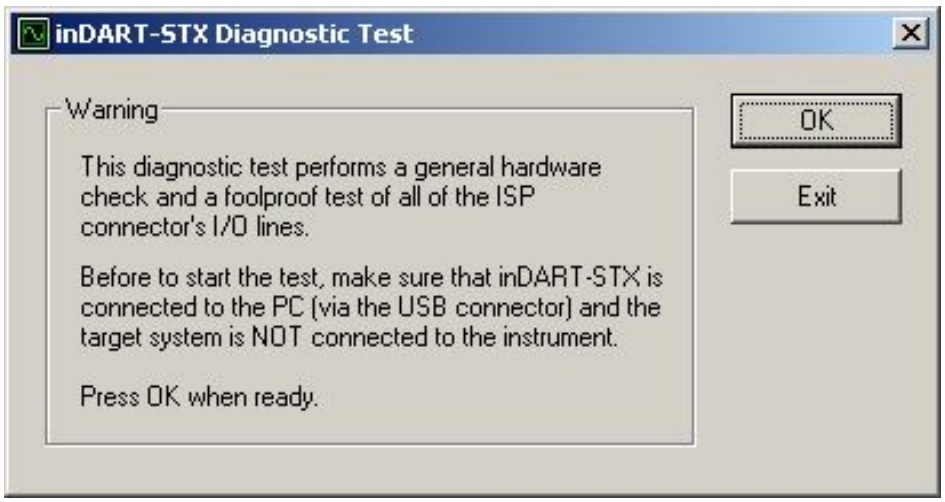
Створення структури каталогів
Рекомендується зберігати файли лабораторних робіт таким чином, щоби файли кожної з робіт знаходилися в окремих каталогах. Приклад наведено нижче:

C:\MCU_ST7Lite2	**LabWork_1** **LabWork_2** **…** **LabWork_N**

Встановлення програмного забезпечення
Перед початком робіт необхідно встановіти програмне забезпечення:

- Першим потрібно встановити пакет: "st7_toolset.exe"
- Другим потрібно встановити пакет: "st7 2006.exe"

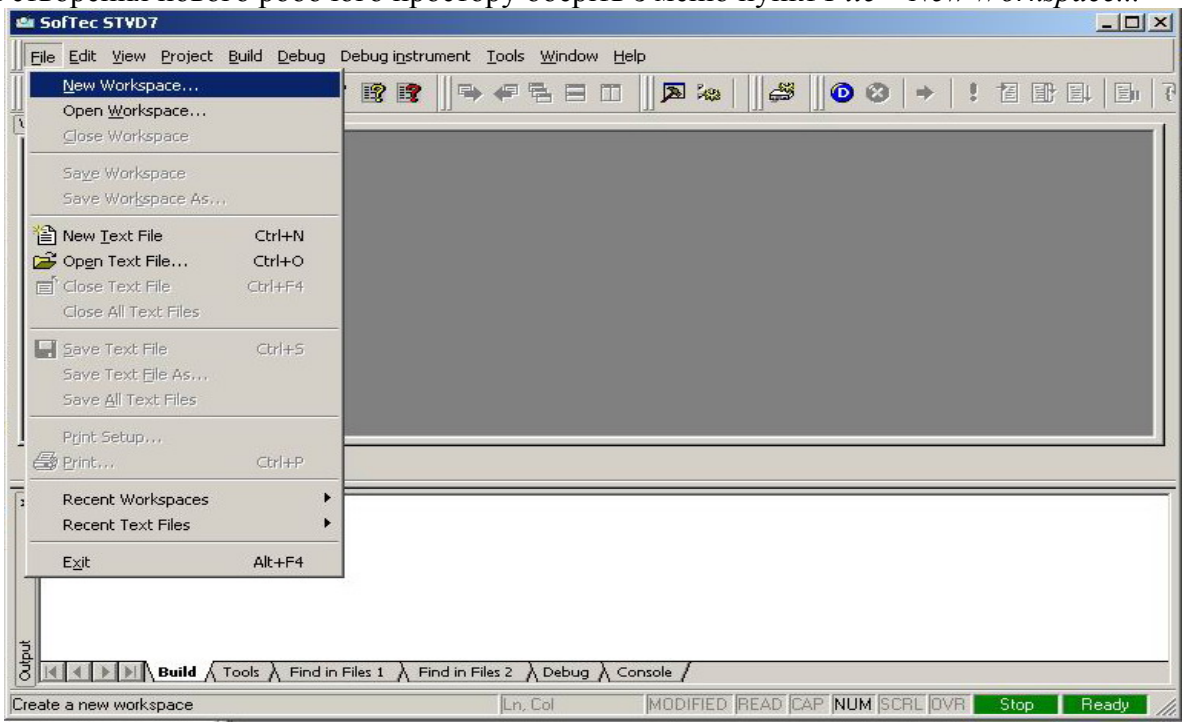
Під'єднання та перевірка inDART-STX
Перед початком робіт необхідно під'єднати та перевірити працездатність inDART-STX за допомогою *inDART-STX Diagnostic Test* (плата ST7 на цьому етапі повинна бути від'єднаною)



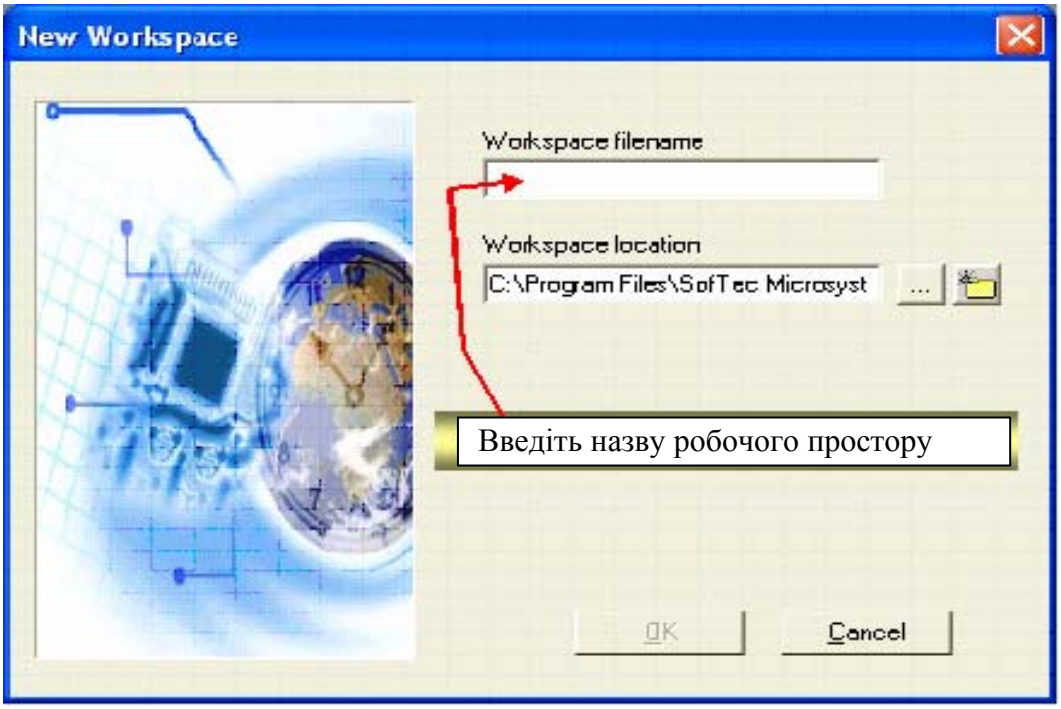
Вікно програми *inDART-STX Diagnostic Test*

Створення нового робочого простору

Для створення нового робочого простору оберіть з меню пункт *File->New Workspace...*

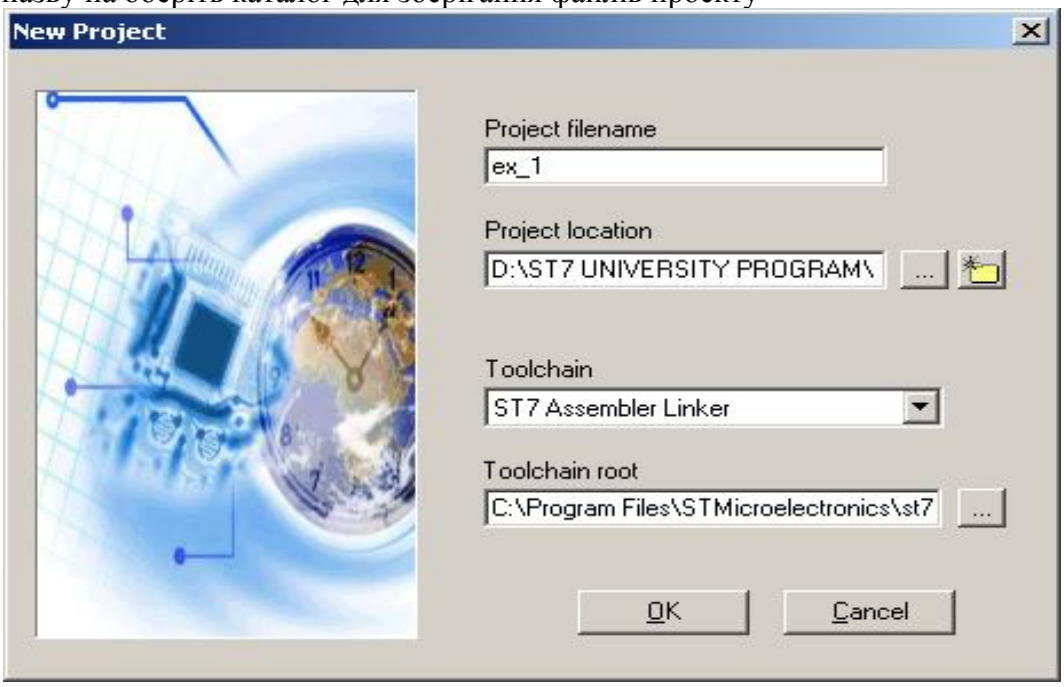


Введіть назву на оберіть каталог для зберігання файлів робочого простору

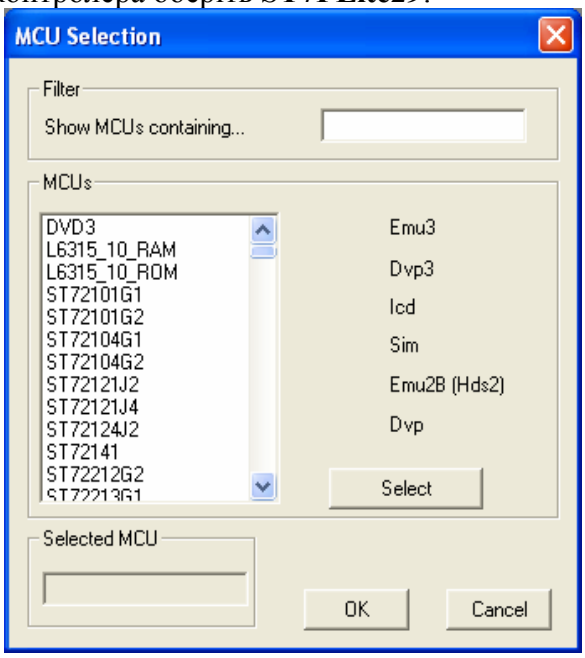


Створення нового проекту

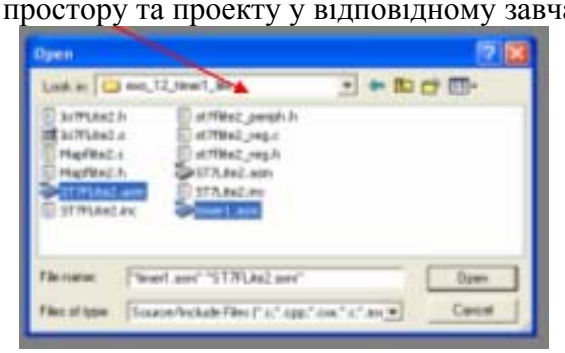
Введіть назву на оберіть каталог для зберігання файлів проекту



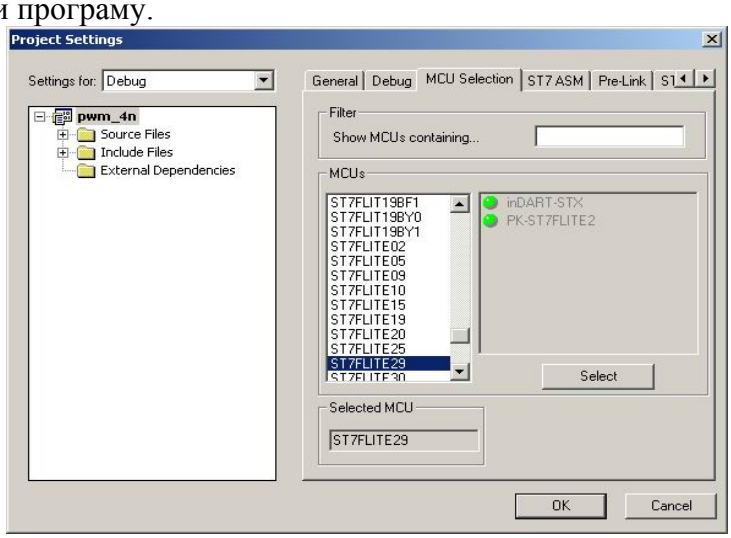
У якості цільового мікроконтролера оберіть **ST7FLite29**:



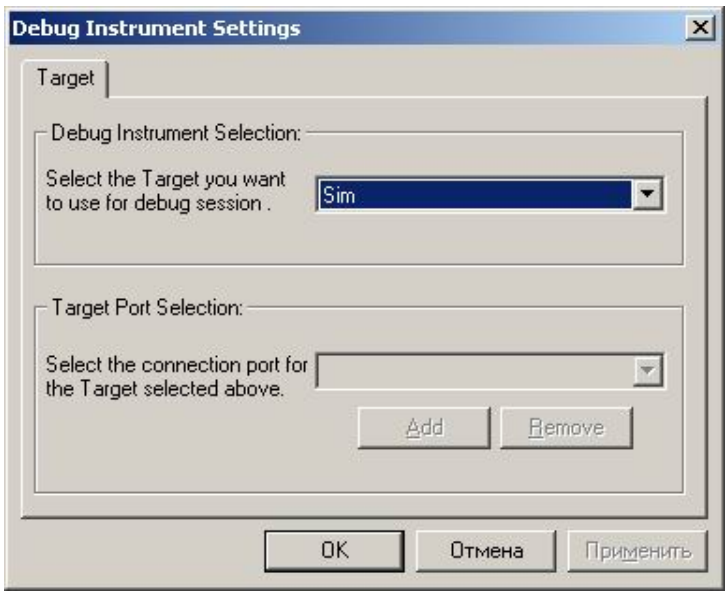
Запишіть файли робочого простору та проекту у відповідному завчасно створеному каталозі.



Оберіть у меню *Project->Settings…* (закладка *MCU Selection*), мікро контролер, для якого планується написати програму.



Для роботи з емулятором мікроконтролера необхідно обрати у якості “Цілі” (*Target*) емулятор (*Sim*) в меню *Debug Instruments -> Target settings…*:



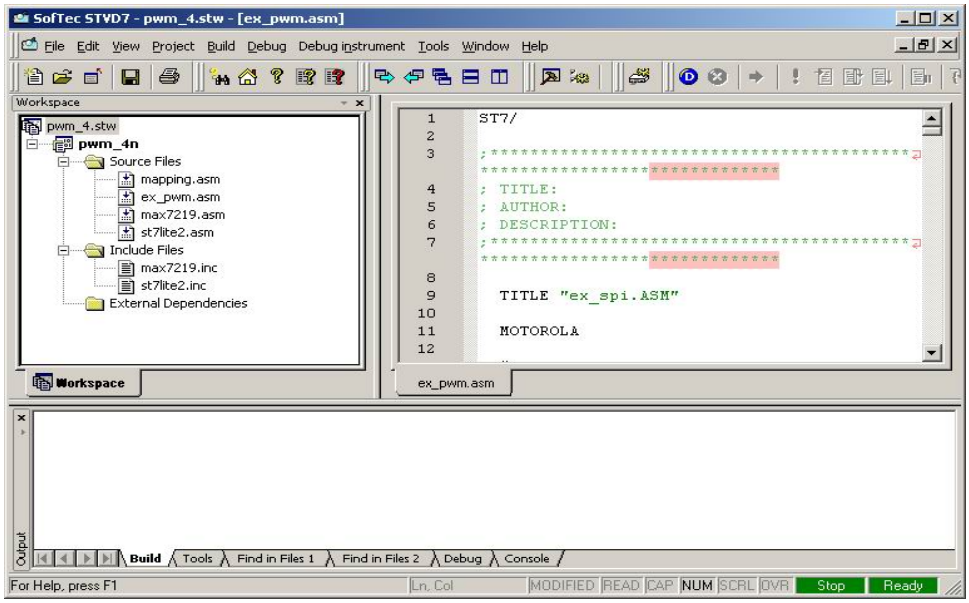
Додавання файлів до проекту

Перед початком розробки необхідно спочатку скопіювати до папки з проектом та додати файли до проекту в STVD7 (можна також використати шаблон робочого простору):

- mapping.asm – копіювати не треба, створюється автоматично
- st7lite.asm
- max7219.asm – для роботи з індикатором по SPI
- template.asm – основний файл проекту, заготовка (змінити ім'я на відповідне лабораторній роботі)

Заголовочні файли:

- st7lite.inc
- max7219.inc

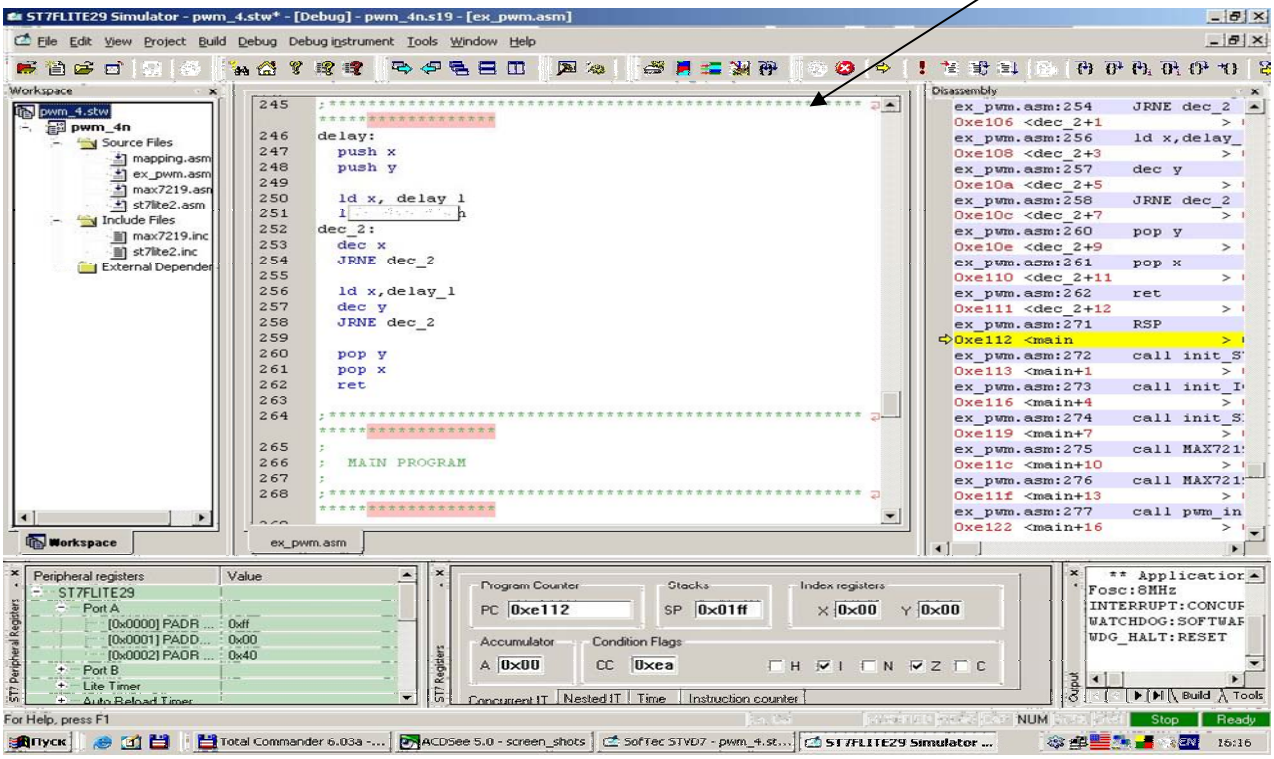


Вікно програми STVD7. Перелік файлів проекту зверху зліва.

Відладка проекту

Для увімкнення відладки потрібно натиснути відповідну кнопку на панелі інструментів (див. рис. нижче).

Включення відладки



Вікно програми STVD7. Режим відладки.

Для виконання наступного кроку програми натискайте кнопку F10 на клавіатурі.

Апаратна відладка програми

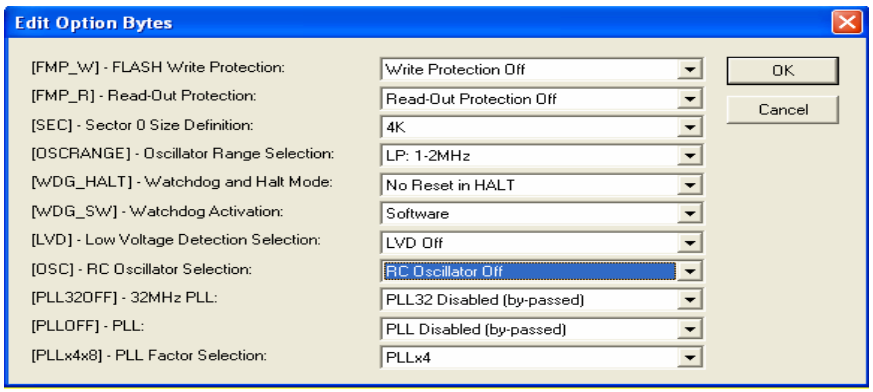
Середовище SofTec STVD7 повністю повторює інтерфейс від ST7 Visual Develop (STVD7)

- Файли робочого простору цих пакетів сумісні.
- Призначення середовища – апаратна відладка програми.

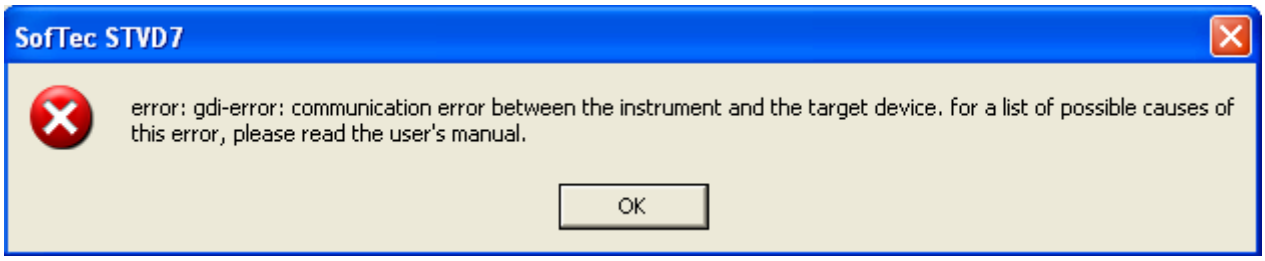
Увага!!! Перед роботою з SofTec STVD7:

В програмній середі STVD7 for InDart-STX натиснути піктограму *Start Debugging*. Потім в меню *Debug Instrument* пункт *MCU Configuration*, далі пункт *Set Option Bytes*.

Для запобігання збоям зв'язку мікроконтролера з відладчиком пункт байта опцій *RC Oscillator Selection* повинен бути встановленим у стан ***RC Oscillator On*** (див. рис. нижче).



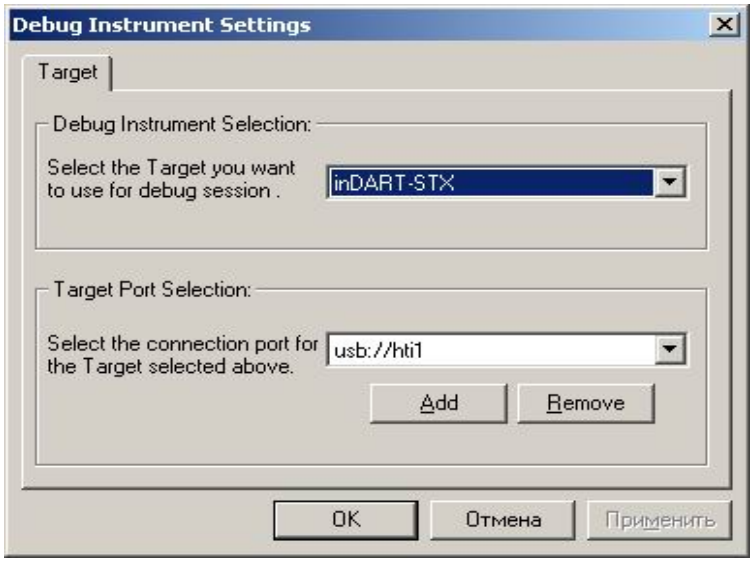
Якщо цього не зробити, то при вході до режиму відладки автоматично програмується невірний байт опцій, який вимикає вмонтований RC генератор. Мікроконтролер втрачає джерело тактових імпульсів і зв'язок з ним стає неможливим (див. рис. нижче):



Для відновлення нормальної роботи необхідно перепрограмувати байт опцій, що вже неможливо без зовнішнього джерела тактових імпульсів.

Вибір inDART-STX

Для роботи з мікроконтролером необхідно обрати у якості "Цілі" (*Target*) інтерфейс *inDART-STX* в меню *Debug Instruments -> Target settings*...(див. рис. нижче):



Тоді:

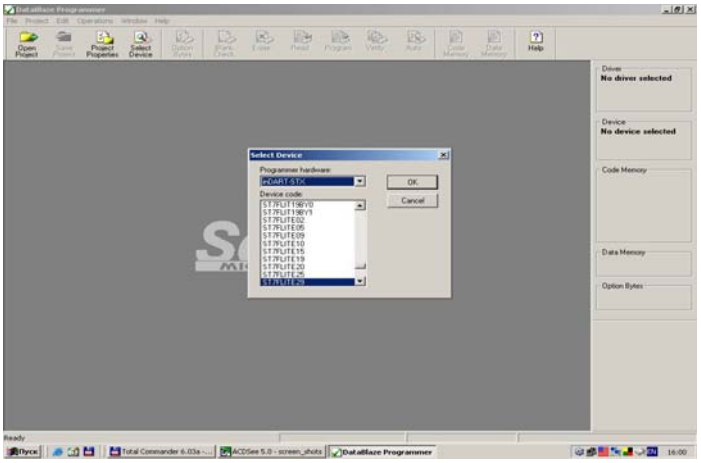
- Для роботи з мікроконтролером буде використовуватися інтерфейс STX
- Перетворення інтерфейсу USB -> STX буде виконуватися за допомогою адаптера *inDART-STX* (*перед роботою з мікроконтролером не забувайте перевіряти байт опцій*)

Відладка у SofTec STVD7
Відладка в середовищі SofTec STVD7 виконується ідентично до відладки у ST7 Visual Develop.

ПЗ DataBlaze Programmer

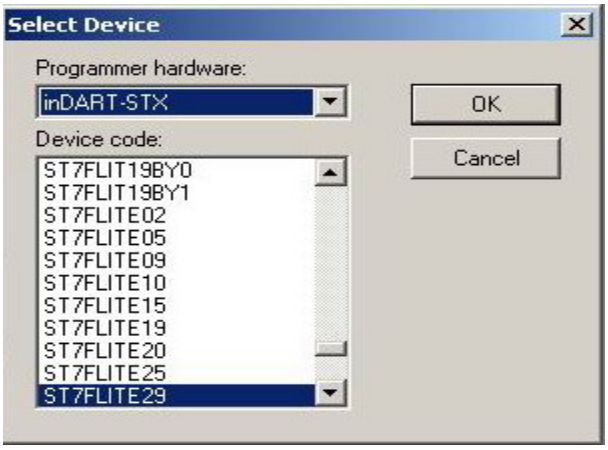
DataBlaze Programmer – це програмне забезпечення для роботи з пам'ятю програм та даних мікроконтролерів ST.
Можливості програми: установка байтів опцій, стирання, перевірка стирання, запис, читання, перевірка запису пам'яті програм та даних.

УВАГА! Перед роботою з DataBlaze Programmer перевірте байт опцій.
Є ризик виводу з ладу мікроконтролера!

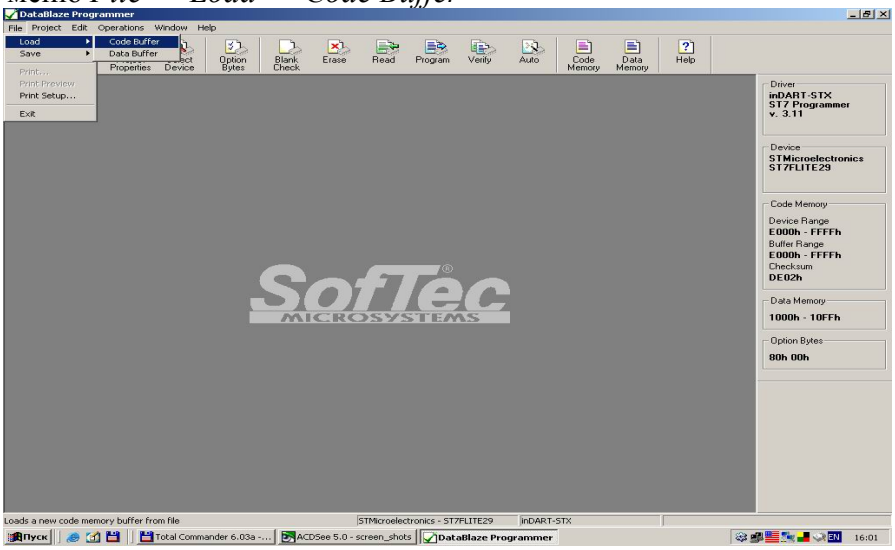


Інтерфейс DataBlaze Programmer

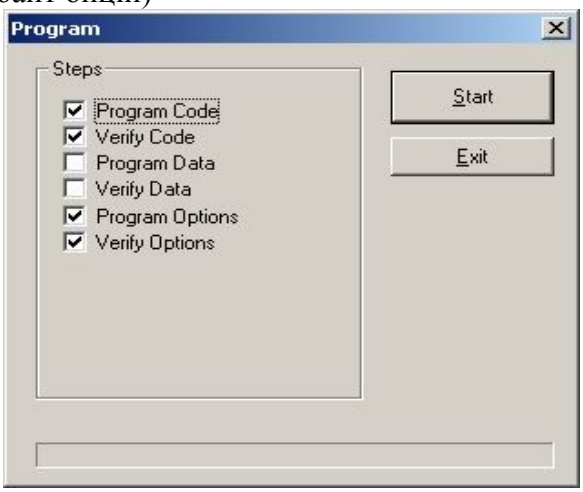
Вибір моделі мікроконтролера
Натисніть кнопку *Select Device*



Завантаження відкомпільованої програми
Оберіть пункт меню *File -> Load -> Code Buffer*



Програмування/запис мікроконтролера
(не забувайте перевіряти байт опцій)



Робота мікроконтролера в автономному режимі

- Для роботи мікроконтролера в автономному режимі необхідно скомпілювати програму у середовищі SofTec STVD7 або ST7 Visual Develop з настройками *Release* замість *Debug* (меню *Build -> Configurations…*).
- Запрограмувати мікроконтролер у DataBlaze Programmer.
- Від’єднати плату inDART-STX від плати мікроконтролера.
- Перезавантажити мікроконтролер.

Додаток Б

Лабораторний стенд ST7/ST5

Конфігурація лабораторного стенду наведено на рис. Б.1

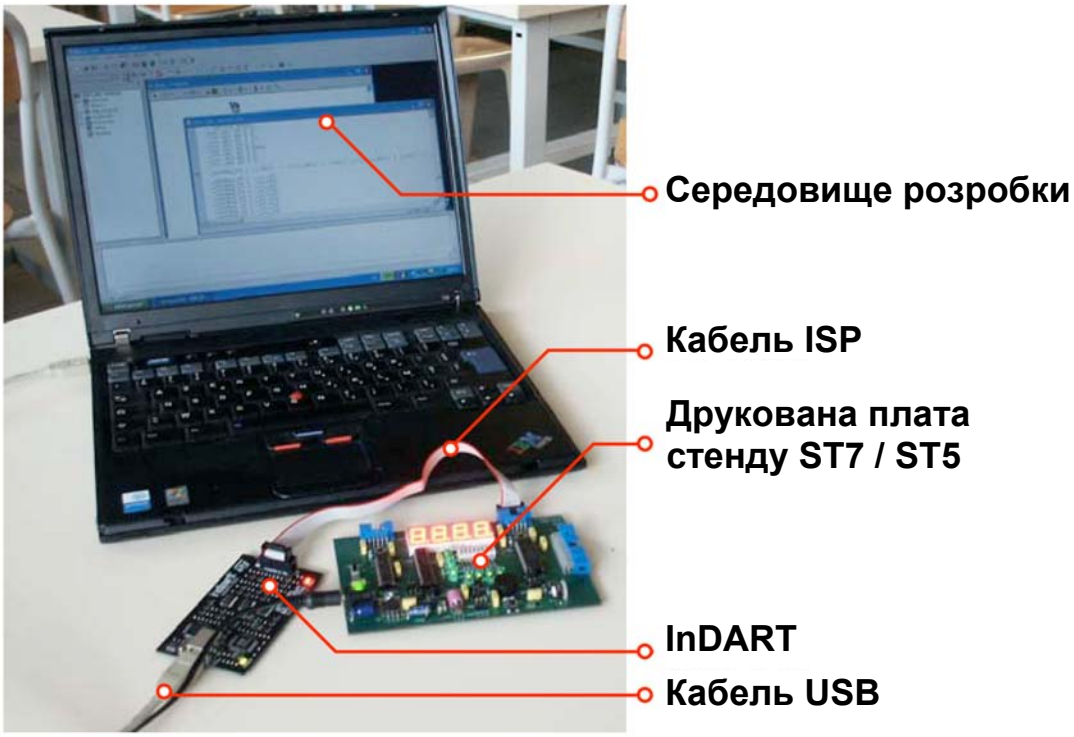


Рис. Б.1.

Структурна блок-схема стенду наведено на рис Б.2

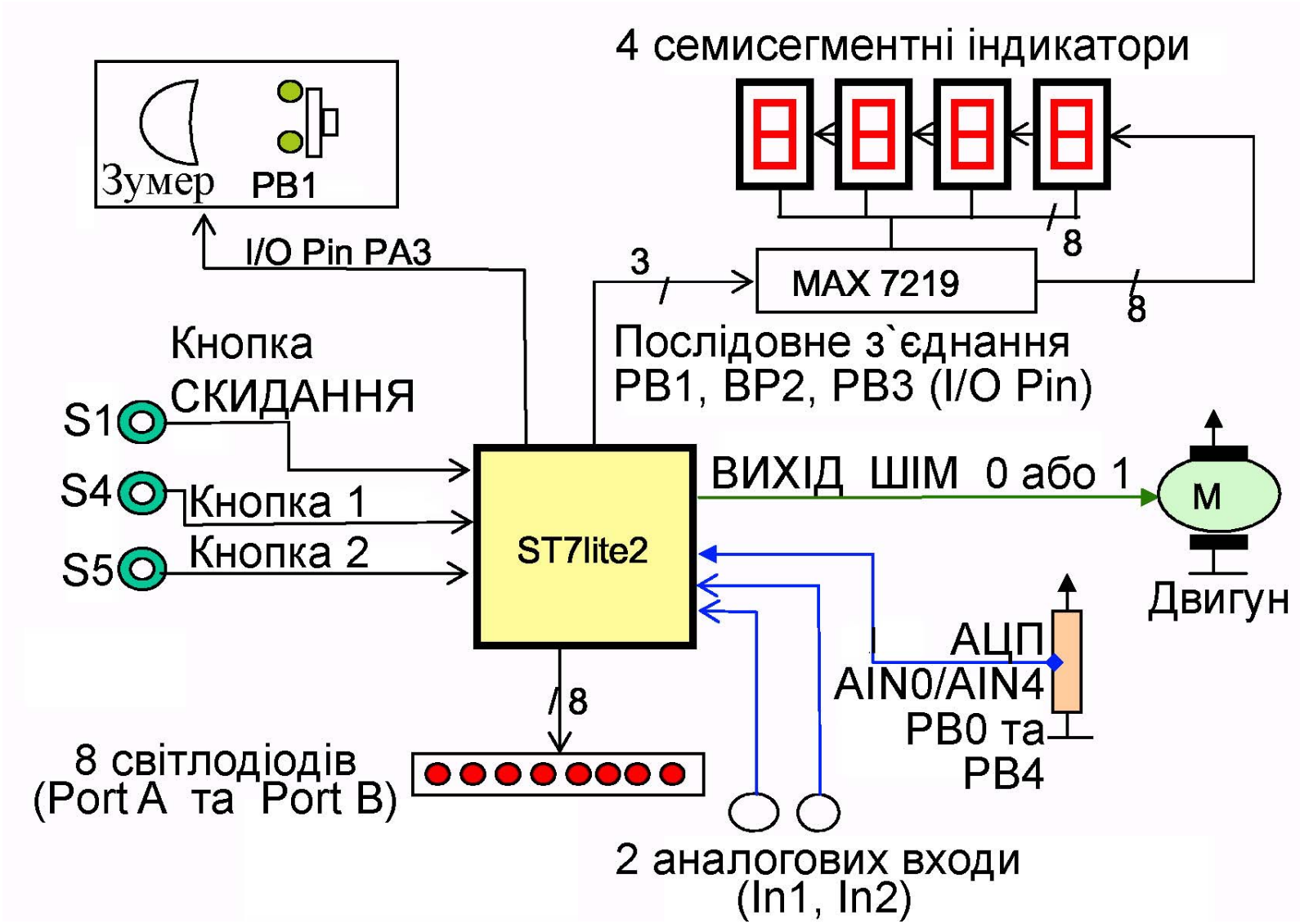


Рис. Б.2.

Друкована плата стенду наведена на рис. Б.3.

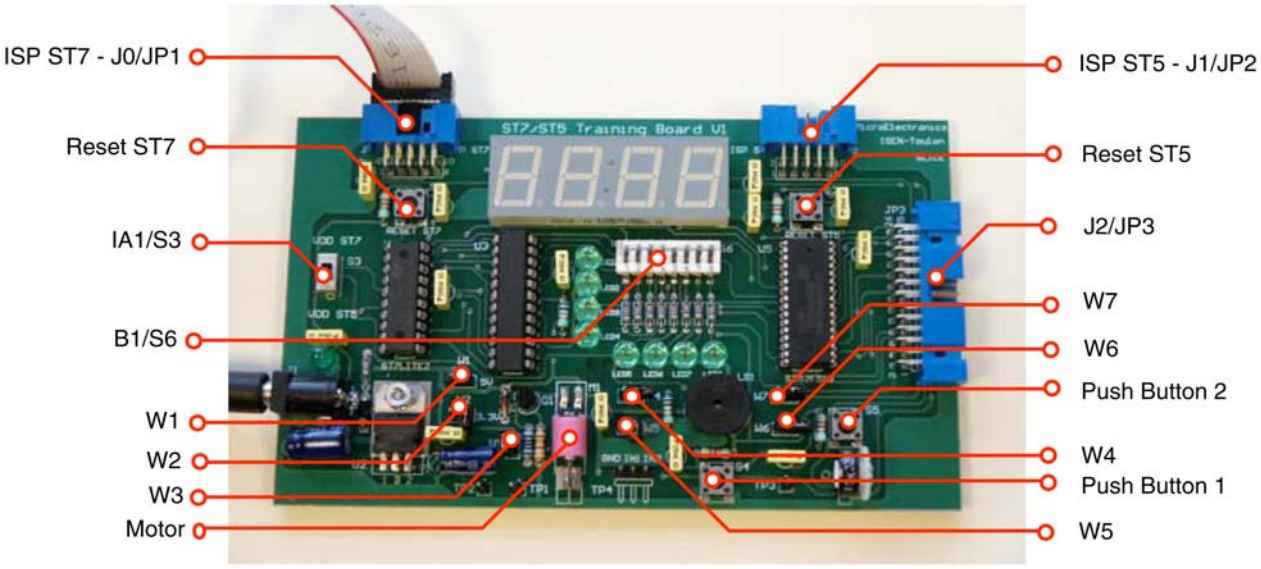


Рис.Б.3

Виводи введення/виведення і з`єднувальні інтерфейси наведено в табл. Б1.

Таблиця Б1. Виводи введення/виведення і з`єднувальні інтерфейси

Назва штиревого виводу (Pin)	Підключений зовнішній пристрій	№ виводу ST7	№ виводу J2/JP3
PA0/LTIC	LED1	13	6
PA1/ATIC	LED2	12	7
PA2/ATPWM0	Двигун	11	4
PA3/ATPWM1	LED3 / Зумер / Кнопка 1	10	13
PA4/ATPWM2	LED4	9	14
PA5/ATPWM3/ICCDATA	Зарезервовано для ISP	8	15
PA6/MCO/ICCCLK/BREAK	Зарезервовано для ISP	7	16
PA7	LED5	6	17
PB0/SS/AIN0	Потенціометр / Кнопка 2 / In1	19	5
PB1/SCK/AIN1	Відображає сигнал управління SPI : CKSPI	20	3
PB2/MISO/AIN2	Відображає сигнал управління: завантаження даних у Max7219	1	1
PB3/MOSI/AIN3	Відображає сигнал управління SPI : MOSI	2	2
PB4/CLKIN/AIN4	LED6 / In2	3	10
PB5/AIN5	LED7	4	11
PB6/AIN6	LED8	5	12
Reset	Кнопка СКИДАННЯ	18	

На рис. Б.4. наведено принципову схему стенда

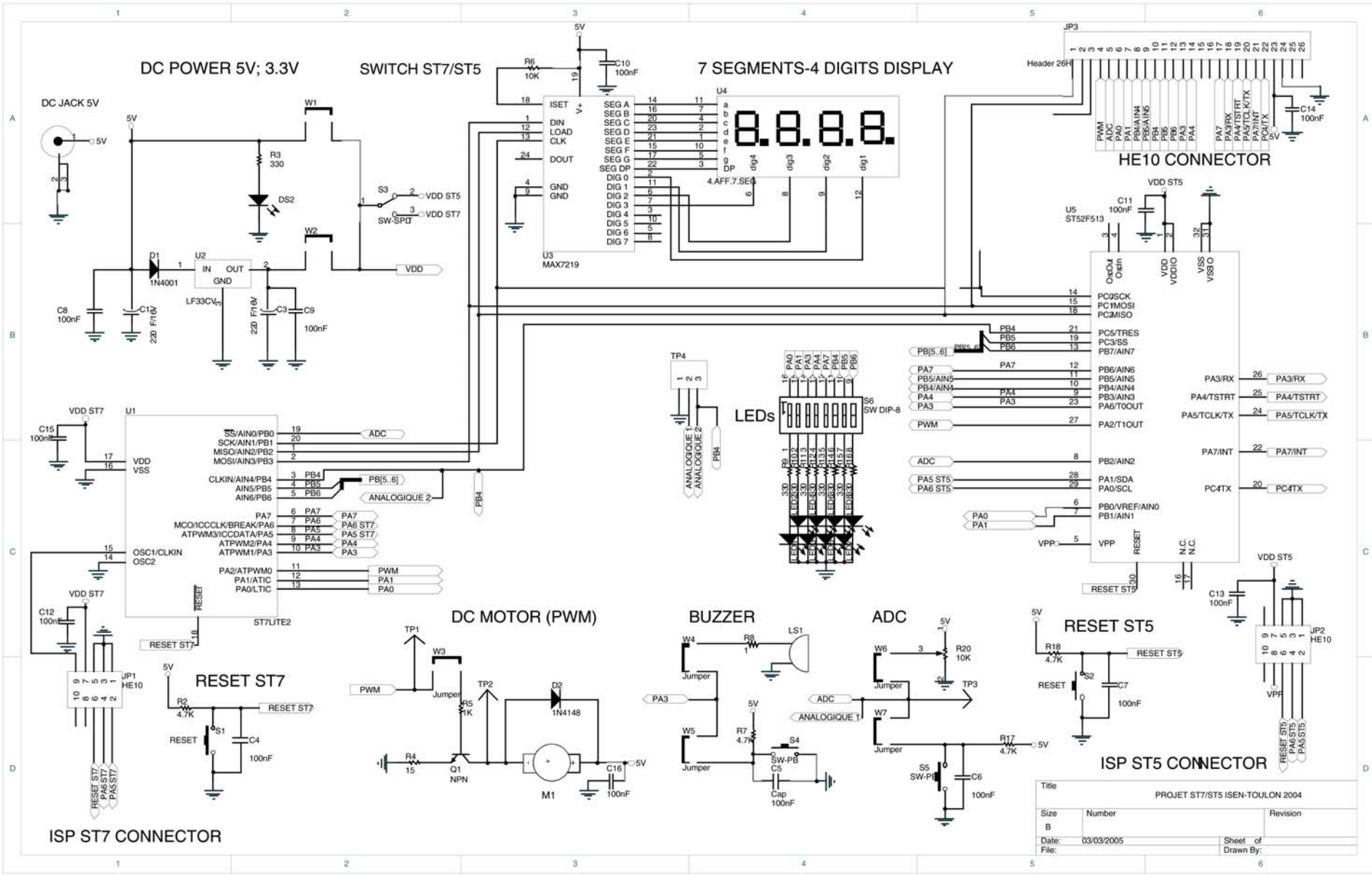

DC POWER 5V; 3.3V
SWITCH ST7/ST5
7 SEGMENTS-4 DIGITS DISPLAY
HE10 CONNECTOR
DC JACK 5V
W1
W2
R3
330
DS2
S3
SW-SPDT
VDD ST5
VDD ST7
D1
1N4001
U2
IN
OUT
GND
LF33CV
VDD
C8
100nF
C1
220 F/16V
C3
220 F/16V
C9
100nF
U3
MAX7219
ISET
DIN
LOAD
CLK
DOUT
GND
SEG A
SEG B
SEG C
SEG D
SEG E
SEG F
SEG G
SEG DP
DIG 0
DIG 1
DIG 2
DIG 3
DIG 4
DIG 5
DIG 6
DIG 7
R6
10K
C10
100nF
U4
4.AFF.7.SEG
JP3
Header 26H
C14
100nF
U5
ST52F513
C11
100nF
PC0/SCK
PC1/MOSI
PC2/MISO
PC5/TRES
PC3/SS
PB7/AIN7
PB6/AIN6
PB5/AIN5
PB4/AIN4
PB3/AIN3
PA6/T0OUT
PA2/T1OUT
PB2/AIN2
PA1/SDA
PA0/SCL
PB0/VREF/AIN0
PB1/AIN1
VPP
RESET ST5
PA3/RX
PA4/TSTRT
PA5/TCLK/TX
PA7/INT
PC4TX
U1
ST7LITE2
SS/AIN0/PB0
SCK/AIN1/PB1
MISO/AIN2/PB2
MOSI/AIN3/PB3
CLKIN/AIN4/PB4
AIN5/PB5
AIN6/PB6
PA7
MCO/ICCCLK/BREAK/PA6
ATPWM3/ICCDATA/PA5
ATPWM2/PA4
ATPWM1/PA3
PA2/ATPWM0
PA1/ATIC
PA0/LTIC
OSC1/CLKIN
OSC2
VDD
VSS
RESET
ADC
PWM
ANALOGIQUE 2
C15
100nF
C12
100nF
C13
100nF
TP4
ANALOGIQUE 1
ANALOGIQUE 2
LEDs
S6
SW DIP-8
DC MOTOR (PWM)
TP1
TP2
W3
R5
1K
D2
1N4148
R4
15
Q1
NPN
M1
C16
100nF
BUZZER
LS1
R8
W4
W5
Jumper
R7
4.7K
S4
SW-PB
C5
Cap
100nF
ADC
R20
10K
TP3
W6
W7
R17
4.7K
C6
100nF
RESET ST5
R18
4.7K
S2
C7
100nF
JP2
HE10
VPP
ISP ST5 CONNECTOR
RESET ST7
R2
4.7K
S1
C4
100nF
JP1
HE10
ISP ST7 CONNECTOR
Title
PROJET ST7/ST5 ISEN-TOULON 2004
Size
B
Number
Revision
Date: 03/03/2005
Sheet of
File:
Drawn By:

Рис. Б.4

Додаток В

Система команд мікроконтролерів ST7FLITE

1) Інструкції очистки (clean) та завантаження (load) регістрів

CLR Clear :	очистити регістр
LD Load :	завантажити значення у регістр/td>

2)Інструкції арифметичних операцій (arithmetic instructions):

ADC Addition with Carry:	додавання із перенесенням
ADD Addition :	додавання без перенесення
MUL Multiply :	множення
SBC Subtraction with Carry:	віднімання із запозиченням
SUB Subtraction :	віднімання без запозичення

3)Інструкції для роботи зі стеком (stack operations):

POP from Stack	отримати дані зі стеку
PUSH Push into the Stack	помістити дані у стек
RSP Reset Stack Pointer	скинути вказівник стеку (встановити у положення #FF)

4)Інструкції зсуву (shift and rotate instructions):

RLC Rotate Left through Carry	зсув ліворуч через перенесення
RRC Rotate Right through Carry	зсув праворуч через перенесення
SLA/SLL Shift Left Arithmetic	зсув ліворуч, арифметичний
SRA Shift Right Arithmetic	зсув праворуч, арифметичний
SRL Shift Right Logical	зсув праворуч, логічний
SWAP Nibbles	обміняти старші та молодші тетради регістрів

5)Інструкції інкременту та декременту (Increment/Decrement Instructions):

DEC Decrement	зменшити на одиницю
INC Increment	збільшити на одиницю

6)Інструкції порівняння та перевірки (Compare and Test Instructions):

BCP Logical Bit Compare	логічне бітове порівняння
CP Compare	порівняння
TNZ Test for Negative or Zero	перевірка знаку змінної та рівність її нулю

7) Логічні операції (Logical Operations):

AND Logical And	логічне І
CPL Logical 1-Complement	інверсія (логічне доповнення до одиниці)
NEG Negate	додатковий код числа (зміна знаку)
OR Logical Or	логічне АБО
XOR Logical Exclusive Or	виключальне АБО

8)Інструкції безумовного переходу та виклику (Unconditional Jump or Call Instructions):

CALL Subroutine Absolute	безумовний виклик підпрограми
CALLR Subroutine Relative	відносний виклик підпрограми
JP Jump Absolute	безумовний перехід
JRA Jump Relative Always	завжди відносний перехід
NOP No Operation	пуста операція
RET Return from Subroutine	повернення із підпрограми

9)Бітові інструкції (Bit Operations):

BRES Bit Reset	скинути біт
BSET Bit Set	встановити біт
BTJF Bit Test and Jump if False	перевірка біту, перехід при неспівпадінні
BTJT Bit Test and Jump if True	перевірка біту, перехід при співпадінні

10)Інструкції умовного переходу (Conditional Jump Instructions):

JRC Jump Relative if Carry	відносний перехід при встановленому перенесенні
JREQ Jump Relative if Equal	відносний перехід при рівності
JRF Jump Relative if False	відносний перехід при логічній нерівності
JRH Jump Relative if Half-Carry	відносний перехід при половинному перенесенні
JRIH Jump Relative if Interrupt High	відносний перехід при перериванні

	високого рівня
JRIL Jump Relative if Interrupt Low	відносний перехід при перериванні низького рівня
JRM Jump Relative if Interrupt Mask	відносний перехід при маскованому перериванні
JRMI Jump Relative if Negative	відносний перехід при від’ємному значенні
JRNC Jump Relative if No Carry	відносний перехід при відсутності перенесення
JRNE Jump Relative if Not Equal	відносний перехід при нерівності
JRNH Jump Relative if No Half-Carry	відносний перехід при відсутності половинного перенесення
JRNM Jump Relative if No Interrupt Mask	відносний перехід при немаскованому перериванні
JRPL Jump Relative if Positive or Zero	відносний перехід при додатному значенні або рівності нулю
JRT Jump Relative if True	відносний перехід при логічній істинності
JRUGE Jump Relative if Unsigned Greater or Equal	відносний перехід якщо значення більше або рівне
JRUGT Jump Relative if Greater Than	відносний перехід якщо значення більше
JRULE Jump Relative if Lower or Equal	відносний перехід якщо значення менше або рівне
JRULT Jump Relative if Lower Than	відносний перехід якщо значення менше

11)Керування перериваннями (Interrupt Management)

HALT	зупинка виконання програми
IRET Interrupt Return	повернення із переривання
TRAP Software Interrupt	програмне переривання
WFI Wait for Interrupt	очікування переривання

12)Управління прапорцями (Condition Code Register)

RCF Reset Carry Flag	скинути прапорець перенесення
RIM Reset Interrupt Mask	скинути маскування переривань
SCF Set Carry Flag	встановити прапорець перенесення
SIM Set Interrupt Mask	встановити маскування переривань

Режими адресації даних мікроконтролера ST7 Microelectronics

Для команд, які працюють з пам'яттю можливі наступні режими адресації:

1. **Регістрова**, в якій значення з одного регістру передається в інший. Приклад: ld A,X.
2. **Безпосереднє** вказування значення. Характеризується наявністю значка " # " перед самим значенням. Приклад: ld A,#$0A – завантажити в акумулятор число 10 або ld X,#$81 – завантажити в індексний регістр X число 129.
3. **Пряме** звернення до комірки пам'яті (або спеціалізованого регістру) з вказуванням адреси. Приклад: ld A,$0A – завантажити в акумулятор число, що знаходиться за адресою $0A, тобто, вміст регістру LTCNTR, або ld $81,Y – вивантажити в комірку пам'яті за адресою $81 індексний регістр Y.
4. Звернення до комірки **за індексом**. Приклад: ld A,(12345,X) – завантажити в акумулятор число, що знаходиться за адресою (X +12345). Замість регістру X може бути регістр Y. Можливі варіанти, коли відсутнє зміщення, наприклад: ld A,(X) або ld (X),A. В цьому випадку адреса комірки буде не більше $FF.
5. **Непряма** адресація використовує комірку як індекс. Приклад: ld A,([$80],X) – адреса формується як сума регістру X та вмісту комірки $80. З отриманої адреси значення передається в акумулятор. Замість регістру X може бути регістр Y, наприклад: ld ([$80],Y),X – переписати регістр X в пам'ять за адресою (Y + вміст комірки $80).