МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«Харківський політехнічний інститут»

Ю. С. Грищук

МІКРОКОНТРОЛЕРИ:

АРХІТЕКТУРА, ПРОГРАМУВАННЯ ТА ЗАСТОСУВАННЯ В ЕЛЕКТРОМЕХАНІЦІ

Навчальний посібник
для студентів електромеханічних спеціальностей

Харків
НТУ «ХПІ»
2019

УДК 004.315(075)
Г85

Р е ц е н з е н т и :
О. Д. Черенков, д-р техн.наук, професор ХНТУСГ;
О. Б. Богаєвський, д-р техн. наук, професор ХНАДУ

Рекомендовано Вченою радою НТУ «ХПІ» як навчальний посібник
для студентів електромеханічних спеціальностей
протокол № 3 від 01.03.2019 р.

Грищук Ю. С.

Г85 Мікроконтролери: Архітектура, програмування та застосування в електромеханіці : навч. посіб. / Ю. С. Грищук. – Харків : НТУ «ХПІ», 2019. – 384 с.

ISBN

Викладено навчально-теоретичний матеріал, що описує основні терміни і поняття, архітектуру мікропроцесорів і однокристальних мікроконтролерів, інтерфейси, запам'ятовуючі і спеціалізовані периферійні пристрої. Посібник містить лабораторний практикум, особливості і приклади програмування та практичного застосування мікроконтролерів, контрольні запитання і варіанти завдань з прикладами їх виконання.

Призначено для студентів спеціальності 141 «Електроенергетика електротехніка і електромеханіка» спеціалізацій 141.07 «Електричні апарати» та 141.08 «Електропобутова техніка» і може бути корисним для студентів технічних вузів, аспірантів та інженерно-технічних працівників.

Іл. 83. Табл. 51. Бібліогр.: 44 назви.

УДК 004.315(075)

ISBN

ЗМІСТ

ПЕРЕДМОВА

Однією з найбільш важливих умов, що забезпечують розвиток енергетики, промисловості, транспорту й усіх інших галузей виробництва є їх комплексна автоматизація на основі сучасної мікропроцесорної техніки.

Конкретні задачі науково-технічного прогресу в паливно-енергетичному, електротехнічному, машинобудівному і транспортному комплексах висувають у цей час на перше місце проблему забезпечення необхідної надійності, живучості і безаварійності складних технічних об'єктів і їх систем керування. До таких об'єктів належать сучасні потужні електроенергетичні системи й їх елементи (наприклад, електричні станції, далекі лінії електропередачі, електричні розподільні системи великих промислових підприємств, автономні електроенерге-тичні, перетворювальні, електротехнічні і електромеханічні установки і системи керування ними), що включають пристрої релейного захисту і протиаварійної автоматики, контактні розподільні пристрої. Комутація, захист, керування, регулювання й інші функції в цих об'єктах, як правило, виконуються різними електричними апаратами (ЕА).

Останніми роками у зв'язку з появою мікроконтролерів (МК) зявилася тенденція до широкого використання їх в електропобутовій техніці (ЕПТ) і електроапаратобудуванні, до безпосереднього впровадження в пристрої керування ЕА, техніку релейного захисту, системи протиаварійної автоматики, в автоматизовані системи керування технологічними процесами (АСКТП) виробництва, випробування і дослідження ЕПТ і ЕА. Це викликано появою нових підвищених і різноманітних вимог до систем автоматизації керування об'єктами. У зв'язку з цим традиційні системи керування і регулювання, виконані на аналогових елементах, стали не взмозі конкурувати з цифровими пристроями, в яких використовуються мікропроцесори і мікроконтролери.

Відмінними рисами пристроїв і систем, виконаних на базі МП і МК, є таке: можливість значного розширення функцій шляхом додавання нових алгоритмів і програм до системи програмного забезпечення, високий рівень уніфікації елементів, можливість перепрограмування для реалізації тих або інших функцій без зміни комплексу технічних і апаратних засобів і автоматизації процесів діагностики і настроювання апаратури. Сучасні МК мають малі габаритні розміри і можуть розміщатися поруч з керованими об'єктами, мають високу надійність і розвинені логічні можливості, характеризуються низькими енергоспоживанням і вартістю. У багатьох пристроях сучасні МК можуть складатися тільки з однієї ВІС з одним рівнем напруги від 1,8 до 5 В. Крім того, їх застосування в системах керування забезпечує високу швидкодію, продуктивність, гнучкість і ефективність.

Застосування МП і МК вимагає від розробників корінного перегляду традиційних методів при проектуванні систем керування ЕПТ, ЕА і електронними апаратами, заміни у багатьох випадках проектування схем і систем керування розробкою програм настроювання мікропроцесорної апаратури на виконання певних функцій. Вирішення цих задач вимагає підготовки кваліфікованих фахівців, здатних проектувати, розробляти, експлуатувати й обслуговувати таку складну МП техніку, як ЕА і ЕПТ з мікропроцесорним керуванням.

Для підготовки таких фахівців виникає необхідність у виданні додаткової навчальної літератури, що містить одночасно загальні відомості про мікропроцесорну техніку і нову інформацію про розвиток мікропроцесорних засобів. Зокрема, потрібна інформація про однокристальні мікроконтролери , які часто в публікаціях називають мікро-ЕОМ, і їх застосування в ЕПТ, електроапаратобудуванні й інших галузях. Доцільним і корисним, на думку автора, є включення інформації про лабораторний практикум, який виконується на реальних лабораторних стендах, створених на базі МК, що дозволяє досліджувати їх будову, принцип дії і різні режими роботи, програмування сучасних, найбільш широко вживаних МК.

Мета посібника – сприяти поглибленню знань студентів, що навчаються за спеціальностями 141 «Електроенергетика, електротехніка і електромеханіка» спеціалізацій 141.07 «Електричні апарати» та 141.08 «Електропобутова техніка» й за іншими технічними спеціальностями, в області мікропроцесорних систем, і отриманню ними практичних нави-

чок у роботі з реальними мікропроцесорними пристроями й однокристальними МК. Посібник складений за матеріалами лекцій, лабораторних і практичних занять, курсового проектування дисциплін, що повя'зані з мікропроцесорною технікою й її використанням в ЕПТ і ЕА, які включені в навчальний процес НТУ «ХПІ» і проводяться автором з 1984 року до теперішнього часу.

Зміст посібника визначений, перш за все, прагненням викласти ті аспекти мікроконтролерів і їх застосування в ЕПТ, електроапаратобудуванні та в інших напрямках, які недостатньо відображені в літературі, що є на теперішній час. Зокрема, в посібнику описаний один з дев'яти лабораторних стендів, розроблених на основі мікроконтролерів сімейства MCS51 (МК51) на кафедрі «Електричні апарати» НТУ «ХПІ» за ініціативи і при особистій участі автора посібника. Стенд дозволяє досліджувати 24 режими роботи МК. Тут же подано лабораторний практикум до вивчення МК, складений автором і апробований у навчальному процесі.

У лабораторному практикумі передбачається ознайомлення із структурою мікроконтролерів та їх системою команд, дослідження різних режимів роботи, складання і введення програм з клавіатури лабораторного стенда та програмування МК на мові Асемблер з використанням комп'ютера, налагодження програм крос-засобами з використанням емулятора, дослідження роботи аналого-цифрових перетворювачів і портів введення-виведення та ін.

У посібнику викладені відомості про сучасні МК провідних зарубіжних виробників: сімейств MCS51, MCS151 і MCS251 фірми Intel та сумісних з ними високопродуктивних (від 20 до 100 MIPS) МК х-51 різних інтерпретацій інших фірм, серій AT89, AT90 і Atmega16 фірми Atmel, сімейств PIC16/17 і PIC16X7XX фірми Microchip, сімейств HC05, HC08, HC11 фірми Motorola, сімейств цифрових сигнальних процесорів (ЦСП) TMS320Cxxx (платформи C2000, C5000 і C6000 продуктивністю від 20 MIPS до 8800 MIPS) і сімейств 16-розрядних МК з ультранизьким споживанням MSP430, MSP430F фірми Texas Instruments. Розглянуто архітектуру, структурні схеми, особливості різних модифікацій сімейств, вказано їх основні технічні характеристики, а також сфери практичного застосування. Наведено приклади застосування різних МК в системах керування транспортом, комплектними розподільними пристроями (КРП), гнучкими системами релейного захисту (ГСРЗ), електромеханічними си-

стемами і електроприводами, технологічними процесами випробувань і досліджень електричних апаратів та електропобутової техніки, в автоматичних вимикачах та ін.

Автор вважає своїм приємним обов'язком подякувати: ст. наук. співробітника кафедри «Автоматики та управління в технічних системах» *В. М. Лещенка* за консультації і допомогу в розробці і налагодженні мікроконтролерних лабораторних стендів і програм; *С. Ю. Грищуку* за надані приклади програм і завдань до лабораторного практикуму; студентам і співробітникам кафедри «Електричні апарати за надану допомогу при створенні лабораторних стендів і модернізації навчальної лабораторії мікропроцесорних систем; *усім*, чия допомога і підтримка зробили можливим появу цієї книги.

Автор висловлює глибоку подяку рецензентам:

О. Д. Черенкову, доктору техн. наук, професору кафедри теоретичної електротехніки Харківського національного технічного університету сільського господарства ім. Петра Василенка;

О. Б. Богаєвському, доктору техн. наук, професору кафедри автомобільної електроніки Харківського національного автодорожнього університету.

Ряд цінних порад і зауважень, які були зроблені ними при рецензуванні рукопису, суттєво сприяли поліпшенню даного посібника.

ВСТУП

У розвитку електронних цифрових обчислювальних машин відповідно до використовуваних технологічних принципів розрізняють п’ять поколінь цифрових машин: 1) на електронних лампах; 2) на транзисторах; 3) на інтегральних схемах (ІС); 4) на інтегральних схемах з великим ступенем інтеграції (ВІС); 5) на інтегральних схемах з надвеликим ступенем інтеграції (НВІС).

Перший проект електронно-цифрової машини розробив у 1937–1939 рр. у м. Эймс (штат Айова, США) професор фізики і математики Джон В. Анатасов, болгарин за походженням [1], який повідомив в своєму листі до професора математики, академіка Національної академії наук України, Кравчука Михайла Пилиповича, що при розробці цього проекту він скористався його математичними розробками [https // ru. wikipedia.org /wiki/ Кравчук Михаил Ф.]. За цим проектом у керованій частині машини містилося 300 електронних ламп і 1642 конденсатори [1].

У 1945 р. американський математик Джо фон Нейман істотно розвинув ідею програмного керування обчислювальним процесом і сформулював принципи організації пам’яті. Після винаходу транзистора в 1948 р. почався процес заміни електронних ламп дискретними напівпровідниковими пристроями, а потім інтегральними схемами, великими і надвеликими інтегральними схемами.

Створення мікропроцесора (програмно-керованого пристрою, що здійснює процес оброблення інформації і керування, побудованого на одній або декількох ВІС) почалося порівняно недавно і стало наслідком розвитку і вдосконалення технології виробництва ІС і ВІС.

Перші повідомлення про розробку мікропроцесора І-4004 опублікувала фірма Intel у 1971 р., мікроконтролера І8048 в 1976 році. Класичний зразок мікроконтролера Intel 8051 був випущений в 1980 році.

Створення мікропроцесора як функціонально закінченої частини ЕОМ на одній інтегральній схемі є одним із найбільших досягнень минулого двадцятого століття. Здатність програмувати послідовності виконуваних функцій, тобто здатність працювати за заданою програмою, є основною відмінністю МП від елементів «жорсткої» логіки (інтегральних схем малого і середнього ступеня інтеграції). Удосконалення технологій виробництва ІС і ВІС привело до того, що за порівняно невеликий

час з'явилися чотири покоління МП, які відрізняються своїми технічними характеристиками [2]:

перше – повільно діючі (час виконання команди 10–20 мкс), чотирирозрядні МП, що мають відносно обмежений набір команд, обсяг пам'яті і види адресації;

друге – чотири- і восьмирозрядні МП з часом виконання команди 2–5 мкс, розширеним набором команд і обсягом пам'яті та різними видами адресації;

третє – швидкодіючі (час виконання команди 100–300 нс), секціоновані МП, виконані з використанням біполярної технології і мікропрограмним принципом керування, а також 16-розрядні процесори і спецпроцесори;

четверте – однокристальні мікро-ЕОМ з вбудованими портами введення-виведення і пристроями, що запам'ятовують, 32-розрядні МП.

Сучасні МП – 64-розрядні дво-чотири- восьмиядерні МП та 8-16-32-розрядні мікроконтролери (МК), що містять в одному корпусі НВІС таке: МП, роздільні пам'ять даних і пам'ять програм, флеш-пам'ять, паралельні і послідовні порти введення-виведення, таймери/лічильники, АЦП, ЦАП, сторожовий таймер, внутрішній температурний сенсор та інші елементи, необхідні для ефективного використання МК в системах керування різним устаткуванням.

Основні причини широкого впровадження мікропроцесорної техніки такі: використання в мікропроцесорних системах цифрового способу подання інформації, що дозволяє значно підвищити швидкість її передачі і перешкодостійкість; застосування програмного способу оброблення інформації, компактність, висока надійність і низьке споживання електроенергії.

УМОВНІ СКОРОЧЕННЯ І ПОЗНАЧЕННЯ

1. Українська нотація

(*у дужках наведено англійську нотацію, див. п. 2*)

А – акумулятор (див. ACC)
АВ – автоматичний вимикач
АЛП – арифметико-логічний пристрій
АСКТПВ – автоматизована система керування технологічним процесом випробувань (досліджень)
АЦП – аналого-цифровий перетворювач (див. ADC)
ВІС – велика інтегральна схема
ВК – вибір корпусу (див. CE)
ВРПП – сигнал відключення резидентної пам'яті програм (див. EA)
ВИХПЕР – вихід передавача УАПП (див. TXD)
ВХПР – вхід приймача УАПП (див. RXD)
ДЗПП – сигнал дозволу зовнішньої пам'яті програм (див. PSEN)
ЕА – електричний апарат
ЕПТ – електропобутова техніка
ЗПР – запит переривання (див. INT)
ЗПД – зовнішня пам'ять даних
ЗПП – зовнішня пам'ять програм
ЗП – сигнал запису, що керує (див. WR)
ІС – інтегральна схема
КОП – код операції (поле в тілі команди)
КРОК – керуючий сигнал покрокового (покомандного) режиму, роботи (див. SS)
КРП – комплектно-розподільний пристрій
ЛК – лічильник команд (див. PC)
МК – мікроконтролер (однокристальний мікроконтолер)
МК51 – мікроконтролери серії 1816: КМ1816ВЕ51
МП – мікропроцесор
МПРЗ – мікропроцесорний релейний захист
МПС, МПК – мікропроцесорна система, мікропроцесорний комплект
НВІС – надвелика інтегральна схема
ОЗП – оперативний запам'ятовуючий пристрій (див. RAM)
ПП – пам'ять програм
ПРОГ – сигнал, що керує програмуванням резидентної пам'яті програм (див.PROG)
РА – регістр адреси (див. RAR)

РВВ – ВІС розширювача введення/виведення
РЗП – регістр загального призначення
РК – регістр команд (див. IR)
РКП – регістр керування потужністю (див. PCON)
РКПП – регістр керування послідовного порту (див. SCON)
РКСТ – регістр керування/статусу таймера (див. TCON)
РМП – регістр маски переривань (див. IE)
РП – регістр пріоритетів (див. IP)
РПД – резидентна пам’ять даних
РПП – резидентна пам’ять програм
РУС – регістр указівника стека (див. SP)
РРТЛ – регістр режиму таймера/лічильника (див. TMOD)
РСФ – регістри спеціальних функцій (див. SFR: PSW, TMOD, TCON, SCON, PCON, IE, IP)
РУД – регістр указівника даних (див. DPTR)
САЗП – сигнал строба адреси зовнішньої пам’яті (див. ALE)
СКД – керуючий сигнал скидання (див. RST)
ССП – слово стану програми (див. PSW)
Т/Л – таймер/лічильник подій (див. TCNT)
УАПП – універсальний асинхронний приймач-передавач
ЦАП – цифро-аналоговий перетворювач (див. DAC)
ЦПП – центральний процесорний пристрій (процесор) (див. CPU)
ЧТ – керуючий сигнал читання (див. RD)
ША, ШД – шина адреси пам’яті (див. MAB), шина даних пам’яті (див. MDB)
ШЗ – швидкодіючий запобіжник

2. Англійська нотація

A – регістр-акумулятор
AC – допоміжне перенесення (Auxiliary Carry flag in PSW)
ACC – символічне ім’я регістра-акумулятора A
ad (dir) – пряма 8-бітова адреса байта РПД (0-127), порту або РСФ
ADC – аналого-цифровий перетворювач
add – пряма 8-бітова адреса призначення
ads – пряма 8- бітова адреса джерела
ad11 – пряма 11-бітова адреса передачі керування
ad16 – пряма 16-бітова адреса передачі керування
ad16h – старший байт прямої 16-бітової адреси
ad16l – молодший байт прямої 16-бітової адреси

B – регістр-розширювач акумулятора
bit – пряма 8-бітова адреса біта (МК51)
C – прапор перенесення
CAN – контролер локальної мережі (Controller Area Network)
CE (CS) – Chip Enable (Chip Select) (див. ВК)
CLK – синхросигнал (Clock)
CPU – центральний процесорний пристрій (Central Processor Unit)
$C/\overline{T}$ – керуючий біт вибору режиму таймера/лічильника, (Timer or Counter selector in TMOD)
– 8-бітовий безпосередній операнд (константа)
#d16 – 16-бітовий безпосередній операнд (константа)
#d16h – старший байт 16-бітового безпосереднього операнда
#d16l – молодший байт 16-бітового безпосереднього операнда
DAC – цифро-аналоговий перетворювач
DPH – Data Pointer High (старший байт РУД)
DPL – Data Pointer Low (молодший байт РУД)
EA – керуючий біт зняття блокування всіх переривань (Enable All Control bit in IE)
ECCP – модернізований РСР
$\overline{EA}$ / VPP – External Address/Voltage Power Programming (див. ОРПП)
EPROM – Erasable Programmable Read Only Memory (див. РПП)
ES – керуючий біт дозволу переривання від УАПП (Enable Serial port control bit in IE)
ET – керуючий біт дозволу переривання від таймера (Enable Timer port control bit in IE)
EX – керуючий біт дозволу зовнішнього переривання (Enable External interrupt control bit in IE)
F0, F1 – прапори, які специфікуються користувачем
GATE – біт керування блокуванням T/C (Gating control bit in TMOD)
GFLOPS – міліард операцій з плаваючою крапкою в секунду
GFO, GF1 – прапори користувача (General Flags in PCON) в МК51
i – біт в КОП, що визначає регістр непрямої адреси: i = 0, 1 (R0, R1)
I^2C – інтерфейс інтегральних схем
IDL – керуючий біт холостого ходу (Idle mode in PCON)
IE – 1) Interrupt Enable register (див. РМП)
2) прапор зовнішнього переривання, встановлений за спадом з сигналу ЗПР (Interrupt Edge flag in TCON)
IP – Interrupt Priority control register (див. РПП)
INT – Interrupt (див. ЗПР)

IR – регістр команд (Instruction Register) (див. РК)
IT – біт вибору типу (рівень/спад) керуючого сигналу ЗПР (Interrupt Type control bit in TCON)
I/O – введення-виведення
MAB, MDB – шина адреси пам'яті , шина даних пам'яті
M0, M1 – керуючі біти вибору режиму роботи Т/С (Operating Mode inTMOD)
MI^{2}C – ведучий I^{2}C
MIPS – мільйон операцій у секунду
OV – прапор переповнювання (Overflow flag in PSW)
P – прапор паритету (Parity flag in PSW)
PC – лічильник команд (Program Counter) (див. ЛК)
PCON – регістр керування потужністю Power Control register (див. РКП)
PD – біт керування потужністю (Power Done in PCON)
PROG – Programming EPROM (див. ПРОГ)
PS – керуючий біт пріоритету УАПП (Serial port Priority control bit in IP)
PSEN – Program Store Enable (див. ДЗПП)
PSP – паралельний ведений порт
PSW – Program Status Word (див. ССП)
PT – керуючий біт пріоритету таймера (Timer Priority controbit in IP)
PWM – широтно-імпульсний модулятор
PX – керуючий біт пріоритету зовнішнього переривання (External interrupt Priority control bit in IP)
PX.Y – символічне ім'я біта Y порту X; Y = 0 ÷ 7 для МК51: X = 0, 1, 2, 3
RAM – Random Access Memory (див. ОЗП)
RAR – RAM Address Register (див. РА, укр.)
RB8 – де'вятий (bit 8) прийнятий біт (Receive Bit 8 in SCON)
RD – Read (див. ЧТ)
rel – 8-бітова відносна адреса передачі керування (-127÷+128) у
МК51 REN – керуючий біт дозволу прийому в УАПП (Receive Enable Control bit in SCON)
RI – прапор переривання від приймача (Received Interrupt flag in SCON)
Ri – узагальнене ім'я регістра непрямої адреси (R0 або R1)
Rn – узагальнене ім'я робочого регістра (n = 0 ÷ 7)
rrr – 3-бітове поле в коді операції, що визначає регістр загального призначення (R0-R7)

RS – керуючий біт вибору банку регістрів (Register bank Select in PSW)
RST – Reset (див. СКД)
RST/VPD – Reset/Voltage Power Done (див. СКД)
RTI – система переривань реального часу (Real Time Interrupt)
RXD – Receive Data pin (див. ВХПР)
Sn – стан пристрою керування МК51 n = 1 ÷ 6 (State)
SCON – Serial Port Control/status register (див. РУПП)
SCSI – високошвидкісний інтерфейс, призначений для жорстких дисків, а також деяких зовнішніх пристроїв, призначених для введення даних у комп'ютер
SM0, SM1, SM2 – керуючі біти режиму роботи УАПП (Serial port Mode control bits in SCON)
SMOD – керуючий біт подвійної швидкості передачі (Double Baud rate in PCON)
SP – Stack Pointer (див. РУС)
SPI – синхронний послідовний інтерфейс
SS – Single Step (див. КРОК)
STB – Strobe (див. СТБ)
SXPY – пояснення на тимчасових діаграмах і схемах, що прив'язують сигнали до станів S пристрою керування і фаз P синхросигналів (X = 1÷6; Y = 1, 2) для МК51
T0, T1 – тест-входи МК
TB8 – дев'ятий біт (bit 8), що передається (Transmit Bit8 in SCON)
TCNT – Timer/Counter events (див. Т/С)
TCON – Timer/Counter Control/status register (див. РКСТ)
TF – прапор переповнювання таймера (Timer overflow Flag in TCON)
TH – старший байт таймера (Timer High byte)
TI – прапор переривання від передавача (Transmit Interrupt flag in SCON)
TL – молодший байт таймера (Timer Low byte)
TMOD – Timer/counter Mode register (див. РРТЛ, укр.)
TR – керуючий біт пуску таймера (Timer Run control bit in TCON)
TXD – Transmit Date (див. ВИХПЕР укр.)
USART – універсальний синхронно-асинхронний прийомопередавач
USB – універсальна послідовна шина
Vref – джерело опорної напруги
WDT – сторожовий таймер

WR – Write (див. ЗП)
AND, NOT, OR, XRL – логічні операції відповідно І, НІ, АБО, виключаюче АБО
HIGH, LOW – логічні операції виділення відповідно старшого і молодшого байтів із d16 при асемблюванні

3. Спеціальні символи

← – оператор присвоєння (заміщення)
↔ – оператор взаємного обміну
∧, ∨, ∀ – оператори логічних операцій: І (кон'юнкція) АБО (диз'юнкція), виключаюче АБО
@ – префікс непрямої адресації
# – префікс безпосереднього операнда: B, H – суфікси відповідно двійкового (Binary) і шістнадцяткового (Hex.) кодів
(Y) – вміст регістра або елемента пам'яті з ім'ям Y
((Y)) – вміст елемента пам'яті, що адресується вмістом Y (непряма адресація)
¤ – поточний вміст лічильника команд МК

ОСНОВНІ ТЕРМІНИ І ВИЗНАЧЕННЯ

Мікропроцесорна техніка (МПТ) має свою специфічну термінологію, в якій постійно з'являються і входять у практику нові терміни, визначення і поняття. Нижче наводяться рекомендовані Міжнародним центром наукової і технічної інформації, Міжнародним науково-дослідним інститутом проблем керування [1] і загальноприйняті в сучасній літературі з МП і МК [1,2,4,5] основні терміни і визначення, які використовуються в даному посібнику.

Знання основних термінів украй важливе, оскільки є істотним чинником для студентів на початковому етапі вивчення основ МП і МК техніки і дозволяє надалі якісно сприймати матеріал цього курсу, самостійно вивчати сучасну літературу та освоювати і застосовувати на практиці нові МП та МК пристрої.

Основні терміни, які використовуються в МПТ

Адреса – вказівка місцерозташування об'єкта в пам'яті ЕОМ.

Адресний простір – максимальне число секцій пам'яті (у кілобайтах, мегабайтах, гігабайтах), які можна пронумерувати (проадресувати) / за допомогою шини адреси, реалізованої в мікропроцесорі. Якщо шина адреси має 16 дротів (16 розрядів), то адресний простір дорівнює 2^{16} байт, тобто 64 Кбайт.

Алгоритм – набір розпоряджень, що однозначно визначають зміст і послідовність виконання операцій для систематичного розв'язання певної задачі.

Аналого-цифровий перетворювач (АЦП) – пристрій, що перетворює безперервний (аналоговий) сигнал у дискретні цифрові величини.

Арифметико-логічний пристрій *(АЛП)* – функціональна частина процесора, що виконує арифметичні і логічні дії над даними.

Архітектура мікропроцесора – складові частини мікропроцесора, а також їх взаємне з'єднання і взаємодія між ними. Архітектура включає: 1) структурну схему самого МП; 2) програмну модель МП (опис функцій регістрів); 3) інформацію про організацію пам'яті (місткість пам'яті і способи її адресації); 4) опис організації процедур введення-виведення і керування; 5) опис системи команд та ін. Існують два основних типи архітектури, що застосовуються в даний час – *Неймановська* і *Гарвардська. В Неймановській архітектурі програма користувача й оброблюваних да-*

них зберігаються в одному ОЗП . Цей принцип сформулював американський математик Джо фон Нейман у 1945 році. На цьому принципі побудований класичний мікропроцесор I8080 (К580). Пізніше в Гарвардському університеті (США) з метою підвищення швидкодії ЕОМ за рахунок зміни її структурної схеми, а не тільки підвищуючи її частоту, було розроблено *Гарвардську архітектуру*. Вона передбачає наявність двох шин даних і двох розділених запам'ятовуючих пристроїв: окремо ПЗП для програми, що виконується, і окремо ОЗП для оброблюваних даних, з можливістю одночасного до них звернення. На її основі побудований класичний мікроконтролер I8051 (К1816ВЕ51) та будуються сучасні, найбільш продуктивні мікроконтролери.

Асемблер – системна обслуговуюча програма, що перетворює символічні інструкції в команди машинної мови і дозволяє проводити діагностику, формування посилань для редактора зв'язків і т. ін.

ASCII-код – стандартизована система позначень різних символів (букв і цифр клавіатури комп'ютера) для пересилання інформації (American Standart Code for Information Interchange).

Байт – оброблюваний як єдине ціле елемент даних, що складається з послідовності двійкових розрядів. У мікро-ЕОМ, як правило, використовують восьмибітовий байт (Кбайт – 2^{10} = 1024 байт; Мбайт – мегабайт = 2^{20} байт; Гбайт – гігабайт = 2^{30} байт ≈ 1 000 000 Кбайт).

Біт – один двійковий розряд машинного слова, або одиниця інформації, що набуває значення 0 або 1.

Бод – одиниця швидкості передачі інформації послідовним двійковим кодом (біт у секунду).

Буфер – запамя'товуючий пристрій для тимчасового зберігання даних з метою узгодження асихронно працюючих пристроїв, або область ОЗП, що тимчасово резервується для виконання процедури введення-виведення.

Вбудовувана мікро-ЕОМ – мікро-ЕОМ, конструктивно пристосована для роботи у складі приладів і устаткування.

Відеотермінал — пристрій, що забезпечує можливість обміну даними по каналу зв'язку з віддаленою ЕОМ. Включає клавіатуру для введення і дисплей для виведення інформації.

Графічний пристрій – пристрій виведення, призначений для подання даних у вигляді графічного зображення на папері.

Дані – інформація (числа), призначена для оброблення в ЕОМ.

Діалоговий режим – режим взаємодії користувача з ЕОМ, при якому кожен запит користувача викликає негайну у відповідь дію ЕОМ.

Дисплей – пристрій, що забезпечує візуальне подання цифрової, алфавітно-цифрової і (або) графічної інформації на екрані електронно-променевої трубки, в плазмових панелях, на рідких кристалах, світлодіодах і т.ін. у формі, зручній для оператора.

Довжина слова – кількість бітів в одному машинному слові.

Доступ (звернення) – процедура встановлення зв'язку з пристроєм, що запам'ятовує, для вибірки / запису даних.

Емуляція – імітація функціонування однієї системи засобами іншої системи без втрати функціональних можливостей або спотворення одержуваних результатів.

Завантажувач – обслуговуюча програма для завантаження об'єктної програми в пристрій, що оперативно запам'ятовує (ОЗП).

Запам'ятовуючий пристрій (ПЗ) – виріб, що реалізовує функціональну частину ЕОМ, яка призначена для запам'ятовування і (або) видачі інформації.

Інтерпретатор – обслуговуюча програма, що здійснює пооператорну трансляцію і виконання початкової програми.

Інтерфейс – сукупність уніфікованих технічних і програмних засобів, необхідних для підключення даних пристроїв до системи або однієї системи до іншої.

Канал передачі даних – сукупність технічних засобів і пристроїв, що забезпечують передачу інформації і перетворення сигналів.

Канал прямого доступу до пам'яті – сукупність технічних засобів і пристроїв, що забезпечують прямий доступ до пам'яті без використання ЦП.

Кеш-пам'ять – допоміжна оперативна пам'ять, недоступна для програміста. Вона розміщується функціонально між процесором і оперативним пристроєм, що запам'ятовує, і служить для підвищення швидкодії мікропроцесорної системи. В ній зберігається і оновлюється вміст секцій пам'яті, що здубльований з пристрою ОЗП з командами, які найбільш часто вживаються в програмі.

Команда – розпорядження, що визначає крок процесу виконання програми; містить вказівку операції, адреси операндів та інші службові ознаки.

Компілятор – обслуговуюча програма, що виконує трансляцію на машинну мову програми, записаної на початковій мові програмування.

Контролер – пристрій, що виконує функції керування, передачі даних і звільняє від цих функцій процесор.

Контроль парності – метод контролю даних, при якому сума за модулем двох двійкових одиниць в машинному слові, включаючи контрольний розряд, повинна мати певну парність, тобто бути завжди парною або непарною.

Користувач – особа, що використовує даний обчислювальний пристрій для виконання необхідних йому робіт.

Лічильник команд – регістр, на основі вмісту якого здійснюється адреса наступної команди.

Магістраль – сукупність шин, що зв'язують між собою всі пристрої мікропроцесорної системи.

Маркер (курсор) – спеціальний знак на екрані дисплея для вказівки певних позицій або елементів.

Машинне слово – послідовність бітів або знаків, яка трактується в процесі обміну або оброблення як єдиний елемент даних.

Машинний код – двійковий код, в якому за специфічними для даної ЕОМ правилами кодується її система команд.

Мікро С, мікро Паскаль, – мови високого рівня, які призначені для програмування МК за допомогою ЕОМ.

Мікропроцесор (МП) – програмно-керований пристрій, що здійснює процес оброблення цифрової інформації і керування ним, побудований, як правило, на одній або декількох великих інтегральних схемах.

Мікропроцесорний комплект – сукупність мікропроцесорних та інших інтегральних мікросхем, сумісних за конструктивно-технологічним виконанням і призначених для сумісного застосування.

Мікроконтролер – ЕОМ, що виконує функціїї керування якимнебудь об'єктом або процесом і яка виконана, як правило, на одному кристалі.

Мікро-ЕОМ – ЕОМ, що складається з мікропроцесора, напівпровідникової пам'яті, засобів зв'язку з периферійними пристроями і при необхідності – пульта керування і джерела живлення, об'єднаних загальною конструкцією.

Місткість пам'яті – найбільший обсяг даних, виражений в одиницях інформації, який може одночасно зберігатися в запам'ятовуючому пристрої.

Мова Асемблер – символічна мова програмування, структура операторів якого визначається форматами команд і даними машинної мови.

Мова високого рівня – мова програмування, засоби якої допускають опис проблеми в наочному, легко сприйманому вигляді.

Мова програм низького рівня – машинний код, машинна мова, Асемблер.

Модем – модулятор і демодулятор, що об'єднаний в одному пристрої, який здійснює перетворення сигналів для передачі їх по лінії зв'язку.

Монітор – записана в ПЗП системна програма, що реалізовує операції обміну із зовнішніми пристроями і що допомагає здійснити налагодження програм.

Накопичувач на гнучкому магнітному диску – зовнішнє ЗП, в якому носіями інформації є змінні, гнучкі магнітні диски.

Накопичувач на компакт-дисках (CD-R, CD-RW, і DVD) – зовнішній ЗП, в якому носієм інформації є рельєфна підкладка з полікарбонату, на яку нанесений тонкий шар металу (звичайно алюмінію CD-R, CD-RW), що відбиває світло, і спеціального матеріалу (у дисках DVD), який під впливом лазерного променя змінює свій стан, переходячи з кристалічного в аморфний.

Непряма адресація – система адресації, при якій адресна частина інструкції містить адресу елемента пам'яті, що містить пряму адресу або іншу непряму адресу.

Обмін – обмін інформацією, процедура пересилання інформації у формі паралельного або послідовного коду.

Однокристальна ЕОМ – мікро-ЕОМ, побудована у вигляді однієї великої або надвеликої інтегральної схеми.

Операнд – елемент даних, над яким виконується операція.

Оперативний запам'ятовуючий пристрій (*ОЗП*) – ЗП з прямою адресацією, що відрізняється швидкістю доступу.

Оператор – допустима в мові програмування синтаксична конструкція, що відображає певну дію в програмі (присвоєння значення, передачу керування і т. д.).

Операційна система – комплекс взаємозв'язаних керуючих і обслуговуючих програм, що забезпечують автоматичне керування обчислювальними процесами і ресурсами ЕОМ при розв'язанні задач.

Паралельний інтерфейс – мікросхема, що реалізовує пересилання інформації в паралельному двійковому коді по шині з 8, 12, 16 паралельних електричних дротів і забезпечує введення-виведення інформації, наприклад з комп'ютера на принтер.

Паралельний порт – порт введення –виведення, через який дані передаються і приймаються паралельно, тобто одночасно всі розряди, що належать до даного символу або блоку даних.

Перепрограмований постійний запам'ятовуючий пристрій (*ППЗП*) – ЗП, в який інформація, що підлягає зберіганню, заноситься багато разів, але при цьому час запису значно перевищує час вибірки.

Переривання – тимчасове припинення виконання поточної програми і перехід до виконання програми обслуговування пристрою, що викликав переривання.

Підпрограма – частина програми, що допускає багаторазове звернення до неї з різних точок програми.

Указівник стека – регістр, що визначає адресу верхнього осередку використовуваного стека.

Порт – адреса зовнішнього пристрою (інтерфейсу, таймера), по якому можна зробити запис або зчитування інформації, і елемент памяті, який тимчасово зберігає інформацію, що пересилається.

Послідовний інтерфейс – мікросхема, яка реалізує пересилання інформації в послідовному коді по двопровідній лінії. Імпульси напруги двійкових кодованих чисел (біти) передаються послідовно один за одним (біт за бітом). Послідовний інтерфейс служить для пересилання інформації між окремими комп'ютерами (по телефонних лініях) або між керуючим комп'ютером і окремим об'єктом керування (більш ніж на 5–10 м). Як правило, послідовний інтрерфейс використовується в парі з модемом – модулятором-демодулятором, а також може використовуватися для зв'язку МК з комп'ютером.

Послідовний порт – порт введення-виведення, через який дані передаються і приймаються послідовно розряд за розрядом по двопровідній лінії.

Постійний запам'ятовуючий пристрій (*ПЗП*) – запам'ятовуючий пристрій з незмінним змістом пам'яті.

Пристрій введення-виведення – пристрій, що забезпечує обмін даними між оперативною пам'яттю ЕОМ і периферійними пристроями.

Програма – послідовність інструкцій, що реалізовують алгоритм. Програми звичайно можуть бути написані: а) у двійковому аб шістнадцятковому (машинному) коді, який безпосередньо сприймається процесором; б) мовою типу Асемблер; в) мовою високого рівня.

Програматор – спеціальний пристрій для запису підготовлених користувачем програм в ППЗП або РПЗП.

Програмований постійний запам'ятовуючий пристрій (*ППЗП*) – ПЗП, в який інформація заноситься одноразово користувачем і потім не змінюється.

Програмна сумісність – можливість виконання одних і тих самих програм на ЕОМ різних типів з отриманням ідентичних результатів.

Програмне забезпечення – сукупність програм, що забезпечують реалізацію функцій мікро-ЕОМ, мікропроцесорного пристрою або системи.

Пряма адресація – система адресації, при якій адресна частина інструкції містить адресу, яка визначає безпосередньо елемент пам'яті або місце на носії, що містить необхідний операнд.

Прямий доступ до пам'яті – метод, що дозволяє з великою швидкістю здійснювати завантаження даних з периферійного пристрою прямо в оперативний запам'ятовуючий пристрій (ОЗП).

Регістр – функціональний блок для зберігання машинного слова або його частини.

Редактор – обслуговуюча програма для редагування, набору даних з метою подання їх у вигляді, що сприймається засобами оброблення, або у відповідному форматі виведення.

Режим роботи в реальному масштабі часу – режим роботи системи, що забезпечує прийом до оброблення даних у міру їх надходження без яких-небудь обмежень і видачу результатів у необхідні інтервали часу.

Розряд – позиція для запису цифр числа в якій-небудь системі числення. В МПТ і ЕОМ використовуються двійкова, вісімкова, десяткова, двійково-десяткова і шістнадцяткова системи числення. Команди, що виконуються ЕОМ, зберігаються в пам'яті, пересилаються і виконуються тільки в двійковому коді. Кожному розряду двійкового числа відповідає провідник (лінія) шини, по якій передається це число в паралельному коді.

Розрядність мікропроцесора – одна з основних характеристик МП, яка визначається в загальному випадку розрядністю двійкового числа, що обробляється на основних операціях в АЛП за один раз і визначає його швидкодію. Як правило, розрядність МП відповідає розрядності регістрів загального призначення і шини даних.

Секційний мікропроцесор – мікропроцесор, одержаний на основі з'єднання однотипних 2-, 4-, 8- або 16-розрядних мікропроцесорних інте-

гральних схем, кожна з яких має в своєму складі АЛП і декілька загальних регістрів. Паралельне з'єднання цих мікросхем дозволяє побудувати мікро-ЕОМ з будь-якою бажаною довжиною машинного слова.

Символ – окремий знак із заданого набору умовних позначень, використовуваних для подання даних в ЕОМ.

Система команд – повний набір усіх інструкцій, допустимих у машинній мові даної ЕОМ.

Стек — пам'ять магазинного типу.

Таймер – пристрій (мікросхема), призначений для реалізації функцій, пов'язаних з відліком часу, наприклад для підрахунку кількості імпульсів напруги, їх затримки на заданий час, організації часових інтервалів, реалізації серії імпульсів заданої частоти і т. ін.

Тактова частота – основний параметр МП, який визначає швидкість оброблення інформації процесором, а також рядом інших мікросхем і окремих плат, які входять до складу комп'ютера.

Технічне забезпечення – сукупність технічних компонентів мікро-ЕОМ, мікропроцесорного пристрою або системи.

Утиліта – допоміжна невелика програма, що входить до системної програми, яка забезпечує нормальну роботу комп'ютера, його обслуговування і настроювання. До таких програм належить операційна система комп'ютера.

Файл – послідовність записів, що розміщується на зовнішніх запамятовуючих пристроях і яка розглядається в процесі пересилання і оброблення як єдине ціле.

FLASH-пам'ять – зовнішня для мікропроцесора пам'ять, яка реалізована мікросхемно і виконує функції жорсткого диска, стійка від ударів і вібрації.

Центральний процесор (ЦП) – процесорна ВІС, що безпосередньо здійснює процес оброблення даних.

Цифро-аналоговий перетворювач (*ЦАП*) – пристрій, що перетворює дискретний цифровий сигнал в безперервний аналоговий сигнал.

Шина – група ліній передачі інформації, об'єднаних загальною функціональною ознакою (наприклад, шина даних, адреси, керування).

1. ЗАГАЛЬНІ ВІДОМОСТІ ПРО МІКРОПРОЦЕСОРНІ ПРИСТРОЇ

Розвиток технологій виготовлення мікроелектронних схем привів до створення надвеликих інтегральних схем (НВІС), що являють собою універсальні за призначенням, функціонально закінчені пристрої, які за своїми функціями й структурою нагадують спрощений варіант ЕОМ, але мають незрівнянно менші розміри. Такі ВІС одержали назву мікропроцесорів.

Мікропроцесор (МП) – це мікросхема або сукупність невеликого числа мікросхем, що виконує над даними арифметичні й логічні операції і здійснює програмне керування обчислювальним процесом.

Усе різноманіття мікропроцесорів зручно ділити на два істотно різних типи:

1. Однокристальні МП із фіксованою розрядністю слова, з фіксованою системою команд і, як правило, з керуючим пристроєм з «схемною» логікою.

2. Багатокристальні мікропрограмовані МП зі змінюваною розрядністю слова і з фіксованим набором мікрооперацій.

1.1. Архітектура мікропроцесорів

Під *архітектурою мікропроцесора* мають на увазі складові частини мікропроцесора, а також їх взаємне з'єднання і взаємодію між ними. Архітектура включає: 1) структурну схему самого МП; 2) програмну модель МП (опис функцій регістрів); 3) інформацію про організацію пам'яті (місткість пам'яті і способи її адресації); 4) опис організації процедур введення-виведення і керування; 5) опис системи команд та ін.

Існують два основних типи архітектури – *Нейманівська* і *Гарвардська. Нейманівську архітектуру* запропонував в 1945 році американський математик Джо фон Нейман. Особливістю цієї архітектури є те, що програма і дані знаходяться в загальній пам'яті, доступ до яких здійснюється по одній шині даних і команд. Прикладом такої архітектури є класичний мікропроцесор КР580ІК80 (І 8080).

Гарвардська архітектура вперше реалізована в 1944 р. в релейній обчислювальній машині Гарвардського університету (США). Особливістю цієї архітектури є те, що пам'ять даних і пам'ять програм розділені і мають окремі шину даних і шину команд, що дозволяє збільшити швидкодію МП системи за рахунок можливості одночасного звернення по цих двох шинах до пам'яті програм і пам'яті даних. Прикладом такої архітектури є мікроконтролери фірми Intel сімейства MCS51 (К1816ВЕ51) та інші МК багатьох фірм-виготовлювачів.

1.2. Типова структура мікропроцесора

Розглянемо особливості організації процесу оброблення інформації в цифрових пристроях (цифрових автоматах).

Завдання створення цифрового автомата, що виконує певні дії над двійковими сигналами, полягає у виборі елементів і способі їх з'єднання, що забезпечує задане функціональне перетворення. Ці задачі вирішують за допомогою математичної логіки або алгебри логіки.

Пристрої, що формують функції алгебри логіки, називають *логічними, або цифровими,* і класифікують за різними відмітними ознаками.

За схемним рішенням і характером зв'язку між вхідними і вихідними змінними з урахуванням їх зміни за тактами роботи розрізняють два типи цифрових пристроїв – *комбінаційні і цифрові*.

У комбінаційних цифрових пристроях сукупність сигналів на виходах у кожен конкретний момент часу повністю визначається вхідними сигналами, що діють у цей момент на його входах. Алгоритм функціонування цих пристроїв може бути поданий у вигляді таблиці відповідності, що містить значення вихідних сигналів для усіх можливих комбінацій значень вхідних сигналів.

Цифрові пристрої послідовного типу істотно відрізняються від комбінаційних, перш за все, наявністю пам'яті. Їх вхідні сигнали є функцією не тільки вхідних сигналів, але і внутрішнього стану, в якому пристрій знаходився до надходження вхідних сигналів.

На основі цифрового пристрою послідовного типу може бути спроектовано пристрій, який залежно від послідовності вхідних сигналів ви-

конуватиме один з багатьох алгоритмів. Такий пристрій може бути названий пристроєм з програмованою логікою, або програмованим пристроєм. До таких пристроїв належить і мікропроцесор.

Архітектурою мікропроцесора є логічна організація, що визначає можливості апаратної або програмної реалізації функцій, необхідних для побудови мікро-ЕОМ.

Мікропроцесори визначаються такими характеристиками: розрядність адреси і даних, тип корпусу, кількість джерел живлення, потужність розсіяння, температурний діапазон, можливість розширення розрядності, час циклу виконання команд (мікрокоманд), рівні сигналів, перешкодостійкість, здатність навантаження, об'єднання сигналів на виходах, надійність і т. ін.

За числом ВІС в комплекті (МПК) розрізняють однокристальні, багатокристальні, багатокристальні секційні мікропроцесори [8].

Однокристальні мікропроцесори утворюються при реалізації всіх апаратних засобів процесора в одній ВІС. У міру збільшення ступеня інтеграції елементів у кристалі і числа виводів корпусу параметри однокристальних мікропроцесорів поліпшуються. Проте можливості однокристальних мікропроцесорів обмежені апаратурними ресурсами кристала і корпусу. Тому поширеніші багатокристальні і багатокристальні секційні мікропроцесори.

Багатокристальні мікропроцесори отримують при розбитті його логічної структури на функціонально закінчені частини, які реалізують у вигляді ВІС. Функціональна закінченість ВІС багатокристального мікропроцесора означає, що його частини виконують наперед певні функції і можуть працювати автономно, а для побудови розвиненого процесора не потрібна організація великої кількості нових зв'язків і яких-небудь інших інтегральних схем.

Одним з можливих варіантів розбиття структури процесора є створення трикристального мікропроцесора, що містить ВІС операційного процесора, процесора, що керує, і інтерфейсного процесора. Операційний процесор (ОП) служить для оброблення даних, процесор, що керує (КП), виконує функції вибірки, декодування і обчислення адрес операндів, а також генерує послідовності мікрокоманд. Автономність роботи і велика швидкодія ВІС дозволяють вибирати команди з пам'яті з більшою

швидкістю, ніж ВІС операційного процесора. При цьому в КП утворюється черга ще не виконаних команд, наперед готуються ті дані, які будуть потрібні ОП в наступних циклах роботи. Така випереджаюча вибірка команд економить час ОП на очікування операндів, необхідних для виконання команд програм. Інтерфейсний процесор (ІП) дозволяє підключити пам'ять і периферійні засоби до мікропроцесора. Велика інтегральна схема ІП виконує також функції каналу прямого доступу до пам'яті.

Вибрані з пам'яті команди розпізнаються і виконуються кожною частиною мікропроцесора автономно, і тому може бути забезпечений режим одночасної роботи всіх ВІС МП, тобто конвейєрний *потоковий режим виконання послідовності команд програми* (виконання послідовності з невеликим зміщенням у часі). Такий режим роботи значно підвищує продуктивність МП.

Багатокристальні секційні мікропроцесори одержують у тому випадку, коли у вигляді ВІС реалізуються частини (секції) логічної структури процесора. Мікропроцесорна секція – це ВІС, призначена для оброблення декількох розрядів даних або виконання певних керуючих операцій. Багатокристальні секційні МП мають розрядність від 2-4 до 8-16-32-64 біт і дозволяють створювати високопродуктивні процесори ЕОМ.

За призначенням розрізняють *універсальні і спеціалізовані* мікропроцесори. Універсальні мікропроцесори можна застосовувати для вирішення різноманітних задач. Їх ефективна продуктивність мало залежить від проблемної специфіки цих задач. Спеціалізація МП, тобто його проблемна орієнтація на прискорене виконання певних функцій, дозволяє різко збільшити ефективну продуктивність при вирішенні тільки певних задач. Серед спеціалізованих мікропроцесорів можна виділити такі: мікроконтролери, орієнтовані на виконання складних послідовностей логічних операцій; математичні МП, призначені для підвищення продуктивності при виконанні арифметичних операцій за рахунок, наприклад, матричних методів їх виконання; МП для оброблення даних у різних сферах застосування і т.д. За допомогою спеціалізованих МП можна ефективно вирішувати складні задачі паралельного оброблення даних.

За виглядом оброблюваних вхідних сигналів розрізняють *цифрові і аналогові* мікропроцесори. Самі мікропроцесори – це цифрові пристрої,

проте вони можуть мати вбудовані аналого-цифрові і цифро-аналогові перетворювачі. Вхідні аналогові сигнали передаються в МП після перетворення в цифрову форму, обробляються і після зворотного перетворення в аналогову форму поступають на вихід. З погляду архітектури такі мікропроцесори є аналоговими функціональними перетворювачами сигналів і називаються *аналоговими мікропроцесорами.* Вони можуть виконувати функції будь-якої аналогової схеми. Застосування аналогового мікропроцесора значно підвищує точність оброблення аналогових сигналів, а їх відтворення розширює функціональні можливості за рахунок програмного налагодження цифрової частини мікропроцесора на різні алгоритми оброблення сигналів.

Звичайно до складу однокристальних аналогових МП входять декілька каналів аналого-цифрового і цифро-аналогового перетворювачів. У аналоговому мікропроцесорі розрядність оброблюваних даних досягає 24 біт і більше, велике значення приділяється збільшенню швидкості виконання арифметичних операцій.

За характером тимчасової організації роботи розрізняють *синхронні* і *асинхронні* мікропроцесори. *Синхронні мікропроцесори* – це мікропроцесори, в яких початок і кінець виконання операцій задаються пристроєм керування (час виконання операцій в цьому випадку не залежить від виду виконуваних команд і величин операндів). *Асинхронні мікропроцесори* дозволяють початок кожної наступної операції визначити за сигналом фактичного закінчення виконання попередньої операції. Для ефективнішого використання кожного пристрою мікропроцесорної системи до складу асихронно працюючих пристроїв вводять електронні кола, що забезпечують автономне функціонування пристроїв. Закінчивши роботу над якою-небудь операцією, пристрій виробляє сигнал запиту, що означає його готовність до виконання наступної операції. При цьому функції природного розподільника робіт приймає пам’ять, яка відповідно до наперед установленого пріоритету виконує запити решти пристроїв щодо забезпечення їх командною інформацією і даними.

За кількістю виконуваних програм розрізняють *одно- і багатопрограмні* мікропроцесори.

В однопрограмних мікропроцесорах виконується тільки одна програма. Перехід до виконання іншої програми відбувається після завершення поточної програми.

У багато- або мультипрограмних мікропроцесорах одночасно виконуються декілька (звичайно декілька десятків) програм. Організація мультипрограмної роботи мікропроцесорних керуючих систем дозволяє здійснювати контроль за станом і керувати великим числом джерел або приймачів інформації.

Типова структура мікропроцесора наведена на рис. 1.1 [2, 5]. Мікропроцесор складається з трьох основних блоків: арифметико-логічного пристрою (АЛП), блоку внутрішніх регістрів і пристрою керування. Для передачі даних між цими блоками використовується внутрішня шина даних.

Арифметико-логічний пристрій виконує одну з головних функцій мікропроцесора – оброблення даних. Перелік функцій АЛП залежить від типу мікропроцесора. Деякі АЛП здатні виконувати безліч різних операцій, у інших набір операцій обмежений. Функції АЛП визначають архітектуру мікропроцесора в цілому. В більшості мікропроцесорів АЛП виконує такі операції: складання, віднімання, І, АБО, виключаюче АБО, інверсію, зсув вправо, зсув вліво, прирости позитивні і негативні.

Важлива складова частина мікропроцесора – *регістр*. Кожен регістр мікропроцесора можна використовувати для тимчасового зберігання одного слова даних. Деякі регістри мають спеціальне призначення, інші – багатоцільове. Останні називаються *регістрами загального призначення (РЗН)* і можуть використовуватися програмістом на його розсуд. Кількість і призначення регістрів у мікропроцесорі залежать від його архітектури. Розглянемо призначення основних регістрів, наявних майже у всіх мікропроцесорах.

Акумулятор – це головний регістр мікропроцесора. Він використовується при різних маніпуляціях з даними. Більшість арифметичних і логічних операцій здійснюються шляхом використання АЛП і акумулятора. Будь-яка з таких операцій над двома словами даних (операндами) припускає розміщення одного з них в акумуляторі, а іншого – в пам’яті або якому-небудь регістрі. Так, при складанні двох слів, названих умовно А і В і розташованих в акумуляторі та пам’яті відповідно, результуюча сума С завантажується в акумулятор, заміщаючи слово А. Результат виконання операції АЛП теж звичайно розміщується в акумуляторі, вміст якого при цьому втрачається.

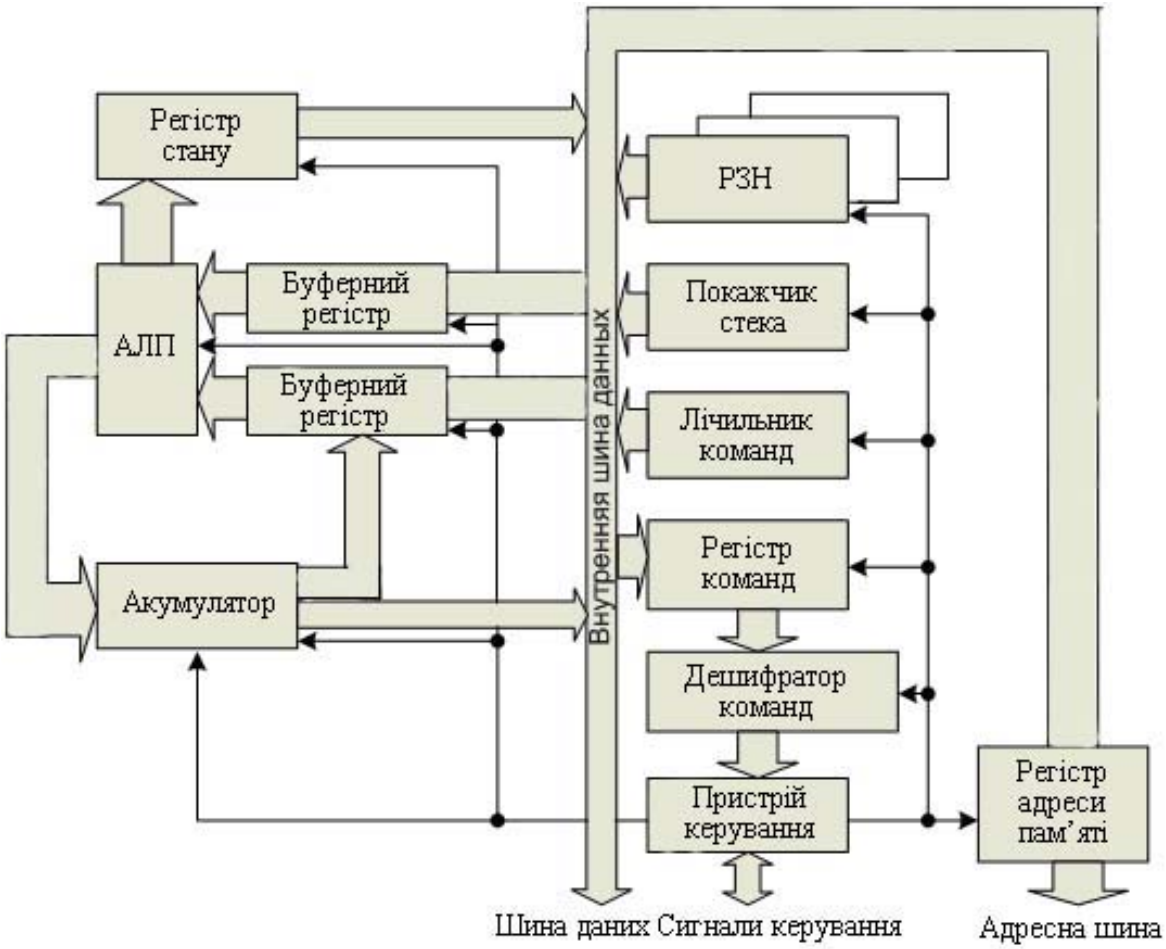


Рисунок 1.1 – Типова структурна схема мікропроцесора

Операцією іншого типу, що використовує акумулятор, є програмована передача даних з однієї частини мікропроцесора в іншу. Наприклад, пересилання даних між портом введення-виведення і пам'яттю, між двома областями пам'яті і т. ін. Виконання операції «програмована передача даних» здійснюється в два етапи: спочатку виконується пересилання даних з джерела в акумулятор, потім – з акумулятора в пункт призначення.

Мікропроцесор може виконувати деякі дії над даними безпосередньо в акумуляторі. Наприклад, акумулятор можна очистити шляхом запису двійкових нулів у всі його розряди, встановити в одиничний стан шляхом запису у всі його розряди двійкових одиниць. Вміст акумулятора можна зсувати вліво або вправо, надавати йому інвертованого значення, обнулювати, а також виконувати інші операції.

Акумулятор є найбільш універсальним регістром мікропроцесора. Для виконання будь-якої операції з даними перш за все необхідно помістити їх в акумулятор. Дані надходять в нього з внутрішньої шини даних мікропроцесора. У свою чергу акумулятор може посилати дані на цю шину.

Кількість розрядів акумулятора відповідає довжині слова мікропроцесора, проте деякі мікропроцесори мають акумулятори подвійної довжини. У додаткові розряди акумулятора записуються при цьому біти, що з'являються при виконанні деяких арифметичних операцій.

Наприклад, при множенні двох 8-бітових слів результат (16-бітове число) розміщується в акумуляторі подвійної довжини.

Лічильник команд – це один з найбільш важливих регістрів мікропроцесора. Як відомо, програма – це послідовність команд (інструкцій), що зберігаються в пам'яті мікро-ЕОМ, і призначених для того, щоб інструктувати машину, як вирішувати поставлену задачу. Для коректного її виконання команди повинні надходити в суворо певному порядку. Лічильник команд забезпечує формування адреси чергової команди, записаної в пам'яті.

Коли мікропроцесор починає працювати, то за командою початкової установки в лічильник команд завантажуються дані з області пам'яті, заданої проектувальником мікропроцесора. Коли програма починає виконуватися, першим значенням вмісту лічильника команд є ця, наперед визначена, адреса.

На відміну від акумулятора, лічильник команд не може виконувати операції різного типу. Набір команд, що його використовують, украй обмежений в порівнянні з подібним набором для акумулятора.

Перед виконанням програми лічильник команд необхідно завантажити адресою, вказуючою на першу команду програми. Адреса першої команди програми посилається по адресній шині до схем керування пам'яттю, внаслідок чого прочитується її вміст за вказаною адресою. Далі ця команда передається в спеціальний регістр мікропроцесора, званий *регістром команд.*

Після витягання команди з пам'яті мікропроцесор автоматично дає приріст вмісту лічильника команд. Цей приріст лічильник команд набу-

ває в той момент, коли мікропроцесор починає виконувати команду, тільки що завантажену з пам'ять. Отже, з цієї миті лічильник команд містить адресу наступної команди.

Лічильник команд можна завантажити іншим вмістом при виконанні осбливої групи команд. Може виникнути необхідність виконати частину програми, яка «випадає» з послідовності команд основної (головної) програми. Наприклад, таку частину програми, яка повторюється в процесі виконання всієї програми. Замість того щоб писати цю частину програми кожного разу, коли в ній виникає необхідність, програму записують один раз і повертаються до її повторного виконання, відступаючи від указаної послідовності. Частина програми, що виконується шляхом відступу від послідовності команд головної програми, називається *підпрограмою.* В даному випадку в лічильник команд безпосередньо записується необхідна адреса. Часто лічильник команд має набагато більше розрядів, ніж довжина слова даних мікропроцесора. Так, у більшості 8-розрядних мікропроцесорів число розрядів лічильника команд дорівнює 16.

Регістр команд містить команду в процесі її дешифрування і виконання. Вхідні дані надходять в регістр з пам'яті в міру послідовної вибірки команд. Звичайно існує можливість запису даних у регістр команд за допомогою набору перемикачів і кнопок на пульті керування ЕОМ. Як правило, цією можливістю користуються для передачі керування в початок програми.

Регістр адреси пам'яті при кожному зверненні до пам'яті мікро-ЕОМ указує адресу ділянки пам'яті, що підлягає використанню мікропроцесором. Регістр адреси пам'яті містить двійкове число-адресу ділянки пам'яті. Вихід цього регістра називається *адресною шиною* і використовується для вибору ділянки пам'яті або порту введення-виведення.

Протягом вибірки команди з пам'яті регістри адреси пам'яті і лічильника команд мають однаковий вміст, тобто регістр адреси пам'яті вказує місцеположення команди, витягуваної з пам'яті.

Після декодування команди лічильник команд одержує приріст на відміну від регістра адреси пам'яті.

У процесі виконання команди вміст регістру адреси пам'яті залежить від виконуваної команди. Якщо відповідно до команди мікропро-

цесор повинен провести ще одне звернення до пам'яті, то регістр адреси пам'яті підлягає вторинному використанню в процесі оброблення цієї команди. Для деяких команд, наприклад, команди очищення акумулятора, адресація до пам'яті не потрібна. При обробці таких команд регістр адреси пам'яті використовується лише один раз – протягом вибірки команди з пам'яті.

У більшості мікропроцесорів регістри адреси пам'яті і лічильника команд мають однакову кількість розрядів. Як і лічильник команд, регістр адреси пам'яті повинен мати в своєму розпорядженні кількість розрядів, достатню для адресації будь-якої ділянки пам'яті мікро-ЕОМ. У більшості 8-розрядних мікропроцесорів кількість розрядів регістру адреси пам'яті дорівнює 16.

Оскільки регістр адреси пам'яті підключений до внутрішньої шини даних мікропроцесора, він може завантажуватися від різних джерел. Більшість мікропроцесорів має в своєму розпорядженні команди, що дозволяють завантажувати цей регістр вмістом лічильника команд, регістру загального призначення або якої-небудь ділянки пам'яті. Деякі команди надають можливість змінювати вміст регістру адреси пам'яті шляхом виконання обчислень: нове значення вмісту цього регістру виходить шляхом складання або віднімання вмісту лічильника команд з числом, вказаним в самій команді. Адресація такого типу називається *адресацією з використанням зсуву.*

Буферний регістр – призначений для тимчасового зберігання (буферування) даних.

Регістр стану – призначений для зберігання результатів деяких перевірок, здійснюваних у процесі виконання програми. Розряди регістру стану приймають те або інше значення при виконанні операцій, що використовують АЛП і деякі регістри. Запам'ятовування результатів згаданих перевірок дозволяє використовувати програми, що містять переходи (порушення природної послідовності виконання команд).

За наявності в програмі переходу за заданою ознакою виконання команд починається з деякої нової ділянки пам'яті, тобто лічильник команд завантажується новим числом. У разі умовного переходу така дія має місце, якщо результати певних перевірок збігаються з очікуваними значеннями. Вказані результати знаходяться в регістрі стану.

Регістр стану надає програмісту можливість організувати роботу мікропроцесора так, щоб за певних умов змінювався порядок виконання команд.

Розглянемо деякі найбільш часто використовувані розряди регістра стану.

1. *Перенесення/позика*. Даний розряд указує, що остання виконана операція супроводжувалася перенесенням або позикою (негативним перенесенням). Значення розряду перенесення встановлюється рівним 1, якщо в результаті складання двох чисел має місце перенесення із старшого розряду АЛП. Негативне перенесення (позика) фіксується в регістрі стану при відніманні більшого числа від меншого.

2. *Нульовий результат*. Набуває одиничного значення, якщо після закінчення операції у всіх розрядах регістра результату виявлені двійкові нулі. Установка цього розряду в 1 відбувається не тільки при негативному прирості вмісту регістру, але і при будь-якій іншій операції, результат якої – число з двійкових нулів.

3. *Знаковий*. Набуває одиничне значення, коли старший значущий біт вмісту регістра, призначеного для запису результату операції, стає рівним 1. При виконанні арифметичних операцій з числами в додатковому коді одиничне значення старшого значущого біта показує, що в регістрі знаходиться від'ємне число.

Багато мікропроцесорів мають у своєму розпорядженні додаткові розряди станів. У деяких передбачені спеціальні команди для скидання або очищення всіх розрядів стану.

Регістри загального призначення (РЗП). Більшість МП має в своєму складі набір регістрів, використовуваних як запам'ятовуючі пристрої. Оскільки АЛП може здійснювати операції з вмістом РЗП без виходу на зовнішню магістраль адрес і даних, то вони відбуваються набагато швидше, ніж операції із зовнішньою пам'яттю. Тому іноді РЗП *називають надоперативною пам'яттю.* Кількість РЗП і можливості програмного доступу до них у різних мікропроцесорів різні.

Указівник стека. *Стек* – це набір регістрів мікропроцесора або елементів оперативної пам'яті, звідки дані або адреси вибираються «зверху» за принципом: перший, що надійшов останнім. Указані процедури ілюструє рис. 1.2.

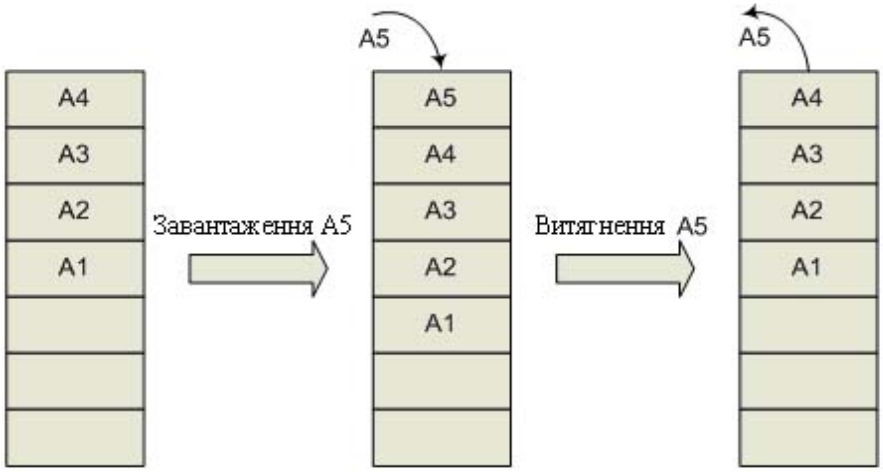


Рисунок 1.2 – Процедури роботи стека

При записі в стек чергового слова всі раніш записані слова зміщуються на один регістр униз. При вибірці слова зі стека слова, що залишилися, переміщаються вгору на один регістр. Тут стек складається з семи регістрів. Якщо в стек завантажується яке-небудь слово, наприклад A5, то воно записується у верхньому регістрі, а кожне зі слів A1…A4 переміщається в сусідні нижні регістри. Якщо ж A5 витягується зі стека, то кожне зі слів A1..A4 переміщається в сусідні верхні регістри. Не можна витягнути A4 раніше A5, тобто автоматично реалізується відзначений вище принцип.

Стек звичайно використовується в мікропроцесорах для зберігання адрес повернення при зверненні до підпрограм, а також для запам'ятовування стану внутрішніх регістрів при обробці переривань.

При організації стека в пам'яті час на звернення до нього буде дорівнювати циклу звернення до пам'яті.

Ця операція виконується значно швидше, якщо стек у вигляді набору регістрів входить до складу мікропроцесора.

Важливим параметром у такому разі є число регістрів стека. При спробі записати в стек більшої кількості слів, ніж число його регістрів, перше слово буде втрачено. У деяких мікропроцесорах при переповнюванні регістрів стека відповідні слова записуються в стек пам'яті.

Часто стек реалізується таким чином, що процес його функціонування нагадує роботу з пачкою документів, коли кожен новий документ кладеться зверху пачки. При такій організації стека необхідний спеціаль-

ний регістр – указівник стека (РУС) для зберігання адреси останнього за часом надходження елемента стека.

Наведений на рис. 1.3 указівник стека є трирозрядний регістр з двійковим поданням інформації. Спочатку указівник стека містить число 011_2. Це означає, що останній елемент – «верхівка стека» знаходиться в регістрі з адресою 011_2 (або 3_{10}). При операції завантаження в регістр 4_{10} записується число А5, а вміст указівника стека змінюється так, що він указує на регістр 4_{10}. При операції витягування із стека проводяться зворотні дії.

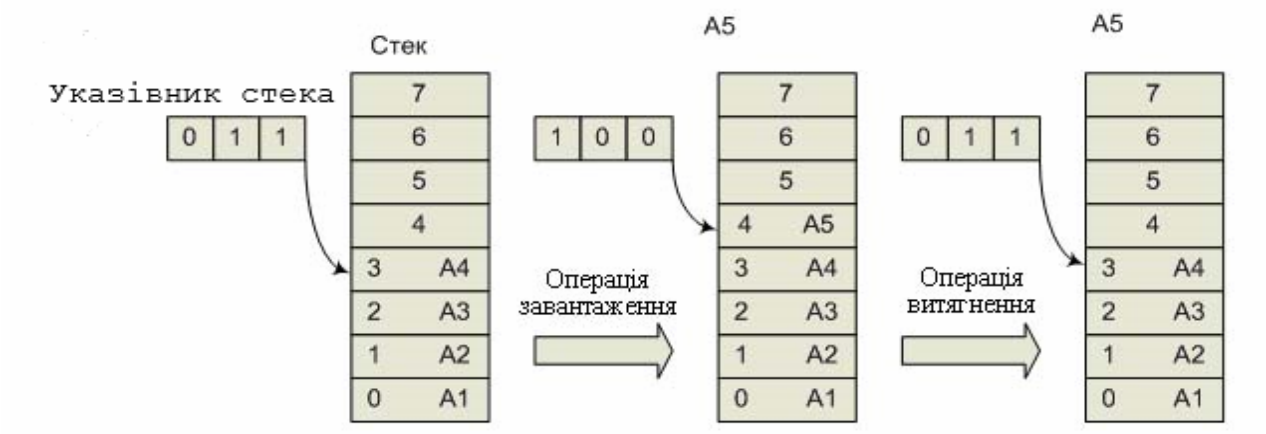


Рисунок 1.3 – Адресація елемента стека з використанням указівника стека

Схеми керування. Роль схем керування в мікропроцесорі полягає в підтримці необхідної послідовності функціонування всієї решти його ланок.

За сигналами схем керування чергова команда витягується з регістру команд. При цьому визначається, що необхідно робити з даними, а потім забезпечується послідовність дій для виконання поставленого завдання.

Одна з головних функцій схем керування – декодування команди, що знаходиться в регістрі команд, за допомогою дешифратора команд, який в результаті видає сигнали, необхідні для її виконання.

Крім вказаних вище дій, схеми керування виконують деякі спеціальні функції керування послідовністю включення живлення і процесами переривань.

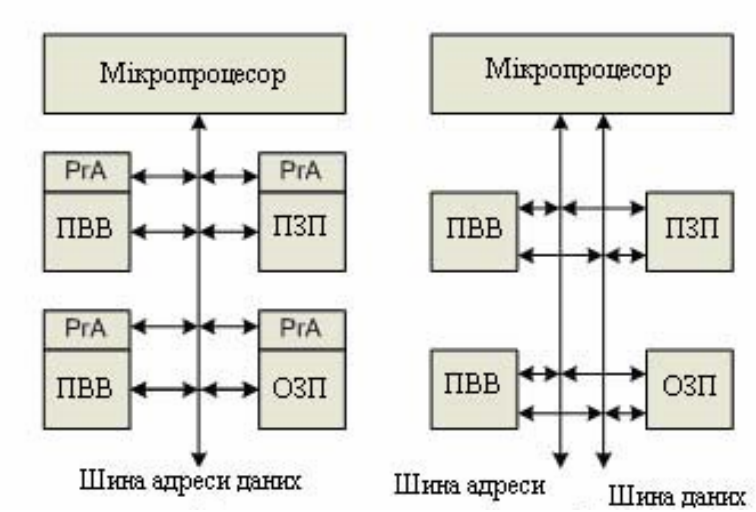
Переривання – це свого роду запит, що надходить на схеми керування з інших пристроїв (пам’яті, введення-виведення). Переривання

пов’язане з використанням внутрішньої шини даних мікропроцесора. Схеми керування приймають рішення, коли і в якій послідовності інші пристрої можуть користуватися внутрішньою шиною даних.

Система шин. На характеристики мікропроцесора впливає спосіб організації його зв’язку із зовнішнім середовищем – пристроями введення-виведення (ПВВ) і пристроями, що запам’ятовують. За способом організації зв’язків із зовнішнім середовищем розрізняють мікропроцесори з мультиплексованою шиною адреси і даних (рис. 1.4, *а*) і з роздільними шинами адреси і даних (рис. 1.4, *б*) [12]. Мікропроцесор з роздільними шинами адрес і даних зображений на рис. 1.1.

У мікропроцесорах з мультиплексованою шиною адреса зберігається на шині тільки короткий проміжок часу, тому пристроям, підключеним до шини, потрібні регістри адреси (РГА). Для організації обміну інформацією в таких мікропроцесорах необхідно використовувати керуючий сигнал «дані-адреси». При роздільних шинах адреси і даних керуючий сигнал не потрібен.

Крім того, у пристроїв, підключених до шин, відпадає необхідність в регістрі адреси, оскільки він може бути розміщений безпосередньо на кристалі мікропроцесора. Розрядність адресної шини в таких мікропроцесорах не пов’язана з розрядністю шини даних. Характерним прикладом МП з роздільними шинами адрес і даних є мікропроцесорний комплект КР580, а з мультиплексованою шиною адреси і даних – мікропроцесорний комплект К588.



а *б*

Рисунок 1.4 – Система шин мікро-ЕОМ

Терміни і визначення, арифметичні, логічні і фізичні основи, архітектура, структурні схеми мікропроцесорів і комплектів КР580, К588 та ін., запамятовючі прстрої, інтерфейси і таймери, їх будова та програмування більш детально викладені в [1–19].

1.3. Мікроконтролер «Електроніка МС2702»

Одним із перших вітчизняних, мікроконтролерів, що вироблялись на заводі в місті Борзна (Українська РСР), є багатокристальний мікроконтролер «Електроніка МС2702» (в подальшому МС2702). Він побудований на МП комплекті К580 і виконаний на окремих мікросхемах. У його склад входять: центральний процесор КР580ВМ80А (нейманівська архітектура); тактовий генератор К580ГФ24; системний контролер К589ІР12; пристрої пам’яті; пристрої послідовного введення-виведення КР580ВВ51; пристрій паралельного введення-виведення КР580ВВ55; таймер КР580ВІ53; контролер переривання КР580ВН59; пульт керування, що складається з 9-сегментного індикатора (дисплея) та блоку клавіатури з літерно-цифровими операційними клавішами [3].

Система комад мікроконтролера МС2702 (така як і у МП КР580) наведена в додату 1, а будова і приклади програмування інтерфейса, таймера та застосування мікроконтролера МС2702 в додатку 2, завдання 1–3, відповідно.

Пам’ять МС2702 має байтову структуру, з можливою адресацією до будь-якого її байта. При звертанні до пам’яті використовуються 16-розрядні (двобайтні) адреси. МП виконує логічні й арифметичні операції.

При розгляді структури МП можна виділити наступні її частини: блок регістрів, арифметико-логічний пристрій (АЛП), буферні схеми, керуючий і синхронізуючий пристрій.

Організація КР580 відзначена такими основними особливостями:

- тришинною структурою із шинами даних, адреси й керування;
- магістральним принципом зв’язків, реалізованим у вигляді з’єднаних двонаправленою шиною даних основних вузлів МП;

- наявністю регістрової пам’яті, утвореної програмно-доступними загальними й спеціалізованими регістрами (лічильник команд, указівник стека, указівник даних), а також регістрами тимчасового зберігання;
- наявністю засобів організації стекової пам’яті (регістр – указівник стека, схеми виконання операцій, спеціальних команд стекових операцій);
- наявністю 16-розрядної шини адреси, що забезпечує можливість адресації будь-якого байта в пам’яті ємністю 64 Кбайта;
- наявністю операцій над двобайтовими словами (16-розрядними числами, адресами), що забезпечуються спеціальними командами; використанням трьох форматів команд (однобайтового, двобайтового й трибайтового) і різноманітних способів адресації (мається на увазі, пряма, регістрова, непряма, стекова, безпосередня); можливістю реалізації в МП режиму прямого доступу до пам’яті шляхом підключення спеціальної додаткової ВІС;
- наявністю ефективних засобів роботи з підпрограмами й оброблення запитів переривань (стекова пам’ять, спеціальні команди виклику підпрограм і повернення з підпрограм, у тому числі й умовного).

Структура мікропроцесора КР580. МП має 3 шини: 8-розрядну двонаправлену внутрішню шину даних (ШД), 16-розрядну адресну шину (ША) і шину керування (ШК). Внутрішня шина даних є магістраллю, по якій можуть обмінюватися даними всі підключені до неї блоки (вузли) МП. Одночасно по шині даних здійснюється обмін тільки між двома вузлами МП. Таким чином, вузли МП, приєднані до шини даних, розділяють цю шину в часі.

Шина керування містить лінії для передачі керуючих сигналів, ознак стану процесора й периферійних пристроїв, у тому числі лінії: синхронізації передачі й ідентифікації інформації, переданої по шині даних; сигналів, що інформують мікропроцесор про готовність периферійних пристроїв; сигналу запиту переривання від периферійних пристроїв і сигналу дозволу переривання і т. ін.

Будова, схеми і функціонування МП КР580, інтерфейсу, таймера і їх програмування та застосування мікроконтролера МС2702 наведені в додатку 2 і в [3].

Контрольні запитання і завдання

1. Наведіть типову структурну схему МП і призначення її елементів.
2. Викладіть будову, структурну схему і функціонування МП КР580.
3. Викладіть призначення і принцип дії акумулятора та АЛП.
4. Викладіть призначення і функціонування програмного лічильника.
5. Що міститься в регістрі адреси пам'яті?
6. Назвіть ознаки регістра стану і їх призначення.
7. Яка команда міститься в регістрі команд у процесі виконання поточної команди і коли змінюється адреса в указівнику стека?
8. Наведіть принцип дії стека і призначення указівника стека.
9. Яке призначення буферних регістрів?
10. Викладіть призначення схем керування і синхронізації МП.
11. Протягом якого часу МП витягує команду з пам'яті і виконує її?
12. Що забезпечує таймер у схемах керування МП?
13. Куди поміщається результат виконання операції АЛП?
14. Дайте характеристику регістрів загального призначення (РЗН) МП.
15. Дайте визначення архітектури МП.
16. Наведіть відомі архітектури МП і вкажіть їх відмінності.
17. Наведіть відмінність акумулятора від інших регістрів.
18. Назвіть функції, що виконуються пристроєм керування.
19. Що визначає вміст лічильника команд і як він змінюється?
20. Що таке набір команд МП?
21. Якою може бути довжина команди 8-розрядного МП?
22. З яких двох частин складається команда і яке їх призначення?
23. Дайте характеристики різних типів адресації.
24. Який спосіб адресації має найбільшу кількість мікроциклів?
25. Наведіть приклади застосування арифметичних і логічних команд.
26. Назвіть команди пересилання даних і дайте характеристику команд завантаження, пересилання і запису в пам'ять.
27. Назвіть команди переходу і виклику підпрограм і поясніть їх роботу.
28. Назвіть команди введення-виведення і роботи зі стеком.
29. В яких випадках і де встановлюється та скидається біт перенесення?

30. Назвіть команди умовних переходів.

31. Назвіть команди пересилання даних з акумулятора в ОЗП і навпаки.

32. Складіть програму складання двох чисел і запису результату в ОЗП.

33. Складіть програму віднімання двох чисел і запису результату в регістр *D* і елемент пам'яті з адресою 8020.

34. Складіть програму виконання операцій АБО, І, ВИКЛЮЧАЮЧЕ АБО над двома числами і занесення результатів у пам'ять.

35. Назвіть команди безпосереднього завантаження регістрових пар і стека та наведіть приклади їх застосування.

36. Назвіть команди обміну двох байтів між регістрами *HL* і *DE*.

37. Поясніть роботу команд безумовного і умовного повернень з підпрограм та наведіть приклади.

38. Поясніть роботу команд дозволу і заборони переривань.

39. Призначення команд *HLT, NOP, CMC, CMA, CPI* .

40. Назвіть функції, що виконуються командами *OUT* і *IN*.

41. Назвіть команди пересилання даних з пам'яті в акумулятор.

42. Поясніть роботу команди повернення з підпрограм *RET*.

43. Викладіть роботу стека при виконанні команди виклику підпрограм *CALL* і повернення з них та наведіть приклади.

44. Що таке розрядність мікропроцесора і як вона визначається?

45. Що знаходиться в першому байті трибайтової команди?

46. Скільки розрядів має шина даних МК МС2702 і яке її призначення?

47. Із скількох ліній складається шина керування, які сигнали і як на ній формуються?

48. Дайте характеристику шини адреси і вкажіть її призначення.

49. Які функції виконує регістр команд РК і дешифратор команд (ДШК)?

50. В яких випадках і де встановлюються та скидаються біти перенесення, додаткового перенесення, ознаки нульового результату, знаку та паритету?

51. Наведіть будову мікроконтролера МС2702.

52. Викладіть призначення, будову, режими роботи і приклади програмування інтерфейса, таймера та наведіть приклади застосування мікроконтролера МС2702 в автоматизованих системах керування.

2. ОДНОКРИСТАЛЬНІ МІКРОКОНТРОЛЕРИ

Однокристальні мікроконтролери (ОМК) – однокристальні мікроЕОМ (ОМЕОМ), (вони ж мікоконтролери (МК)) призначені для керування електротехнічними, електроенергетичними і іншими об'єктами або пристроями, а також технологічними процесами виробництва, випробувань і досліджень. Вони є пристроями, конструктивно виконаними в одному корпусі ВІС, яка містить всі складові: процесор, резидентну пам'ять даних і програм, програмовані інтерфейси. Високоінтегровані ОМК можуть мати АЦП, ЦАП, внутрішні температурні сенсори, послідовні порти UART, таймери-лічильники та інші пристрої.

2.1. Мікроконтролери фірми Inte1

Восьмирозрядні високопродуктивні однокристальні мікроконтролери (ОМЕОМ) сімейства MCS-51 гарвардської архітектури виконані за високоякісною *n*–МОП технологією (серія 1816) і КМОП технологією (серія 1830) [20]. Сімейство MCS51 забезпечує збільшення обсягу пам'яті команд і пам'яті даних, нові можливості введення-виведення і периферійних пристроїв, розширює діапазон застосування і знижує загальні витрати системи. ОМЕОМ КР1816ВЕ51 і КР1830ВЕ51 містять програмовані в процесі виготовлення кристала ПЗП пам'яті програм місткістю 4096 байт і розраховані на застосування в масовій продукції. За рахунок використання зовнішніх мікросхем пам'яті загальний обсяг пам'яті програм може бути розширений до 64 Кбайт. ОМЕОМ КР1816ВЕ31 і КР1830ВЕ31 не містять вбудованої пам'яті програм, проте можуть використовувати до 64 Кбайт зовнішньої постійної або перепрограмованої пам'яті програм і ефективно застосовуватися в схемах, що вимагають істотно більшого за обсягом (ніж 4 Кбайт на кристалі) ПЗП пам'яті програм. Кожна з перелічених вище мікросхем є відповідним аналогом ВІС 8051, 80С51, 8751, 8031, 80С31 сімейства MCS-51 фірми Inte1 (США). Кожна ОМЕОМ даного сімейства містить вбудоване ОЗП пам'яті даних місткістю 128 байт з можливістю розширення загального обсягу оперативної пам'яті

даних до 64 Кбайт за рахунок використання зовнішніх мікросхем ЗПД. Порівняльні дані мікросхем наведені в табл. 2.1.

Таблиця 2.1 – Порівняльні дані мікросхем

Мікросхема	Аналог	Обсяг внутрішньої пам'яті програм, Кбайт	Тип пам'яті програм	Обсяг внутрішньої пам'яті даних, байт	Максимальна частота проходження тактових сигналів, МГц	Струм споживання, мА
КР1816ВЕ31	8031 АН	–	зовнішн.	128	12,0	150,0
КР1816ВЕ51	8051 АН	4	ПЗП	128	12,0	150,0
КМ1816ВЕ751	8751Н	4	ППЗУ	128	12,0	220,0
КР1830ВЕ31	80С31ВН	–	зовнішн.	128	12,0	18,0
КР1830ВЕ51	80С51ВН	4	ПЗП	128	12,0	18,0

Однокристальний 8-розрядний мікроконтролер МК1816ВЕ51 (МК51) виконаний на основі гарвардської архітектури і високорівневої *n*-МОП технології, що дозволяє одержати високий ступінь інтеграції, порівняно з біполярними структурами, і забезпечує підвищену швидкодію. Докладний опис архітектури МК51 наведений в п. 2.2 [20]. Основні характеристики і відмінні особливості мікроконтролерів сімейства MCS51 (MCS-51, MCS-151 і MCS-251) наведені в табл. 2.2 [20]. Мікроконтролери сімейства 8XC151SX (MCS-151) за системою команд, набору програмнодоступних ресурсів, системою переривань, набору блоків введення-виведення і функціями виведень корпусу сумісні з мікроконтролерами 8XC51FX (MCS-51). Удосконалення торкнулися в основному операційного ядра. Введені конвеєр команд, режим сторінкової адресації пам'яті та ін. У результаті при конвеєрній вибірці в межах однієї сторінки час виконання команди складає два періоди частоти задаючого кварцевого генератора (замість 12 періодів у попереднього сімейства MCS-51). Мікроконтролери сімейства MCS-251 є розвитком архітектури сімейств MCS-51 і MCS-151.

Таблиця 2.2 – Характеристика мікроконтролерів сімейства MCS-51

Контролер	ROM/EPROM Кбайт	RAM, байт	T/C	Макс.Fosc, Мгц	Особливості групи
8031AH	–	128	2	12	HMOS технологія, базова конфігурація, 4 порти
8051AH	4K ROM	128	2	12	–
8751H	4K EPROM	128	2	12	–
80C31BH	–	128	2	12,16	CHMOS технологія, режим зниженого енергоспоживання, 3 біти захисту
80C51BH	4K ROM	128	2	12,16	–
87C51BH	4K EPROM	128	2	12,16,20	–
8032AH	–	256	3	12	HMOS технологія, 3 таймери/лічильники, 4 порти, 3 біти захисту
8052AH	8K ROM	256	3	12	–
8752BH	8K EPROM	256	3	12	–
80C32	–	256	3	16,20,24	CHMOS технологія, таймер/лічильник з прямим і зворотним рахунком, 3 біти захисту
80C52	8K ROM	256	3	16,20,24	–
87C52	8K EPROM	256	3	16,20,24	–
80C54	8K ROM	256	3	16,20,24	–
87C54	16K EPROM	256	3	16,20,24	–
80C58	32K ROM	256	3	16,20,24	–
87C58	32K EPROM	256	3	16,20,24	–
80L52	8K ROM	256	3	16,20	Версія зі зниженою напругою живлення 2,7–3,6 В
87L52	8K EPROM	256	3	16,20	–
80L54	8K ROM	256	3	16,20	–
87L54	16K EPROM	256	3	16,20	–
80L58	32K ROM	256	3	16,20	–
87L58	32K EPROM	256	3	16,20	–
80C51FA	–	256	3	16,20,24	CHMOS технологія, модуль PCA, таймер/лічильник з прямим і зворотним рахунком, режими зниженого енергоспоживання, сторожовий таймер, 3 біти захисту
83C51FA	8K ROM	256	3	16,20,24	–
87C51FA	8K EPROM	256	3	16,20,24	–

Закінчення табл. 2.2

Контролер	ROM/EPROM Кбайт	RAM, байт	Т/С	Макс.Fosc, Мгц	Особливості групи
83C51FB	16K ROM	256	3	16,20,24	–
87C51FB	16K EPROM	256	3	16,20,24	–
83C51FC	32K ROM	256	3	16,20,24	–
87C51FC	32K EPROM	256	3	16,20,24	–
80L51FA	–	256	3	16,20	CHMOS технологія, версія зі зниженою напругою живлення 2,7– 6 В
83L51FA	8K ROM	256	3	16,20	–
87L51FA	8K EPROM	256	3	12,16	–
83L51FB	16K ROM	256	3	16,20	–
87L51FB	16K EPROM	256	3	16,20	–
83L51FC	32K ROM	256	3	16,20	–
87L51FC	32K EPROM	256	3	16,20	–
80C51GB	–	256	3	16,20,24	CHMOS технологія, АЦП (8 кан/8 розряд), 2 PCA, 6 портів I/O, сторожовий таймер
83C51GB	8K ROM	256	3	16,20,24	–
87C51GB	8K EPROM	256	3	16,20,24	–

В основу покладена «стара» система команд і сталий набір блоків введення-виведення: три таймери-лічильники, блок PCA, послідовний порт і сторожовий таймер. Центральний процесор мікроконтролерів MCS-251 побудований з використанням конвейєра команд (час виконання команд – 2 періоди частоти кварцевого генератора) і регістрового файлу. Система команд доповнена інструкціями, оперуючими 16-ти і 32-розрядними операндами. Нове сімейство 8-бітових мікроконтролерів підвищує функціональність і продуктивність широко розповсюджених мікроконтролерів MSC-51 при збереженні сумісності на рівні двійкових кодів. Завдяки сумісності по контактах з 8XC51FX, МК 8XC251SB може служити засобом підвищення продуктивності існуючих апаратно-програмних систем. До типових областей застосування 8XC251SB можна віднести системи керування. Усім МК сімейства MSC-251 властиві такі загальні особливості:

- 24-бітова лінійна адресація до 16 Мбайт пам’яті;
- ЦПП регістрової архітектури з регістрами, що адресуються як байти, слова і подвійні слова;

- сторінковий режим, прискорюючий вибірку команд із зовнішньої пам’яті;
- конвеєр команд;
- розширена система команд, що включає 16-бітові арифметичні та логічні команди;
- 64 – Кбайтовий зовнішній стек;
- мінімальний час виконання команд (два такти порівняно з 12 тактами у МК MCS-51);
- двійкова сумісність з МК MCS-51.

Першим МК у сімействі MCS-251, розробленим компанією Intel, є мікроконтролер 8XC251SB. Опис його архітектури детальніше наведений в п. 4.4.

2.2. Мікроконтролер сімейства MCS-51 КМ1816ВЕ51 (МК51)

Однокристальний мікроконтролер КМ1816ВЕ51 (МК51) [20] виконаний у корпусі ВІС на основі високорівневої *n*-МОП технології, що дозволяє одержати вищий ступінь інтеграції в порівнянні з біполярними структурами. Корпус ВІС має 40 зовнішніх виводів. Цоколівка корпусу і назва виводів наведені на рис. 2.1. Для роботи потрібне одне джерело живлення +5 В. Через чотири програмовані порти введення-виведення МК51 взаємодіє з середовищем у стандарті ТТЛ-схем з трьома станами виходу.

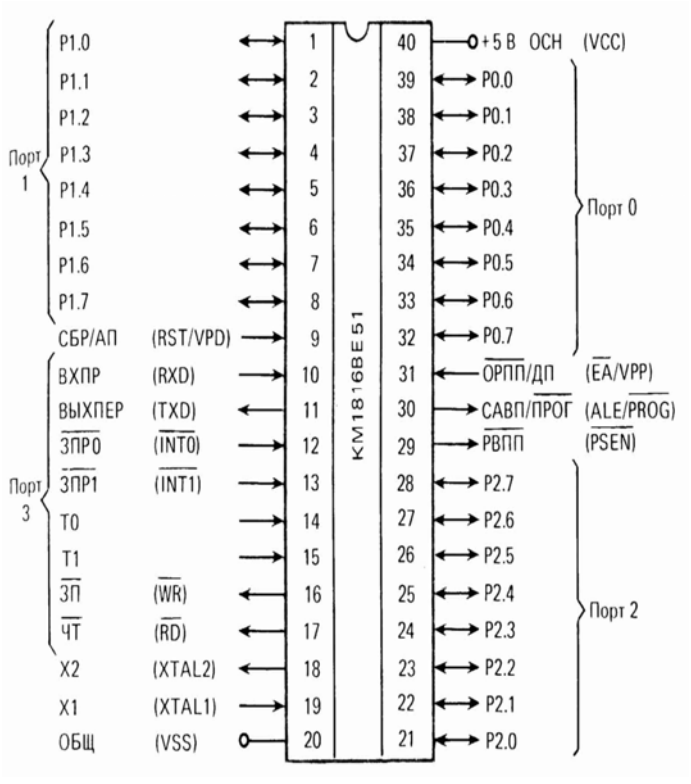


Рисунок 2.1– Корпус мікросхеми МК51 і найменування виводів

Корпус МК51 має два виводи для підключення кварцевого резонатора, чотири виводи для сигналів, керуючих режимом роботи МК, і 8 ліній порту 3, які можуть бути запрограмовані користувачем на виконання спеціалізованих функцій обміну інформацією з середовищем. Назва виводів і значення вхідних і вихідних рівнів такі:

ТТЛ: Uin ≥ 2,0В Uil ≤ 0,8 В – вхідні рівні;

Uoh ≥ 2,4В Uol ≤ 0,4В – вихідні рівні;

СБР (RST) – керуючий сигнал скидання;

ВхПр(RxD)–вхід приймача універсального приймача-передавача (УАПП);

ВихПер(TxD) – вихід передавача УАПП;

ЗПр(INT) – запит переривання;

Т0, Т1– таймер/лічильник подій;

Х1, Х2 – входи кварцевого резонатора;

ЗП(WR) – керуючий сигнал запису;

ЧТ(RD) – керуючий сигнал читання;

ОРПП(EA) – керуючий сигнал відключення резидентної пам’яті програм;

САВП(ALE) – керуючий сигнал скидання адреси зовнішньої пам’яті;

ПРОГ(PROG) – керуючий сигнал програмування РПП;

ДЗПП(PSEN) – дозволи зовнішньої пам’яті програм;

ОБЩ(VSS) – потенціал землі;

+5В(VCC) – напруга живлення + 5В;

Х1(XTAL1) – вхід для підключення виводу кварцового резонатора або вхід для сигналу від зовнішньої початкової сигналізації;

Х2(XAL2) вхід для підключення другого виводу резонатора.

Структурна схема МК51

Узагальнена структурна схема наведена на рис. 2.2, а детальніша – на рис. 3.8. Її основу становить внутрішня двонаправлена 8-бітова шина, яка зв’язує між собою основні вузли і пристрої. До них належать: 8-розрядний центральний процесор (ЦП) (включає АЛП, акумулятор з розширювачем, регістр слова стану, блок регістрів спеціальних функцій, пристрій керування і синхронізації), розміщені на кристалі (всередині); пам’ять програм – ПЗП місткістю 4 Кбайт і пам’ять даних – ОЗП міст-

кістю 128 байт, які становлять резидентну пам’ять; 32 лінії чотирьох паралельних портів (P0–P3) введення-виведення (ВВ); послідовний ВВ УАПП; два 16-розрядні таймери/лічильники і логіку дворівневої системи переривань з п’ятьма або шістьма джерелами запитів.

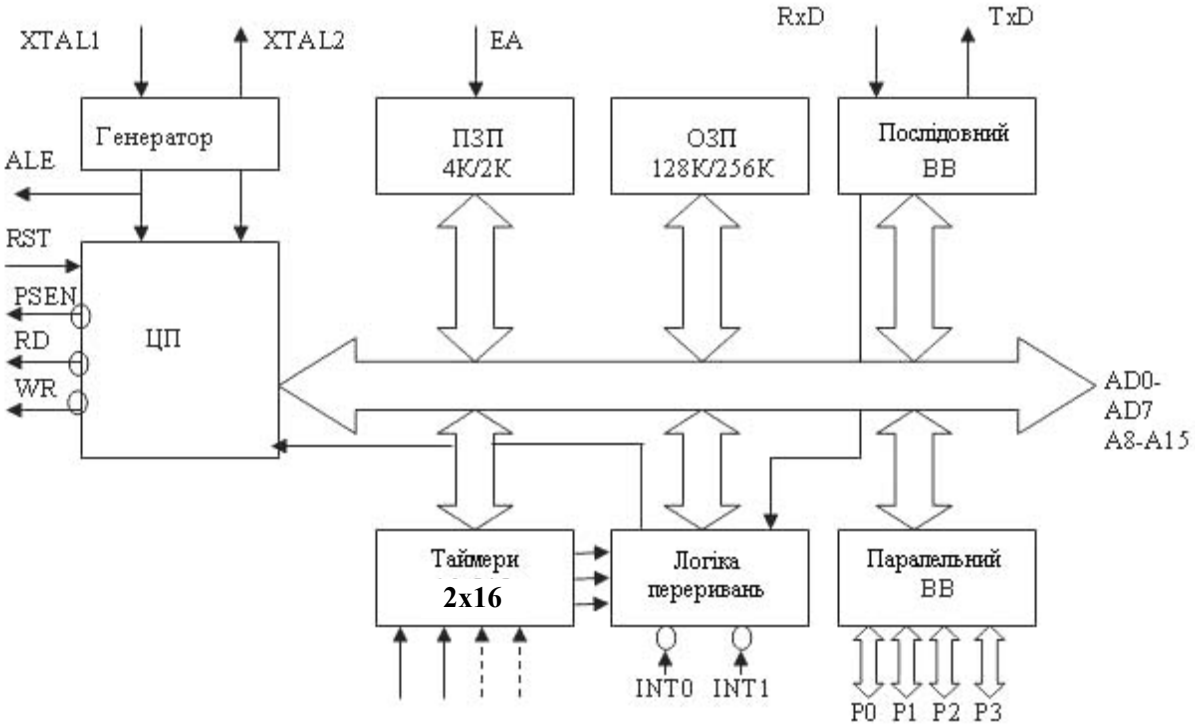


Рисунок 2.2 – Узагальнена структурна схема МК51

Усі ці засоби утворюють резидентну частину МК, розміщену безпосередньо на кристалі. Окрім цього, є можливість реалізувати поза кристалом пам’ять програм і пам’ять даних до 64 Кбайт кожну шляхом підключення зовнішніх ВІС. Для скорочення ширини фізичного інтерфейсу більшість логічних ліній поєднуються. Так, при зверненнях до зовнішньої пам’яті порт P0 виконує роль суміщеної шини адреси даних, а P2 – шини старшої частини адреси. Усі лінії порту P3 можуть виконувати альтернативні функції ліній керування.

В архітектурі МК51 використаний принцип незалежності середовищ для зберігання програм і даних, тобто принцип гарвардської архітектури.

Резидентна пам'ять. Пам'ять програм і пам'ять даних, розміщені на кристалі BE51, фізично і логічно розділені, мають різні механізми адресації, працюють під керуванням різних сигналів і виконують різні функції.

Пам'ять програм (ПЗП або СППЗУ) має місткість 4 Кбайт і призначена для зберігання команд, констант, слів ініціалізації, таблиць перекодування вхідних змінних і т.п.; керована РПП має 16-бітову шину адреси, через яку забезпечується доступ з лічильника команд або з регістра указівника даних (РУД).

Пам'ять даних (ОЗП) призначена для зберігання змінних у процесі виконання прикладної програми, адресується одним байтом і має місткість 128 байт. До адресного простору РПД примикають адреси регістрів спеціальних функцій (РСФ), РПП, РПД. Організація доступу до них подана на рис. 2.3. Набір програмно доступних регістрів наведений на рис. 2.4. Оскільки структура належить до класу акумуляторних, з банками перемикачів робочих регістрів, то центральним регістром набору вважається акумулятор.

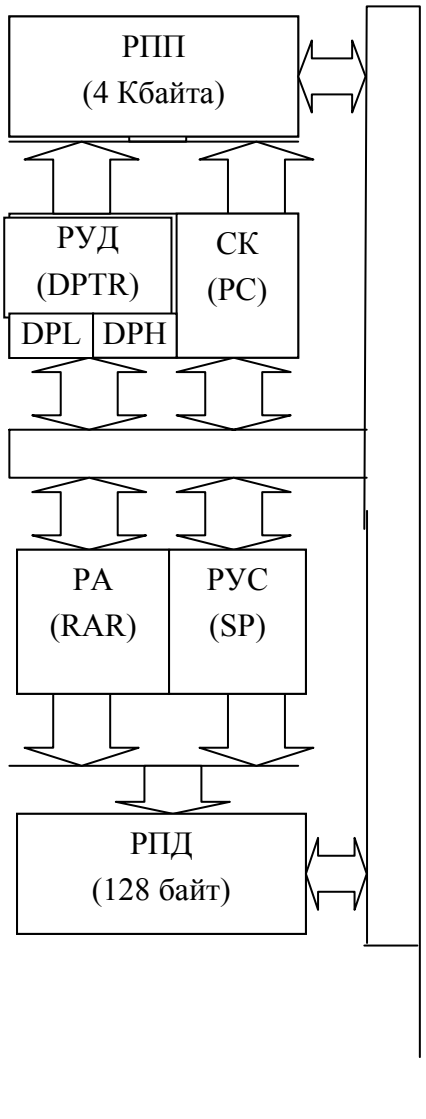
Акумулятор А – 8-розрядний регістр, виконує звичайні функції основного арифметичного регістра, є джерелом операнда і приймачем (місцем фіксації) результату арифметичних і логічних операцій, а також ряду операцій передачі даних. Тільки з використанням акумулятора - можуть бути виконані операції зсуву, перевірки на нуль, з прапором паритету та ін.

Регістр В – 8-розрядний, служить розширенням акумулятора А, необхідний для здійснення операцій множення і ділення, де виступає як джерело і приймач операндів. У всіх інших операціях регістр В виконує функції, які визначені користувачем. При скиданні А і В встановлюють в нуль.

Контролер МК51 може виконувати безліч команд без участі акумулятора. Дані можуть бути передані з будь-якої комірки РПД в будь-який регістр, останній може бути завантажений безпосередньо операндом і т. д.

Ряд логічних операцій може бути виконаний без участі акумулятора. Змінні можуть бути інтерпретовані і перевірені (ТЕСТ) без участі акумулятора.

Аналогічно можуть бути перевірені й змінені прапори та керуючі біти.



Регістри МК51	
*A	Акумулятор
*B	Регістр-розширник акумулятора
*PSW (ССП)	Слово-стан програми
SP (РУС)	Регістр-указівник стека
DPTR (РУД)	Регістр-указівник даних (DPH, DPL)
*P0	Порт 0
*P1	Порт 1
*P2	Порт 2
*P3	Порт 3
*IP (РП)	Регістр пріорітетів
*IE (РМП)	Регістр маски переривань
TMOD (РРТС)	Регістр режиму таймера / лічильника
*TCON (РУСТ)	Регістр керування / статуса таймера
TH0, TL0	Таймер 0
TH1, TL1	Таймер 1
*SCON (РУПП)	Регістр керування прийомо-передавача
SBUF	Буфер прийомо-передавача
PCON (РУМ)	Регістр керування потужністю

* – допускається адресація окремих бітів

Рисунок 2.3 – Організація доступу до регістрів МК51

Регістр слова стану програми (ССП) фіксує ряд ознак операцій (прапорів), які формуються при виконанні ряду операцій АЛП (рис. 2.5).

При скиданні в ССП установлюється 00Н.

1. *Прапор перенесення С.* Керується програмно й апаратно при виконанні арифметичних і логічних операцій, виконує функції «булевого акумулятора» у командах роботи з бітами.

2. *Прапор допоміжного перенесення AC.* Установлюється апаратно при виконанні команд додавання та віднімання (перенесення або позика в біті 3).

3. *Прапор, обумовлений користувачем F0* . Програмно-керований.

4. *Прапори вибору банку регістрів RS0-RS1*, установлюються та скидаються програмно для вибору одного з чотирьох робочих банків регістрів.

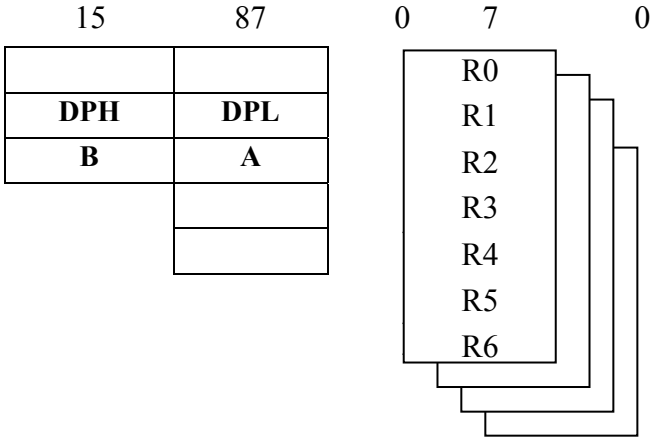


Рисунок 2.4 – Набір програмно-доступних регістрів

5. *Прапор переповнювання OV*. Встановлюється і скидається апаратно при виконанні арифметичних операцій над цілими числами зі знаком.

6. *Прапор паритету P.* Виконується контроль щодо парності вмісту акумулятора.

Регістр – указівник стека (РУС), може адресувати будь-яку область РПД. Його вміст інкрементується перш, ніж дані будуть записані в стек командою PUSH і CALL. Вміст РПС декрементується після виконання команд POP і RET. РПС є 8-бітовим, що дозволяє утворити системний стек 256 байт. Активною завжди є адреса останнього занесеного в стек байта. Стек зростає у бік збільшення вмісту РПС. Після сигналу СБР у РПС автоматично завантажується код 07Н. Це значить, що якщо прикладна програма не перевизначає стек, то перший елемент даних у стеку розташовуватиметься в комірці РПД з адресою 08Н.

Регістр – указівник даних (РУД) – 16-розрядний, кожна його 8-бітова половина (DPL і DPH) може використовуватися (бути адресована) за необхідності незалежно одна від одної.

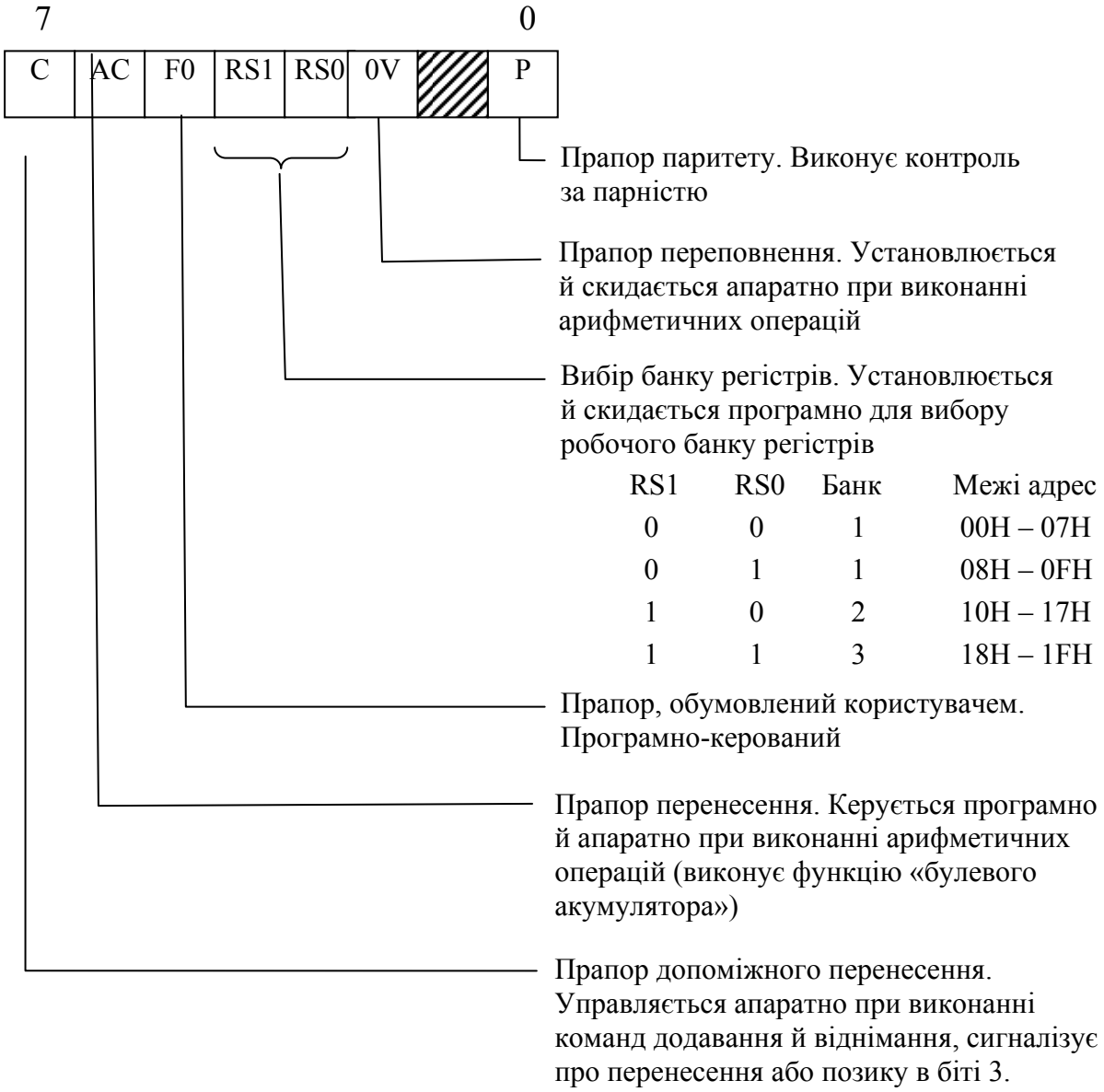


Рисунок 2.5 – Призначення бітів регістру слова стану програми (ССП)

РПД використовується як адреса при пересиланні констант з пам’яті програм і доступі до змінних із зовнішньої пам’яті даних, а також для організації передачі керування. При скиданні DPTR встановлюється 000H.

У МК51 передбачено 4 банки по 8 робочих регістрів R0-R7 у кожному. Вибір робочого банку проводиться програмно установленням 0 або 1 в бітах RS0, RS1 в регістрі ССП. Регістри виконують загальноцільові функції зберігання даних. Регістри R0 і R1 кожного банку реалізують також функції 8-розрядних указівників даних. Використання наборів робо-

чих регістрів дозволяє істотно зменшити тривалість перемикання контекстів ЦП, що дуже важливо для мікросистем реального часу.

Таймер/лічильник. У складі засобів МК51 є регістрові пари TH0, TL0 і TH1, TL1, на основі яких функціонують 2 незалежних програмно-керованих 16-бітових таймери/лічильники подій.

Буфер послідовного порту SBUF. Регістр SBUF має 2 незалежних регістри – буфер приймача і буфер передавача. Завантаження байта в SBUF негайно викликає початок процесу передачі через послідовний порт (ПП). Прочитування вмісту SBUF позначає прийом даних, джерелом яких є приймач ПП.

Регістри спеціальних функцій. Регістри з іменами IP, IE, TMOD, TCON, SCON, PCON використовують для фіксації і програмної зміни керуючих бітів і бітів стану схеми переривання, таймера/лічильника, приймача-передавача послідовного порту і для керування споживаною потужністю МК-51.

Арифметико-логічний пристрій (АЛП) – 8-розрядний, може виконувати:

- арифметичні операції складання, віднімання, множення і ділення;
- логічні операції І, АБО, ВИКЛЮЧАЮЧЕ АБО;
- операції циклічного зрушення, скидання, інвертування і т. п.

У АЛП є програмно-недоступні регістри Т1 і Т2, призначені для часового зберігання операндів, схема десяткової корекції і схема формування ознак.

Прості операції декремента і інкремента утворюють тандеми, необхідні для виконання таких операцій, як інкрементування 16-бітових регістрових пар.

АЛП дозволяє оперувати не тільки з байтами, але і з бітами.

Програмно-доступні біти можуть бути встановлені, скинуті, інвертовані, передані, перевірені, використані в логічних операціях.

АЛП може оперувати з чотирма типами інформаційних об’єктів:

- булевими (1 біт);
- цифровими (4 біт);
- байтовими (8 біт);
- адресними (16 біт).

У багатьох мікросистемах передбачені спеціальні засоби для роботи з десятковими числами. Обробка числових даних безпосередньо в числовій формі часто застосовується в системах з десятковим ВВ (введенням-виведенням), оскільки не вимагає їх проміжного перетворення в двійковий формат і назад. Для зберігання десяткових чисел у пам'яті мікросистеми використовують двійково-десятковий код. Розрізняють два формати: упакований і розпакований.

У 2–10-коді упакованого формату кожна десяткова форма від 0 до 9 подається 4-розрядним двійковим еквівалентом від 0000В до 1001В. Коди 1010В – 1111В дозволяють в одному байті розмістити тільки дві десяткові цифри і подати ціле число з діапазону 0 – 99. Для подання багаторозрядного десяткового числа відводиться більше число байтів (по байту на кожні дві цифри):

9272 = 1001 0010 0111 0010В
380 = 0011 1000 0000В

Перехід від однієї системи до іншої виконується записаною командою десяткової цифри її двійковим еквівалентом і навпаки.

У розпакованому форматі для подання кожної десяткової цифри використовується один байт, молодша тетрада містить код цифри (двійковий), а старша – нулі. Наприклад:

9272 = 00001001 000000010 00000111 00000010В
380 = 00000011 00001000 00000000В

Розпакований формат, порівняно з упакованим, вимагає в два рази більше пам'яті. Проте він дуже добре узгоджується з тестовим поданням цифр у коді КОИ-7. Для перекладу розпакованого 2–10 коду в тестовий формат достатньо в старшу тетраду кожної цифри записати 0011В. За необхідності знак десяткового числа кодується окремою тетрадою або байтом. У обох випадках комбінація 0000В відповідає « + », а 1001В – « – ».

Наприклад, для упакованого формату:

+ 921 = 0000 1001 0010 0001В;
– 350 = 1001 0011 0101 0000В.

У розпакованому форматі

– 350 = 00001001 00000011 00001101 00000000В.

Арифметична операція над десятковими числами виконується в два етапи: спочатку проводиться звичайна двійкова операція над десяткови-

ми числами, потім десяткова корекція одержаного результату. В цьому випадку можна скористатися схемою десяткової корекції АЛП, двійкової операції, що приводить результат до десяткового вигляду.

Корекція виконується, якщо в результаті операції вміст перевершує 9. Для розпакованого формату корекція відбувається шляхом збільшення числа 0F6H, завдяки чому результат стає правильним, десяткове перенесення шифрується і може бути враховано при складанні більш старших десяткових цифр.

Пристрій керування і синхронізації

Кварцевий резонатор, що підключається до зовнішніх виводів X1 і X2 корпусу МК51, керує роботою внутрішнього генератора, який в свою чергу формує сигнали синхронізації.

Пристрій керування МК51 на основі сигналів синхронізації формує машинний сигнал фіксованої тривалості, рівний 12 періодам резонатора або шести станам первинного керуючого автомата. Кожен стан первинного керуючого автомата містить дві фази сигналів резонатора. У першій фазі виконується операція в АЛП, а в другій здійснюється мікрорегістрова передача.

Зовнішніми спостережуваними сигналами є тільки сигнали резонатора і строба адреси зовнішньої пам’яті (САЗП), який формується двічі за машинний цикл (S1P2-S1P1 і S4P2-S5P1).

Більшість команд виконується за один машинний цикл. Деякі команди, що оперують з 2-байтовими словами або пов’язані зі зверненням до зовнішньої пам’яті, виконуються за два машинні цикли. Тільки команди ділення і множення вимагають чотирьох машинних циклів. Розрахунок часу виконання прикладних програм проводиться з урахуванням цих особливостей.

Порти введення-виведення (ВВ)

Підсистема ВВ МК51 розміщується безпосередньо на кристалі. Для ВВ даних і керування процесом їх передачі до складу МК51 введений ряд портів даних і регістрів керування/стану. Фізична система ВВ мікроконтролера складається з чотирьох двонаправлених 8-розрядних портів P0–P3, призначених для забезпечення обміну інформацією мікро-

контролера з зовнішніми пристроями, утворюючи 32 лінії ВВ і може виконувати ряд додаткових функцій. Всі порти відображені в просторі внутрішньої пам'яті і нічим не відрізняються від звичайних елементів пам'яті. Це дозволяє застосувати до вмісту портів команди пам'яті з операндом з внутрішньої пам'яті. Кожний з портів містить фіксатор-защіпку, яка являє собою 8-розрядний регістр, що має байтову і бітову адресацію для скидання установлення програмними засобами. Це забезпечує можливість доступу до окремих розрядів портів.

При зверненні до зовнішньої пам'яті програм або даних порти P0, P2 виконують функції AD і AB відповідно. Молодший байт адреси і дані передаються через P0 в мультиплексному режимі: спочатку вводиться адреса, потім порт використовується для передачі даних. Коли розрядність адреси дорівнює 16 біт, старший байт адреси формується через P2.

Усі виведення порту P3 можуть бути використані для реалізації альтернативних функцій, наведених у табл. 2.3.

Таблиця 2.3 – Альтернативні функції порту 3

Позиція	Символ	Ім'я і призначення
P3.0	RxD	Вхід приймача послідовного каналу порту. Введення-виведення даних у режимі регістра зсуву
P3.1	TxD	Вихід передавача послідовного каналу порту. Вихід синхронізації у режимі регістра зсуву
P3.2	INT0	Вхід запиту на переривання 0. Сприймається сигнал низького рівня або зріз
P3.3	INT1	Вхід запиту на переривання 1. Сприймається сигнал низького рівня або зріз
P3.4	T0	Зовнішній вхід таймера/лічильника 0 або тест вхід
P3.5	T1	Зовнішній вхід таймера/лічильника 1 або тест вхід
P3.6	WR	Запис. Активний сигнал низького рівня формується при зверненні до ЗПД
P3.7	RD	Читання. Активний сигнал низького рівня формується при зверненні до ЗПД

Альтернативні функції можуть бути задіяні шляхом запису 1 у відповідні біти регістра-защіпки (P3.0–P3.7 порту 3).

Порт P0 є двонаправленим, а P1–P3 – квазідвонаправленими.

Кожна лінія портів може бути використана незалежно для введення або виведення інформації. Для режиму введення в регістр-защіпку записується 1.

Кожен порт містить керований регістр-защіпку, вхідний буфер і вихідний драйвер. У портах P0 і P2 у вихідному буфері передбачені ключі, сполучаючі їх виводи з внутрішніми шинами AD і A відповідно.

За сигналом СБР (RST) у регістри-защіпки усіх портів автоматично записуються одиниці, настроюючи їх цим на режим введення.

При роботі порту, коли він одночасно є операндом і місцем призначення результату, пристрій керування автоматично реалізує спеціальний режим «читання – модифікація – запис». При цьому введення сигналів відбувається не із зовнішніх виводів порту, а з його регістра-защіпки, що дозволяє виключити неправильне прочитування раніше виведеної інформації.

Цей механізм звернення до портів реалізується такими командами:

ANL P2, A; ORL P3, A; XRL P1, A; JBC P1. 1, LABEL; CPL P1.2; INC P1; DEC P3; DJNZ P2, LABEL; MOV PX.Y,C; SET PX.Y; CLR PX.Y.

За цими командами спочатку прочитується байт з порту, а потім записується новий байт у регістр-защіпку.

Послідовний інтерфейс (УАПП)

Асинхронний приймач-передавач (УАПП) здійснює прийом і передачу інформації, поданої послідовним кодом у повному дуплексному режимі обміну.

УАПП (послідовний порт) включає:

- приймаючий регістр зсуву ;
- передавальний регістр зсуву;
- спеціальний буферний регістр (SBUF) приймача-передавача.

Наявність буферного регістра приймача дозволяє суміщати операцію читання раніше прийнятого байта з прийомом чергового байта. Якщо до моменту закінчення прийому байта попередній байт не був зчитаний

з SBUF, то він буде втрачений. Послідовний інтерфейс може працювати в чотирьох різних режимах.

Режим 0 – інформація передається і приймається через зовнішній вивід входу приймача (RxD). Приймаються або передаються 8 біт даних. Через зовнішній вивід виходу передавача (TxD) видаються імпульси зсуву, які супроводжують кожен байт. Частота передачі біта інформації дорівнює 1/12 частоти резонатора.

Режим 1 – передаються через TxD і приймається з RxD 10 біт інформації: старт-біт (0), 8 біт даних і стоп-біт (1). Швидкість прийому/передачі – величина змінна і задається таймером.

Режим 2 – через вивід TxD передаються, а через RxD приймаються 11 біт інформації: старт-біт, 8 біт даних, програмований дев'ятий біт і стоп-біт. При передачі дев'ятий біт може набувати значення 0 або 1, або, наприклад, для підвищення достовірності передачі шляхом контролю парності в нього може бути поміщено значення ознаки паритету з ССП. Частота прийому передачі вибирається програмою і може дорівнювати 1/32 або 1/64 частоти резонатора залежно від керуючого біта SMOD.

Режим 3 – збігається з режимом 2 в усіх відношеннях, за винятком частоти прийому передачі, яка є величиною змінною і задається таймером. Керування режимами роботи УАПП здійснюється через спеціальний регістр з символічним ім'ям SCON, наведений у табл. 2.4.

Прикладна програма шляхом завантаження в старші біти спеціального регістра SCON 2-бітового коду визначає режим роботи УАПП.

У всіх чотирьох режимах роботи передача з УАПП (послідовного порту) ініціюється будь-якою командою, в якій буферний регістр SBUF вказаний як одержувач байта.

Прийом з УАПП у режимі 0 здійснюється за умови, що RI = 0 і REN = 1. У режимах 1, 2, 3 прийом починається з приходом старт-біта, якщо REN = 1. У біті TB8 програмно встановлюється значення дев'ятого біта даних, який буде переданий в режимі 2 або 3. У біті RB8 фіксується в режимах 2 і 3 дев'ятий біт даних, що приймається. У режимі 1, якщо SM2 = 0, у біт RB8 заноситься стоп-біт. У режимі 0 біт RB8 не використовується.

Таблиця 2.4 – Призначення бітів регістра керування – статусу УАПП

Позиція	Символ	Ім’я і позиція
SCON.0	RI	Прапор переривання приймача. Встановлюється апаратно при прийомі байта. Скидається програмно після обслуговування переривання
SCON.1	TI	Прапор переривання передавача. Встановлюється апаратно при закінченні передачі байта. Скидається програмно після обслуговування переривання
SCON.2	RB8	Прийом біта 8. Встановлюється / скидається апаратно для фіксації дев’ятого біта, що приймається, в режимі УАПП – 9 біт
SCON.3	TB8	Передача біта 8. Встановлюється / скидається програмно для задання дев’ятого передаваного біта в режимі УАПП-9 біт
SCON.4	REN	Біт дозволу прийому. Встановлюється / скидається програмно для дозволу / заборони прийому послідовних даних
SCON.5	SM2	Біт керування режимом УАПП. Встановлюється програмно для заборони прийому повідомлення, в якому 9-й біт має значення 0
SCON.6	SM1	Біти керування режимом роботи УАПП. Встановлюється / скидається програмно
SCON.7	SM0	

Примітка. Вибір режимів роботи УАПП:

SM0	SM1	Режим роботи УАПП
0	0	Зрушуючий регістр розширення введення/виведення.
0	1	УАПП-8 біт. Змінна швидкість передачі.
1	0	УАПП-9 біт. Фіксована швидкість передачі.
1	1	УАПП-9 біт. Змінна швидкість передачі.

Прапор переривання передавача TI встановлюється апаратно в кінці періоду передачі восьмого біта даних у режимі 0 і на початку періоду передачі стоп-біта в режимах 1, 2 і 3. Відповідна підпрограма обслуговування переривання повинна скидати біт TI. Прапор переривання приймача RI встановлюється апаратно в кінці періоду прийому восьмого біта даних у режимі 0 і у середині періоду прийому стоп-біта в режимах 1, 2 і 3. Підпрограма обслуговування переривання повинна скидати біт RI. Швид-

кість прийому-передачі, тобто частота роботи УАПП в різних режимах, визначається різними способами.

У режимі 0 частота передачі залежить тільки від резонансної частоти кварцевого резонатора $f_0 = f_{рез}/12$. За один машинний цикл послідовний порт передає один біт інформації.

У режимах 1, 2 і 3 швидкість прийому-передачі залежить від значення керуючого біта SMOD у регістрі спеціальних функцій РКП (керування потужністю), наведеному в табл. 2.5.

Таблиця 2.5 – Призначення бітів регістра керування потужністю РКП (PCON)

PCON.0	IDL	Біт холостого ходу. Якщо біт установлений в 1, то МК переходить у режим холостого ходу
PCON.1	PD	Біт зниженої потужності. При установленні біта в 1 МК переходить у режим зниженої споживаної потужності (його пріоритет вище, ніж у IDL)
PCON.2	GF0	Прапори, визначувані користувачем (прапори загального призначення)
PCON.3	GF1	
PCON.4	–	Не використовуються
PCON.5	–	
PCON.6	–	
PCON.7	SMOD	Подвоєна швидкість передачі. Якщо біт установлений в 1, то швидкість передачі вдвоє більше, ніж при SMOD = 0

У режимі 2 частота передачі визначається виразом

$$f_2 = \left(2^{SMOD}/64\right) f_{рез}$$

При SMOD, рівному нулю, частота передачі дорівнює $\left(1/64\right) f_{рез}$, а при SMOD, рівному 1, частота передачі дорівнює $\left(1/32\right) f_{рез}$.

У режимах 1 і 3 у формуванні частоти передачі, окрім керуючого біта SMOD бере участь таймер 1. При цьому частота передачі залежить від частоти переповнювання OVT1 і визначається

$$f_{1,3} = \left(2^{SMOD} \diagup 32\right) \cdot f_{OVT1}.$$

Переривання від таймера 1 у цьому випадку повинне бути заблоковане. Сам таймер-лічильник 1 може працювати і як таймер, і як лічильник в одному з трьох режимів. Нижче в табл. 2.6 наводиться опис способів настройки Т/С 1 для отримання типових частот передачі даних через УАПП.

Таблиця 2.6 – Типові частоти прийому – передачі даних через УАПП

Частоти прийому/передачі (BAUD RATE)	Частота резонатора, Мгц	SMOD	Таймер/лічильник 1		
			С/Т	Режим (MODE)	Перезавантажуване число
Режим 0, max 1 мГц	12	X	X	X	X
Режим 2, max 375 кГц	12	1	X	X	X
Режим 1,3; 62,5 кГц	12	1	0	2	0FFH
19,2 кГц	11,059	1	0	2	0FDH
9,6 кГц	11,059	0	0	2	0FDH
4,8 кГц	11,059	0	0	2	0FAH
2,4 кГц	11,059	0	0	2	0F4H
1,2 кГц	11,059	0	0	2	0E8H
137,5 кГц	11,059	0	0	2	1DH
110 кГц	6	0	0	2	72H
110 кГц	12	0	0	1	0FEEBH

Доступ до зовнішньої пам'яті. У мікроконтролерних системах, побудованих на основі МК51, можливе використання двох типів зовнішньої пам'яті: постійної пам'яті програм (ЗПП) і оперативної пам'яті даних (ЗПД).

Доступ до ЗПП здійснюється за допомогою керуючого сигналу, який виконує функцію строб-сигналу читання. При цьому $\overline{\text{ОРПП}}\left(\overline{\text{EA}}\right)$ активний.

Доступ до ЗПД забезпечується керуючими сигналами $\overline{\text{ЧТ}}\left(\overline{\text{RD}}\right)$, які формуються в лініях Р3.7 і Р3.6 при виконанні портом 3 альтернативних функцій.

При зверненні до ЗПП завжди використовується 16-бітова адреса (через DPTR).

При зверненні до ЗПД можливе використання 16- або 8-бітової адреси (через DPTR або R0-R1).

У будь-яких випадках використання 16-бітової адреси старший байт адреси фіксується (і зберігається незмінно протягом першого циклу читання /запису) в регістрі-защіпці порту 2.

Якщо черговий цикл зовнішньої пам'яті (MOVX A @DPTR) йде не відразу ж за попереднім циклом зовнішньої пам'яті, то незмінний вміст регістра-защіпки порту 2 відновлюється в наступному циклі.

Якщо використовується 8-бітова адреса (MOVX A @Ri), то вміст регістра-защіпки порту 2 залишається незмінним на його зовнішніх виведеннях протягом усього циклу зовнішньої пам'яті. Через порт 0 в режимі тимчасового мультиплексування здійснюється видача молодшого байта адреси і передача байта даних. Сигнал САЗП повинен бути використаний для запису байта адреси в зовнішній регістр. Потім в циклі запису байт даних, що виводиться, з'являється на зовнішніх виводах порту 0 тільки перед появою сигналу $\overline{\text{ЗП}}\left(\overline{\text{WR}}\right)$. У циклі читання байт даних, що вводиться, приймається в порт 0 по фронту стробувального сигналу $\overline{\text{ЧТ}}\left(\overline{\text{RD}}\right)$.

Сигнал САЗП, що забезпечує тимчасове узгодження передачі з порту 0 на зовнішній регістр молодшого байта адреси, використовується тільки для читання з ЗПП.

Доступ до ЗПД можливий тільки в тому випадку, якщо сигнал САЗП відсутній, тому при виконанні команди MOVX (звернення до ЗПД) він блокується.

У системах без ЗПД сигнал САЗП можна використовувати для синхронізації зовнішніх пристроїв з частотою 1/16.

Особливий режим роботи МК51

На етапі налагодження прикладних програм, а також за необхідності оперативного перевантаження і модифікації прикладних програм у реальних системах зовнішня пам'ять мікроконтролера може бути модифі-

кованою для поєднання адресного простору ЗПП і ЗПД. Це здійснюється шляхом підключення зовнішньої логіки, поданої нижче на рис. 2.6.

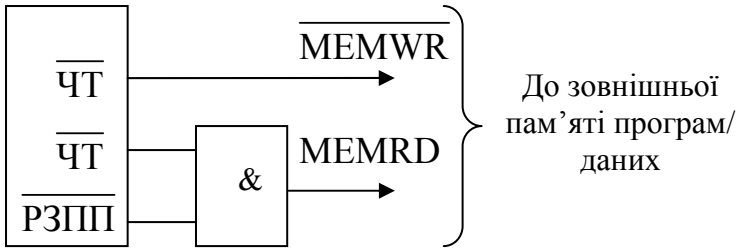


Рисунок 2.6 – Схема підключення зовнішньої пам'яті

Слід пам'ятати, що способи адресації РПП, РПД, ЗПП, ЗПД і блоку регістрів спеціальних функцій різні, тому переміщувана версія прикладної програми, яка відладжується в середовищі зовнішньої пам'яті програм даних, відрізнятиметься від завантаженої в РП версії програми.

Таймер / лічильник

Таймер/лічильник належить до стандартних засобів підтримки режиму реального часу. Він використовується для організації системних позначок реального часу, формування тимчасових затримок і підрахунку зовнішніх подій.

До складу МК51 входять два 16-розрядні таймери/лічильники Т0 і Т1, стан яких відображається програмно-доступними регістровими парами (TL0, TH0) і (TL1, TH1). Т1 і Т0 можуть бути запрограмовані для роботи або як таймер, або як лічильник. Функція таймера полягає в підрахунку числа машинних циклів йдучих з частотою $f_{рг}/12$. Функція лічильника полягає у відстежуванні переходів з 1 в 0 на відповідних входах Т0 і Т1. Опит значення зовнішнього вхідного сигналу виконується в кожному машинному циклі (S5S2). На розпізнавання переходу потрібно два машинні цикли.

Для керування режимами роботи Т/Л і для організації взаємодії таймера з системою переривань використовуються два регістри: регістр режиму таймера/лічильника (РРТЛ) – рис. 2.7 і регістр керування/статусу таймера (РУСТ) – рис. 2.8. РРТЛ (TMOD) розбитий на два 4-розрядні підрегістри T0MOD і T1MOD, які відповідальні за керування Т0 і Т1 відповідно.

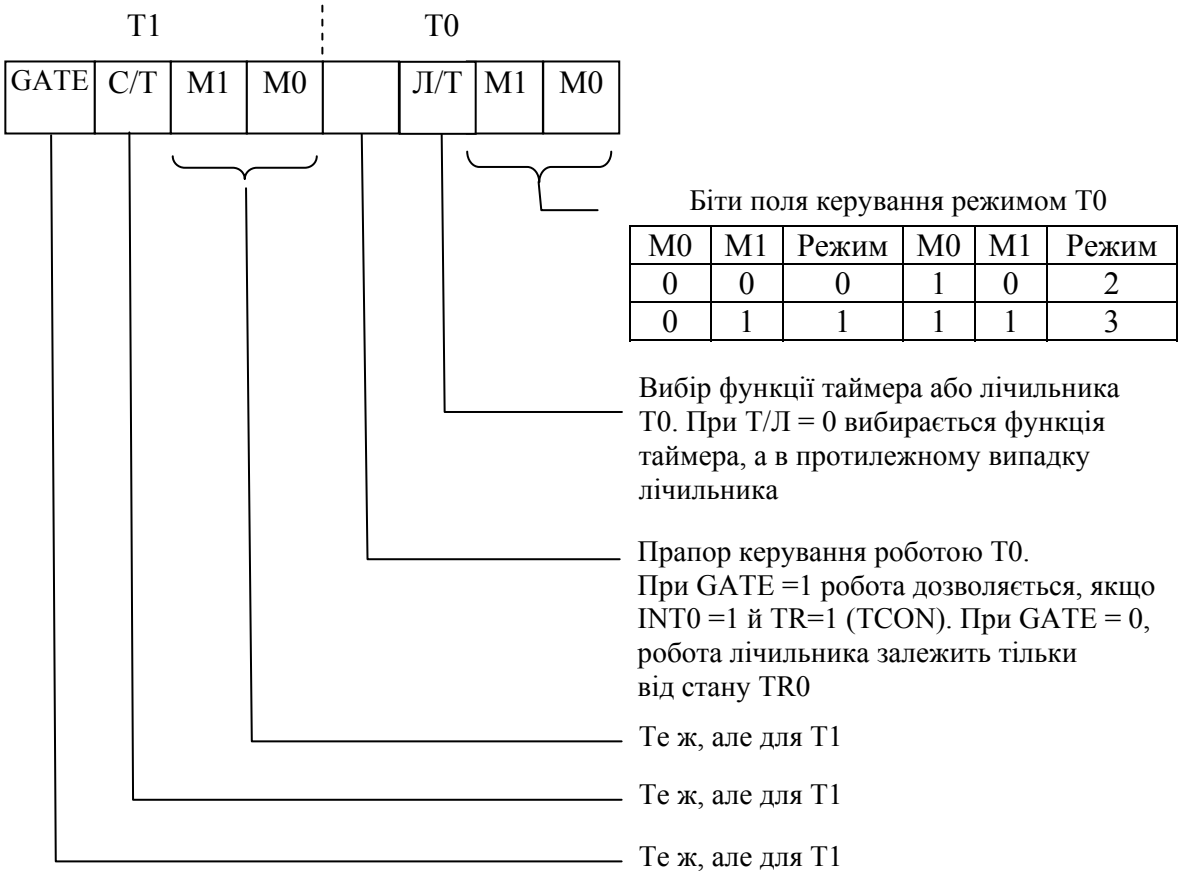


Рисунок 2.7 – Призначення бітів регістра режиму роботи таймера / лічильника (РРТЛ)

Як видно з опису керуючих бітів РРТС, наведеного на рис. 2.7, для обох таймерів/лічильників (Т/Л) режими роботи 0, 1 і 2 однакові. Режими 3 для Т/Л0 і Т/Л1 різні. Нижче наводиться опис роботи Т/Л у всіх 4-х режимах 0, 1, 2, і 3.

Режим 0. У цьому режимі регістр таймера має розрядність 13 біт. При переході зі стану «Всі одиниці» у стан «Всі нулі» встановлюється прапор переривання від таймера TF1. Установлення GATE в одиницю дозволяє використовувати таймер для вимірювання тривалості імпульсного сигналу, що подається на вхід запиту переривання.

Режим 1. Робота будь-якого Т/Л така ж, як і в режимі 0, за винятком того, що регістр таймера має розрядність 16 біт.

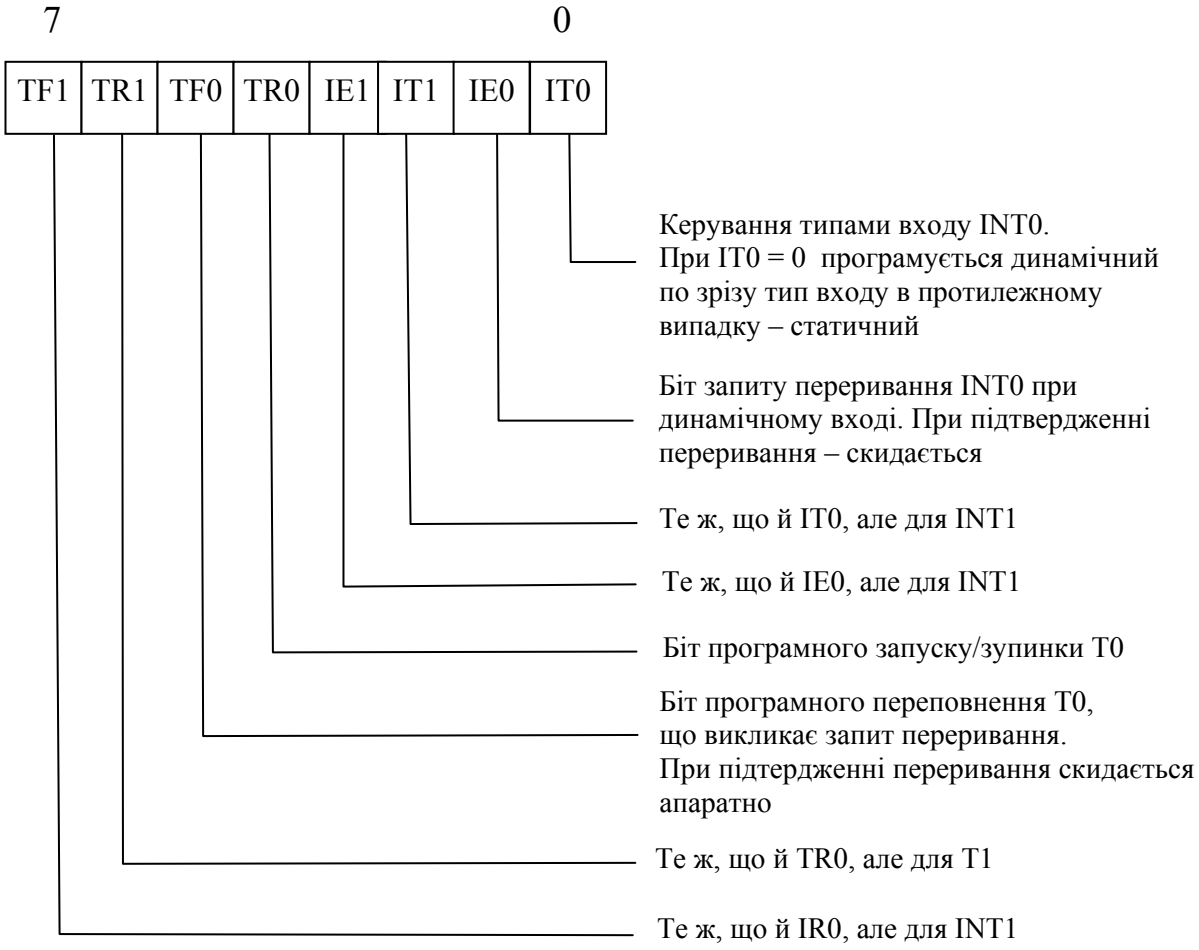


Рисунок 2.8 – Призначення бітів регістра керування статусу таймера (РУСТ)

Режим 2. Робота організована таким чином, що переповнювання (перехід з стану «Всі одиниці» в стан «Всі нулі») 8-бітового лічильника TL1 приводить не тільки до установлення прапора TF1, але і автоматично перезавантажує в TL1 вміст старшого байта (TH1) регістра таймера, який заздалегідь був заданий програмним шляхом. Перезавантаження залишає вміст TH1 незмінним.

У цьому режимі обидва таймери/лічильники працюють однаково.

Режим 3. Робота T/L0 і T/L1 відбувається по-різному. T/L1 зберігає незмінним свій поточний стан, тобто так само, як при скиданні керуючого біта TR1 в нуль. TL0 і TH0 функціонують, як два незалежні 8-бітові лічильники. Роботу TL0 визначають біти GATE0 і TR0, вхідний сигнал

$\overline{INT0}$ і прапор переповнювання TF0. Роботу TH0, який може виконувати тільки функції таймера, визначає біт, що керує TR1. При цьому TH0 використовує прапор переповнювання TF1. Цей режим використовується, коли є необхідність наявності додаткового 8-бітового таймера або лічильника.

Система переривань

Система переривань призначена для реагування на зовнішні і внутрішні події шляхом встановлення відповідних прапорів IE0, IE1, TF0,TF1, RI, TI. Усі перераховані прапори можуть бути програмно встановлені або скинуті, що відповідно викликає або усуває переривання. Спрощена схема переривань наведена на рис. 2.9. Керування системою переривань здійснюється шляхом запису керуючих слів у регістри: IE – регістр дозволу переривань (рис. 2.10) і IP – регістр пріоритетів переривань (рис. 2.11). Регістр дозволу переривань IE призначений для дозволу або заборони переривань від відповідних джерел. Регістр IP призначений для встановлення рівня пріоритету переривання для кожного з п’яти джерел переривань. Зовнішні переривання INT0 і INT1 можуть бути викликані або рівнем, або переходом сигналу з 1 в 0 (регістр TCON). Прапори IE0 і IE1 (регістр TCON) встановлюються залежно від зовнішніх переривань і ініціюють відповідні програми обслуговування переривання. Скидання прапорів відбувається апаратно, тільки якщо переривання було викликано за зрізом (переходу) сигналу.

Якщо виклик відбувається за рівнем переривання, то скидання прапора здійснюється програмно. Прапори запитів переривання від таймера TF0 і TF1 скидаються автоматично під час переходу керування до підпрограми обслуговування. Прапори запитів переривання TI, RI від УАПП установлюються блоком керування УАПП апаратно, але скидатися повинні програмно. Переривання можуть бути викликані або відмічені програмою, оскільки всі перераховані прапори переривань опитуються на кожному машинному циклі (момент S5P2).

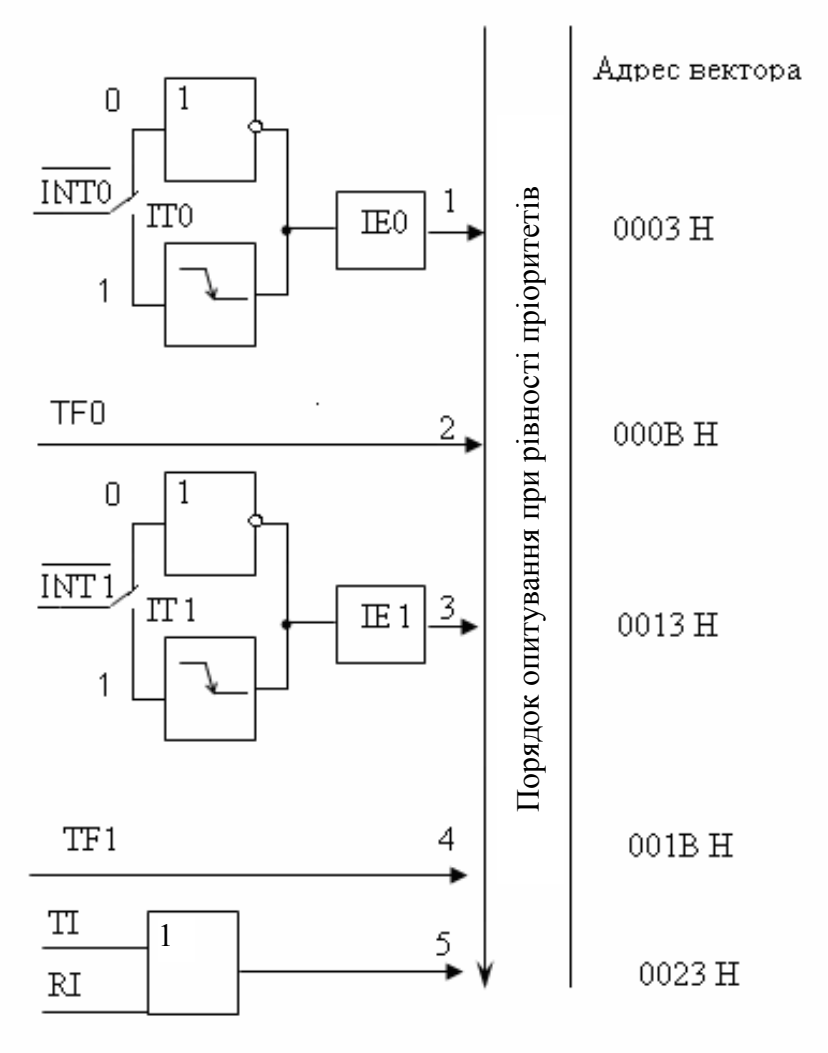


Рисунок 2.9 – Схема переривань МК51

Ранжирування переривань за рівнем пріоритету виконується протягом наступного машинного циклу. Система переривань викликає (апаратно формує виклик LCALL) програму обслуговування переривань, якщо вона не заблокована однією з таких умов:

- у даний момент обслуговується запит переривання рівного або вищого рівня пріоритету;
- поточний машинний цикл не останній в циклі виконуваної команди;
- виконується команда, пов’язана зі зверненням до регістрів IE і IP.

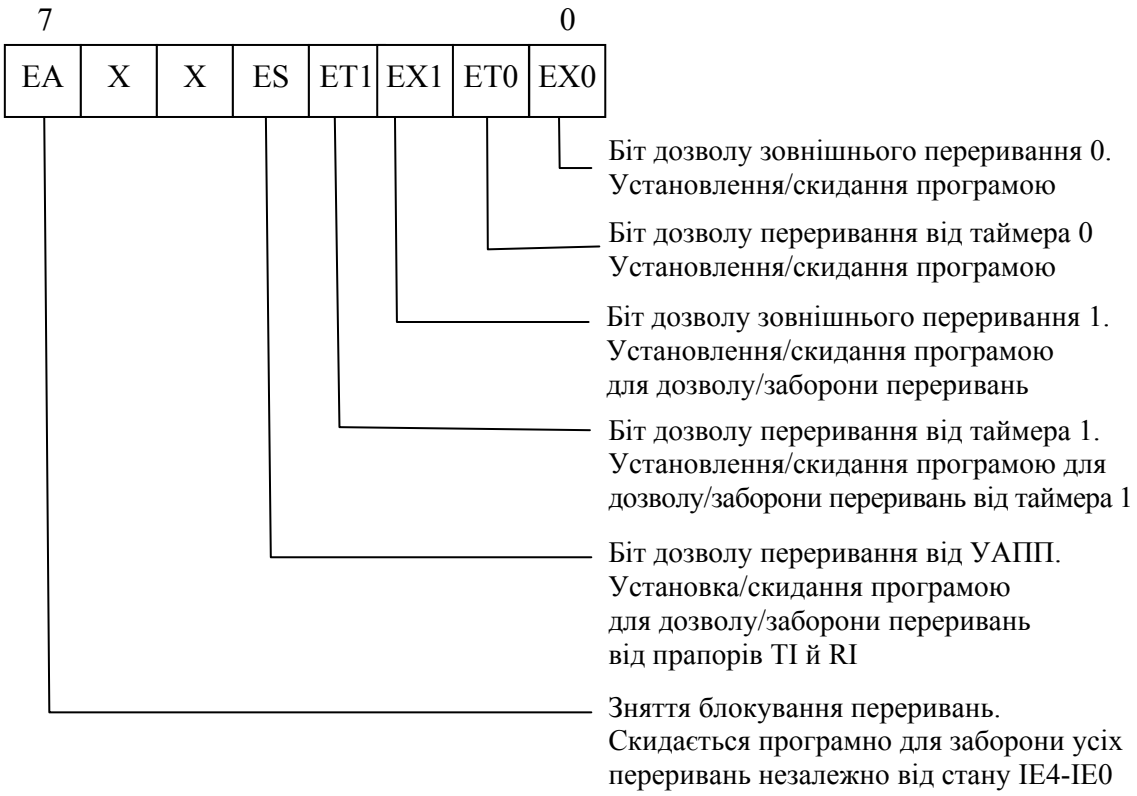


Рисунок 2.10 – Призначення бітів регістру дозволу переривань (IE)

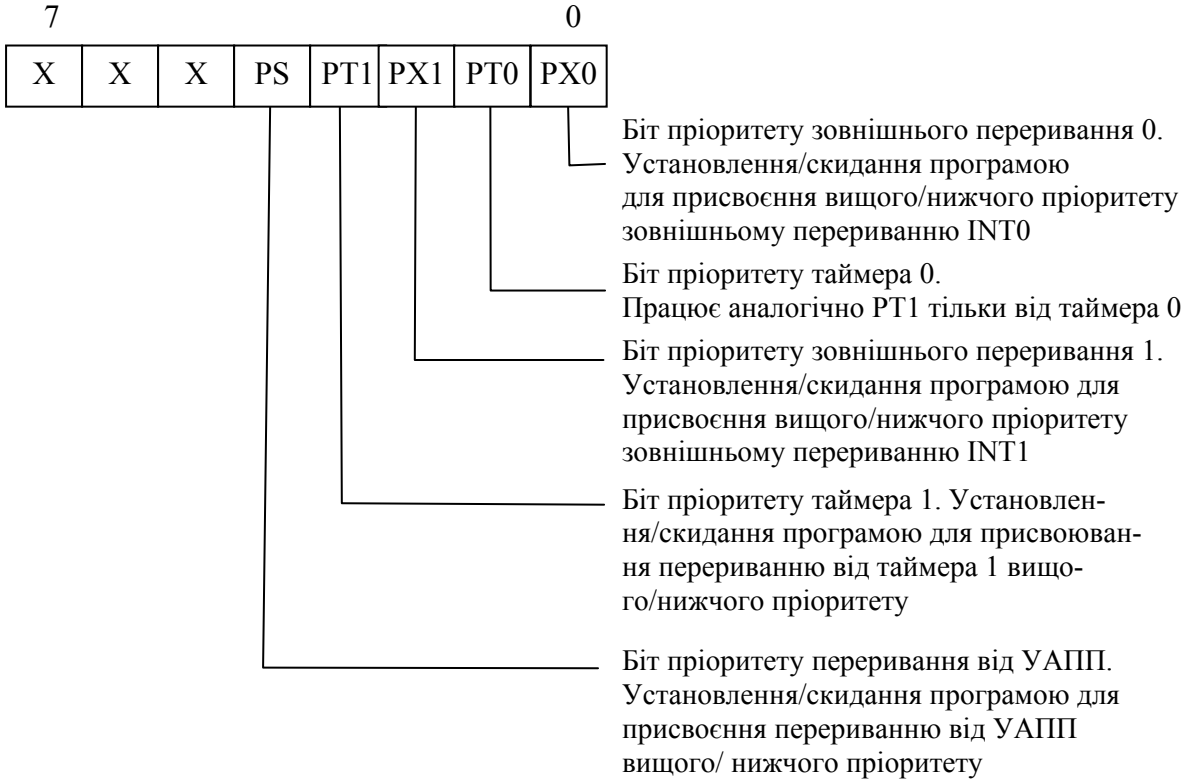


Рисунок 2.11 – Призначення бітів регістру пріоритетів переривань (IP)

За адресою вектора переривання повинна бути розташована команда безумовної передачі керування (JMP) до початкової адреси підпрограми обслуговування переривання. Підпрограми обслуговування переривання обов'язково повинні закінчуватися командою RETI. За нею знімається блокування переривання, і в лічильник команд зі стека перезавантажується збережена адреса повернення в основну програму.

Скидання, режими холостого ходу і зниженого електроживлення

Скидання здійснюється шляхом подачі на вхід СБР (RST) сигналу 1. Для впевненого скидання сигнал 1 повинен утримуватися протягом двох машинних циклів (24 періоди резонатора). Квазідвонаправлені буферні схеми зовнішніх виведень ALE і $\overline{\text{PSEN}}$ знаходяться при цьому в режимі введення. Скидається вміст таких регістрів: PC, A, B, PSW, DPTR, TMOD, TCON T/C0, T/C1, IE, IP, SCON.

У режимі PCON скидається тільки старший біт. У регістр указівника стека SP завантажується 07H, у порти – P0 і P3 коди 0FFH. Стан SBUF невизначений і на вміст комірок РПД сигнал не впливає.

Режим холостого ходу. Будь-яка команда, за якою встановлюється біт IDL регістра PCON.0, переводить МК51 в режим холостого ходу. При цьому продовжує працювати внутрішній генератор синхросигналів, і всі регістри зберігають свої значення. На виходах усіх портів утримується той логічний стан, який на них був у момент переходу в режим холостого ходу. Вийти з режиму холостого ходу можна за сигналом СБР (RST) або перериванням.

Режим зниженого енергоспоживання. Перехід у режим здійснюється за командою установлення біта PD (PCON.1) в 1. При цьому відбувається зупинка генератора синхросигналів, зберігається вміст РПД, РСФ, утримується напруга на виходах портів. Напруга енергоживлення VCC +5В може бути відключена. Живлення здійснюється через виведення RST (СБР). Вихід з режиму зниженого енергоспоживання можливий тільки за сигналом СБР. При цьому вміст регістрів спеціальних функцій перевизначається, а вміст РПД не змінюється.

Примітка. Склад регістра PCON залежить від технології виготовлення контролера. При *n-МОП* технології PCON містить усього лише один прапор SMOD. Повний склад прапорів можна зустріти тільки в КМОП варіантах.

Система команд МК51. Загальні відомості

Система команд, наведена у табл. 2.7, налічує 111 команд, з них: 49 – однобайтових; 45 – двобайтових; 17 – трибайтових.

Таблиця 2.7 – Команди Асемблера МК51

Мнемокод	КОП	Т Б Ц	Коментар
Команди пересилання, обміну і завантаження			
MOV A, Rn	11101rrr	1 1 1	A ← Rn;
MOV A, @Ri	1110011i	1 1 1	A ← (Ri);
MOV A, dir	11100101	3 2 1	A ← (dir);
MOV A, #DAT	01110100	2 2 1	A ← # DAT
MOV Rn, A	11111rrr	1 1 1	Rn ← A;
MOV @Ri, A	1111011i	1 1 1	(Ri) ← A;
MOV dir, A	11110101	3 2 1	(dir) ← A;
MOV DIR, #DAT	1111011i	7 3 2	(dir) ← # DAT;
MOV Rn, #DAT	01111rrr	2 2 1	Rn ← # DAT;
MOV @Ri, #DAT	0111011i	2 2 1	(Ri) ← # DAT;
MOV dir, dir	10000101	9 3 2	(dir) ← (dir);
MOV Rn, dir	10101rrr	3 2 2	Rn ← (dir);
MOV @Ri, dir	1010011i	3 2 2	(Ri) ← (dir);
MOV dir, Rn	10001rrr	3 2 2	(dir) ← Rn;
MOV dir, @Ri	1000011i	3 2 2	(dir) ← (dir);
MOV DPTR, # D16	10010000	3 2 2	DPTR ← # D16;
XCH A, Rn	11001rrr	1 1 1	A ↔ Rn;
XCH A, @Rn	1100011i	1 1 1	A ↔ (Rn);
XCH A, dir	11000101	3 2 1	A ↔ (dir);
XCHD A, @Ri	1101011i	1 1 1	A ↔ (Ri);
SWAP A	11000100	1 1 1	Обмін тетрад акумулятора
PUSH dir	11000000	3 2 2	SP ← SP + 1; SP ← (dir);
POP dir	11010000	3 2 2	(dir) ← (SP); SP ← SP-1;
MOVX A, @DPTR	11100000	1 1 2	A . (DPTR);
MOVX @DPTR, A	11110000	1 1 2	(DPTR) ← A;
MOVX A, @Ri	1110001i	1 1 2	A ← (Ri);
MOVX @Ri, A	1111001i	1 1 2	(Ri) ← A;
MOVC A, @A+DPTR	10010011	1 1 2	A ← ((A + DPTR));
MOVC A, @A+PC	10000011	1 1 2	A ← ((A + PC));
ADD A, Rn	00101rrr	1 1 1	A←A + Rn;
ADDC A, Rn	00111rrr	1 1 1	A ← A + Rn + C;
ADDC A, @RI	0011011i	1 1 1	A ← A + (Ri) + C;
ADDC A, dir	00110101	3 2 1	A ← A + (dir) + C;
ADDC A, #DAT	00110100	2 2 1	A ← A + # DAT + C;
DA A	11010100	1 1 1	Десяткова корекція акумулятора

Продовження табл. 2.7

Мнемокод	КОП	Т Б Ц	Коментар
Арифметичні і логічні команди			
SUBB A, Rn	10011rrr	1 1 1	A ← A – Rn - C;
SUBB A, @Ri	1001011i	1 1 1	A ← A - (Ri) - C;
SUBB A, dir	10010101	3 2 1	A ← A - (dir) - C;
SUBB A, #DAT	10010100	2 2 1	A ← A - # DAT - C;
INC A	00000100	1 1 1	A ← A + 1;
INC Rn	00001rrr	1 1 1	Rn ← Rn + 1;
INC @Ri	0000011i	1 1 1	(Ri) ← (Ri) + 1;
INC dir	00000101	3 2 1	(dir) ← (dir) + 1;
INC DPTR	10100011	1 1 2	DPTR ← DPTR + 1;
DEC A	00010100	1 1 1	A ← A - 1;
DEC Rn	00011rrr	1 1 1	Rn ← Rn - 1;
DEC @Ri	0001011i	1 1 1	(Ri) ← (Ri) - 1;
DEC dir	00010101	3 2 1	(dir) ← (dir) - 1;
MUL AB	10100100	1 1 4	BA ← A * B; Множення A на B B ← старший байт, A ← мол. байт
DIV AB	10000100	1 1 4	A • B ← A/B; Ділення A на B, A ← байт частки B ← залишок
ANL A, Rn	01011rrr	1 1 1	A ← A & Rn ;
ANL A, @Ri	0101011i	1 1 1	A ← A & (Ri);
ANL A, dir	01010101	3 2 1	A ← A & (dir);
ANL A, #DAT	01010100	2 2 1	A ← A & # DAT;
ANL dir, A	01010010	3 2 1	(dir) ← (dir) & A;
ANL dir, #DAT	01010011	7 3 2	(dir) ← (dir) & # DAT;
ORL A, Rn	01001rrr	1 1 1	A ← A ∨ Rn;
ORL A, @Ri	0100011i	1 1 1	A ← A ∨ (Ri);
ORL A, dir	01000101	3 2 1	A ← A ∨ (dir);
ORL A, #DAT	01000100	2 2 1	A ← A # DAT
ORL dir, A	01000010	3 2 1	(dir) ← (dir)/A;
ORL dir, #DAT	01000011	7 3 2	(dir) ← (dir)/ # DAT
Логічні команди			
XRL A, Rn	01101rrr	1 1 1	A ← A (+) Rn
XRL A, @Ri	0110011i	1 1 1	A ← A (+) (Ri)
XRL A, dir	01100101	3 2 1	A ← A (+) (dir)
XRL A, #DAT	01100100	2 2 1	A ← A (+) # DAT
XRL dir, A	01100010	3 2 1	(dir) ← (dir) (+) A
XRL dir, #DAT	01100011	7 3 2	(dir) ← (dir) (+) # DAT
CLR A	11100100	1 1 1	A ← 0; Скидання аккум
CPL A	11110100	1 1 1	A ← $\overline{A}$; Інверсія акумулятора
RL A	00100011	1 1 1	Циклічне зрушення вліво
RLC A	00110011	1 1 1	Циклічне зрушення вліво через перенесення
RR A	00000011	1 1 1	Циклічне зрушення вправо
RRC A	00010011	1 1 1	Циклічне зрушення вправо через перенесення

Продовження табл. 2.7

Мнемокод	КОП	Т Б Ц	Коментар
Команди безумовних переходів і виклику підпрограм			
LJMP 16ADR	00000010	12 3 2	PC[0 15] ← 16ADR; довгий перехід.
AJMP 11ADR	sss00001	6 2 2	PC[0 11] ←11ADR; абсолютний перехід всередині сторінки в 2 Кбайт.
SJMP rel	10000000	5 2 2	PC ← PC + 2 + rel; відносний перехід.
JMP @A+DPTR	01110011	1 1 2	PC ← A + DPTR; непрямий відносний перехід
LCALL 16ADR	00010010	12 3 2	(SP) ← PC + 3; PC[0 15] ← 16ADR; довгий виклик підпрограми.
ACALL 11ADR	sss10001	6 2 2	(SP) ← PC + 2; PC[0 10] ← 11ADR; абсолютний виклик підпрограми в сторінці в 2 Кбайт
RET	00100010	1 1 2	PC ← (SP); повернення з підпрограм
RETI	00110010	1 1 2	PC ← (SP); повернення з підпрограми оброблення переривань.
NOP	00000000	1 1 1	PC ← PC + 1; холоста команда
Команди умовних переходів			
JZ rel	01100000	5 2 2	PC ← PC + 2 + rel, якщо A = 0
JNZ rel	01110000	5 2 2	PC ← PC + 2 + rel, якщо A < > 0
JC rel	01000000	5 2 2	PC←PC+2+rel, якщо прапор C=1
JNC rel	01010000	5 2 2	PC←PC+2+rel, якщо прапор C=0
JB bit, rel	00100000	11 3 2	PC ← PC + 3 + rel, якщо (bit) = 1
JNB bit, rel	00110000	11 3 2	PC ← PC + 3 + rel, якщо (bit) = 0
JBCbit, rel	00010000	11 3 2	PC ← PC + 3 + rel, якщо (bit) = 1
CJNE Rn,#DAT,rel	10111rrr	10 3 2	PC ← PC + 3 + rel, якщо Rn < > #DAT; C ← 1, якщо Rn < # DAT; C ← 0, якщо Rn > # DAT
CJNE @Ri, #DAT, rel	1011011i	10 3 2	PC ← PC + 3 + rel, якщо (@Ri) < > # DAT; C ← 1, якщо (@Ri) < # DAT; C ← 0, якщо (@Ri) > # DAT
CJNE A, #DAT,rel	10110100	10 3 2	PC ← PC + 3 + rel, якщо A < > # DAT; C ← 1, якщо A < # DAT; C ← 0, якщо A > # DAT
CJNE A, dir, rel	10110101	8 3 2	PC ← PC + 3 + rel, якщо A < > (dir); C ← 1, якщо A < (dir); C ← 0, якщо A > (dir)
DJNZ Rn, rel	11011rrr	5 2 2	PC ← PC + 2 + rel, якщо Rn – 1 < > 0
DJNZ dir, rel	11010101	8 3 2	PC ← PC + 3 + rel, якщо (dir) – 1 < >0

Закінчення табл. 2.7

Мнемокод	КОП	Т Б Ц	Коментар
Операції з бітами			
CLR C	11000011	1 1 1	C ← 0; скидання прапора C
SETB C	11010011	1 1 1	C ← 1; встановлення прапора C
CPL C	10110011	1 1 1	C ← $\overline{C}$; інверсія прапора C
CLR bit	11000010	4 2 1	(bit) ← 0; скидання прямоадресованого біта
SETB bit	11010010	4 2 1	(bit) ← 1; установлення "біта"
CPL bit	10110010	4 2 1	(bit) ← $\overline{\text{bit}}$; інверсія "біта"
ANL C, bit	10000010	4 2 2	C ← C & (bit)
ANL C, /bit	10110000	4 2 2	C ← C & ($\overline{\text{bit}}$)
ORL C, bit	01110010	4 2 2	C ← C V (bit)
ORL C, /bit	10100000	4 2 2	C ← C V ($\overline{\text{bit}}$); логічне "АБО" C і інверсії прямоадресованого біта в прапор C
MOV C, bit	10100010	4 2 1	C ← (bit) пересилання біта в прапор C
MOV bit, C	10010010	4 2 2	(bit) ← C; пересилання із C в "біт"

Поява трибайтових команд пов'язана з введенням прямої адресації. Всі команди виконуються за один або два машинні цикли. Виняток становлять команди MUL і DIV, які вимагають чотири цикли.

При тактовій частоті 12 МГц тривалість машинного циклу дорівнює 1 мкс.

Більшість двобайтових команд – одноциклові, а всі трибайтові команди – двоциклові. Це пояснюється тим, що за один машинний цикл може вводитися до 2 байт програмного коду.

Усі команди МК51 можна розділити на *п'ять груп:* пересилання (28); логічних операцій (25); арифметичних операцій (24); передачі керування (17); операцій з бітами (булевого процесора) (17).

У процесі виконання команди впливають на ряд прапорів-ознак результату, що входять до складу ССП (PSW). Ознака P встановлюється щоразу, коли приймачем результату служить акумулятор A, включаючи операції пересилання.

Склад операндів МК51 включає: *біти*, 4-бітові цифри, байти, 16-бітові слова.

МК51 має 128 програмно-керованих прапорів користувача. Є також можливість адресації окремих бітів блоку регістрів спеціальних функцій і портів. Для адресації бітів використовується пряма 8-бітова адреса. Непряма адресація бітів неможлива.

Порти і РСФ адресуються тільки прямим способом. Байти пам'яті також можуть адресуватися непрямим чином через адресні регістри (R0, R1, DPTR і PC).

Способи адресації даних:

- *пряма* – у команді міститься адреса місця розташування даних;
- *безпосередня* – у команді міститься сам операнд, який йде за кодом операції;
- *непряма* – у команді наведена адреса, за якою знаходиться операнд;
- *регістрова непряма* – у команді вказані регістри, в яких зберігається адреса, за якою знаходиться операнд; у МК51 використовуються всі 8 біт адресних регістрів R0 і R1;
- *неявна* – у команді адресне поле відсутнє, адресна інформація міститься в коді операції, тобто сама операція несе дані про місце знаходження одного або усіх операндів, використаних в ній.

Регістр "слово стану програми" (ССП) (PSW) включає чотири прапори:

C – перенесення; AC – допоміжне перенесення; OV– переповнення; P – парітет.

C – встановлюється, якщо в старшому біті результату виникає перенесення/ позика; при виконанні операцій множення і ділення С скидається.

AC – встановлюється, якщо при виконанні операції складання/віднімання між тетрадами байта виникло перенесення/позика.

OV – встановлюється, якщо результат складання/віднімання не укладається в семи бітах і старший (восьмий) біт не може інтерпретуватися як знаковий. При виконанні операцій ділення OV скидається і встановлюється тільки при діленні на нуль. При множенні OV встановлюється, якщо результат більше 255.

P – встановлюється і скидається апаратно в кожному циклі команди, фіксує парне/непарне число одиничних бітів в акумуляторі, тобто виконує контроль щодо парності.

Символічна адресація. При використанні асемблера МК51 (ASM51) для отримання об'єктних кодів програм допускається застосування в програмах символічних імен регістрів спеціальних функцій (РСФ), портів та їх окремих бітів.

Для окремих бітів РСФ і портів (така можливість не у всіх РСФ) можна використовувати символічне ім'я біта такої структури:

«ім'я РСФ або порта».«номер біта».

Символічні імена РСФ і портів є зарезервованими словами та їх не треба перевизначати директивами асемблера.

Біти користувача (0–128), що адресуються, в бітах 32–47 внутрішнього ОЗУ.

Група команд передачі даних. Велику частину команд цієї групи, наведеної у табл. 2.7, становлять команди передачі й обміну байтів. Усі команди даної групи не модифікують прапори результату, за винятком команд завантаження PSW і акумулятора (прапор паритету).

Команди логічних операцій виконують операції логічного І, логічного АБО, ВИКЛЮЧАЮЧОГО АБО, очищення, інверсії, зсуву.

Команди арифметичних операцій виконують операції складання, десяткової корекції, віднімання, множення, ділення, інкремента і декремента даних. В операціях цілочисельного множення і ділення без знаку бере участь акумулятор і регістр В. При множенні 8-розрядне значення А помножується на 8-розрядне значення В, а 16-розрядний результат записується в пару ВА. При діленні 8-розрядного значення А на 8-розрядне значення В частка записується в А, а залишок у В.

Група команд операцій з бітами. У командах цієї групи як операнди виступають окремі біти портів, деяких РСФ і 128 програмних прапорів користувача (байти 32–47). Для адресації бітів використовуються пряма 8-розрядна адреса. Непряма адресація бітів неможлива.

Група команд передачі керування включає команди, що забезпечують умовне і безумовне керування, виклик підпрограм і повернення з них, а також порожні операції NOP.

Довгий перехід. Перехід по всьому адресному простору ПП. В команді міститься повна 16-бітова адреса переходу. На практиці рідко виникає необхідність такого переходу.

Абсолютний перехід. Перехід в межах сторінки пам'яті програми розміром 2048 байт. Такі команди містять тільки 11 молодших бітів адреси переходу.

Відносний перехід. Короткий відносний перехід дозволяє передати керування в межах –128 – +127 байт щодо адреси команди, наступної за командою відносного переходу.

Непрямий перехід. Команда JMP @A+DPTR дозволяє передавати керування за непрямою адресою, що обчислюється самою програмою і невідома при виконанні початкового тексту програми.

Умовні переходи. Розгалуження здійснюється за умови рівності або нерівності нулю вмісту акумулятора.

Команда DJNZ зручна при організації циклів. Лічильник циклів організовується в регістрі або прямо в байті, що адресується. При виконанні команди DJNZ проводиться декремент лічильника, перевірка на нуль, і якщо вміст не дорівнює нулю, то відбувається перехід до початку циклу, а при рівності 0 відбувається вихід з циклу.

Позначки для цієї команди повинні знаходитися в межах однієї сторінки пам'яті. Аналізовані ознаки не фіксуються в спеціальних трігерах (окрім прапорів *C* і *F0*), а подаються миттєвими значеннями сигналів у АЛП або на відповідних входах МК51.

Підпрограми. Для виклику їх використовують команду LCALL або ACALL. Ці команди зберігають в стеку адреси повернення в основну програму. Повернення з підпрограми відбувається при виконанні команди RET. Для повернення з підпрограм обслуговування переривання використовують команду RETI, яка відрізняється від команди RET тим, що дозволяє переривання обслугованого рівня.

Асемблер допускає використання узагальненого виду команд JMP і CALL, які в процесі трансляції замінюють оптимальними за форматом командами переходу (AJMP, SJMP, LJMP) або виклику (ACALL і LCALL).

Засоби розробки і відладки програмного забезпечення

До цих засобів відносять програми асемблювання, лінкування і компонування, програми-налагоджувачі – емулятори.

Асемблювання

Асемблер – програма, створена для спрощення завдання написання програм на обчислювальних машинах. Він здійснює задання трансляції символьної програми у виконуваний об'єктний код.

Програма на асемблері складається з рядків трьох типів:

- команди процесора;
- директиви асемблера;
- керуючі параметри.

Команди процесора – мнемонічне кодування машинних команд, які повинен виконати процесор.

Директиви асемблера – псевдокоманди, які при трансляції не породжують кодів машинних команд і використовуються для визначення змінних, структури програми, завдання констант і т. ін.

Керуючі команди – вказівки транслятору на виконання певних дій, наприклад створення лістингу.

Програма асемблювання ASM51 дозволяє використовувати модульний принцип (розбиття складної програми на окремі функціональні частини, тобто модулі).

Після налагодження всіх модулів окремо вони можуть бути зібрані воєдино спеціальною програмою лінкування RL51.

Початкова програма створюється за допомогою редактора текстових файлів і містить коментарі, параметри асемблера і директиви, а також керуючі команди на мові Асемблер.

Далі викликається програма асемблювання ASM51.EXE з вказівкою імені трансльованої програми, наприклад: asm51 myprog.a51.

Результатом роботи асемблера буде наступне:

- об'єктний файл (виконувана форма вихідних даних асемблера, що є абсолютним форматом 16-річного коду), який може бути запрограмований на МК51 і включає необхідну для збору і налагодження інформацію;
- лістинговий файл (запис початкової програми об'єктного коду), в який асемблер вносить діагностичні повідомлення про синтаксичні та інші помилки кодування.

Директиви асемблера

Асемблер має декілька директив, які дозволяють користувачу: встановити значення символьних імен; зарезервувати і ініціалізувати місце пам'яті; керувати розміщенням програм.

Директиви визначення символічних імен дозволяють створювати символічні імена, які можуть застосовуватися для поліпшення читабельності програми. За допомогою цих директив можна визначати символічні імена для посилань на сегменти, адреси даних, адреси бітів, адреси зовнішніх даних:

SEGMENT DATA BIT
EQU IDATA CODE
SET XDATA

Директиви вибору сегментів призначають подальший код або дані у вибраному сегменті доти, поки інший сегмент не буде вибраний:

RSEG CSEG USING
BSEG DSEG
ISEG XSEG

Директиви керування станом асемблера:

- керування лічильником асемблера – ORG;
- міжмодульний зв'язок: PUBLIC, EXTRN, NAME;
- ініціалізація пам'яті:

  DBIT DB
  DW DS

- кінець програми (обов'язкова директива) – END.

Символічні імена можуть містити Усі 28 латинських літер, цифри 0–9, знаки (?, –). Символічні імена не можуть починатися з цифри. В імені можна використовувати до 255 символів, але значущими є лише перший 31 символ.

Мітка – символічне ім'я. Щоб виділити мітку, після неї потрібно ставити двокрапку (:).

Директиви, які не можуть мати мітку:

BIT; EQU; SET; CODE; IDATA; XDATA; DATA; ORG; END; SEGMENT.

Налагоджувач асемблерних програм дозволяє таке:

- завантажити для налагодження HEX-файли, що виробляються трансляторами з мови асемблера (крос-засоби), а також файли чистого двійкового коду, що прочитуються з ПЗП;
- переглянути на екрані дизасембльований текст завантаженої програми, включаючи адреси і коди команд, область імітованого ОЗУ даних, зовнішньої пам'яті, пам'яті програм, вміст усіх регістрів;
- виконати завантажену програму по кроках з прогляданням результатів після кожного кроку і в безперервному режимі з остановкою по крапках;
- внести зміни в завантажену програму в мнемонічних позначеннях мови асемблера, а також в машинних кодах;
- внести зміни у вміст регістрів, прапорів і пам'яті.

Об'єктний файл перед налагодженням на емуляторі слід перетворити в 16-ковий. Для цього може бути використана крос-програма OH.EXE. Формат: oh myprog.obj. В цьому випадку одержимо результат myprog.hex.

Вигляд вікна налагоджувача-емулятора FD51 наводиться нижче на рис. 2.12.

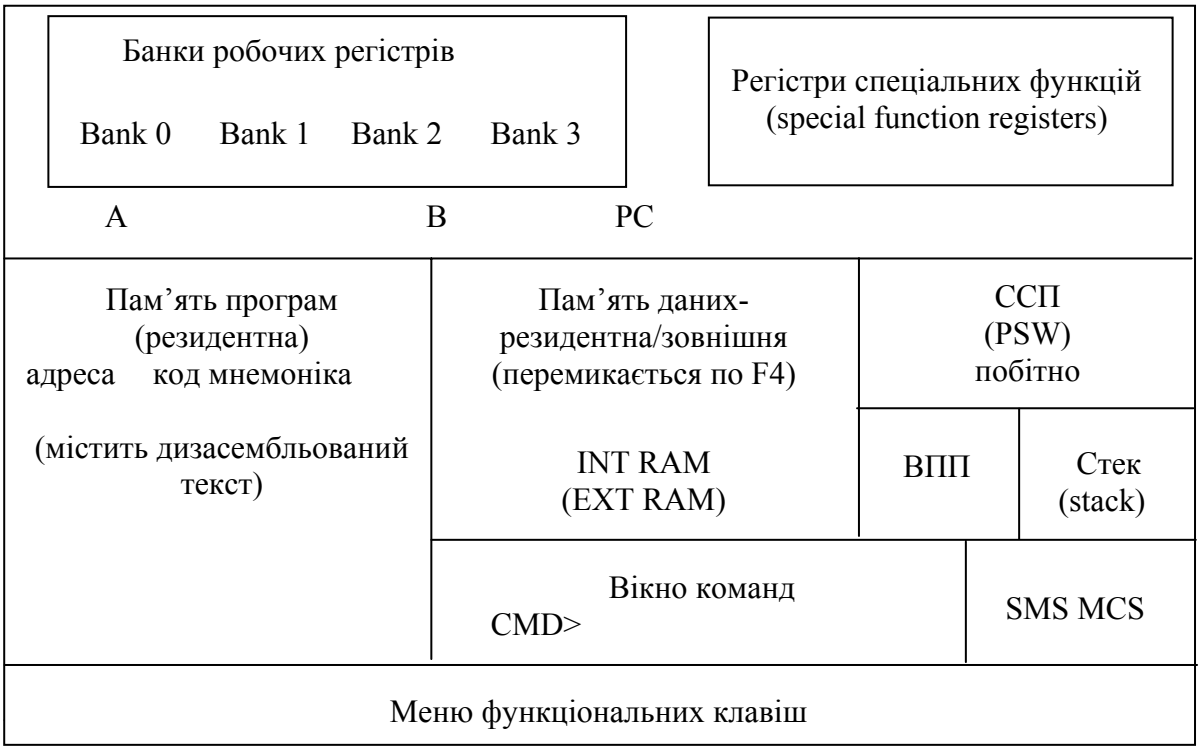


Рисунок 2.12 – Вікно програми налагоджувача-емулятора FD51.EXE

Для швидкого перегортання можна користуватися такими клавішами:

[Home] – перегортає вікно пам’яті даних вгору на одну сторінку;

[End] – перегортає вікно пам’яті даних униз на одну сторінку;

[PgUp] – перегортає вікно пам’яті програм вгору на одну сторінку;

[PgDn] – перегортає вікно пам’яті програм униз на одну сторінку.

Призначення функціональних клавіш, за допомогою яких виконують команди, що найбільш часто зустрічаються, наводиться нижче в табл. 2.8.

Таблиця 2.8 – Призначення функціональних клавіш

F1	Виконує поточну інструкцію завантаженої програми (поточною є інструкція, що підсвічується, у вікні дизасембльованого тексту)
F2	Виконує програму до наступної за адресою за поточною інструкцією
F3	Дозволяє подавати числову інформацію в десятковій, двійковій і шістнадцятковій формах (для регістрів)
F4	Перемикає вікно пам’яті даних з внутрішньої (INT RAM) на зовнішню (EXT RAM) і назад
F5	Установлення крапок переривання
F6	Перемикає форму подання пам’яті у вікні в двійкову і шістнадцяткову
F7	Перегортає вікно пам’яті даних вгору на один рядок
F8	Перегортає вікно пам’яті даних униз на один рядок
F9	Перегортає вікно пам’яті програм угору на один рядок
F10	Перегортає вікно пам’яті програм униз на один рядок

Команди налагоджувача

При описанні команд налагоджувача використовують такі позначення:

- параметри поміщені в кутові дужки, наприклад «адреса»;
- необов’язкові параметри поміщені в квадратні дужки, наприклад [«адреса»];
- усі числові значення повинні мати 16-ковий формат, при цьому не потрібно вказувати букву «h».

H або Ctrl-H – *швидке отримання довідки.*

L [«тип пам’яті»«поч. адреса»] «файл. спец.»[A] – *завантажити файл у пам’ять.* «Тип пам’яті» може бути I, E або P. Відповідно до цього параметра файл завантажується у внутрішню (Int), зовнішню (Ext) або програмну (Pgm) пам’ять. «Поч. адреса» і «тип пам’яті» указується тільки

при завантаженні чистого двійкового коду. При завантаженні файла, виробленого ISIS-II MACRO-ASSEMBLER'ом, потрібно вказати тільки специфікацію файла і ключ /A. Приклад: L I 01F, A:\PGM\T1 – завантажити двійковий файл у внутрішню пам'ять з адреси 01F.

S «тип пам'яті»«поч. адреса»-«кін. Адреса»,«файл. спец.» – *зберегти область пам'яті в дисковому файлі* (взагалі кажучи, у всіх командах як «файл. спец.» допускається будь-яка коректна в DOS специфікація файла, наприклад COM1), «поч.адреса» і «кін.адреса» вказують відповідно початок і кінець області, що зберігається. Збережений командою S файл можна потім знову завантажити командою L. Приклад: S P 20-642, C:\PGMLIB\MYFILE.

PRT «тип пам'яті» «поч. адреса»-«кін. Адреса», [«файл. спец.»] – *роздрукувати дамп області пам'яті в шістнадцятковому форматі*. Якщо не вказано «файл. спец.», то дамп виводиться на принтер.

PRTD «поч. адрес», «кількість команд», [«файл. спец.»] – *роздруковати дизасембльований текст, починаючи з «поч. адреси»*. Виведення за умовчанням на принтер.

R «номер регістра» = «число» – *занести число в регістр поточного банку*. Число повинне бути байтом. Приклад: R4 = FF.

«Ім'я регістра» = «число» – занести число в регістр спеціального призначення. Можна використовувати такі імена: A, B, TH0, TH1, TL0, TL1, DPH, DPL, DPTR, SP, IP, IE, TMOD, TCON, SCON, SBUF, PC. Число для PC і DPTR може бути і двобайтовою величиною. Приклад: SP = 20 DPTR = FF00.

«Ім'я прапора» = «число» – *встановити або скинути прапор в PSW*. Імена прапорів: C, AC, F0, S1, S0, OV, P. Якщо число = 0, то прапор скидається, інакше – встановлюється. Приклад: S1 = 0.

P0 «номер порту» = «число» – *занести число в порт*. Номер порту може бути 0-3. Приклад: P02 = 12.

D «адрес» – *встановити адресу дизасембльованого тексту у вікні*. Приклад: D 0240.

«Тип пам'яті» «адреса»[-«кін. Адреса»] = «число» – *занести число в пам'ять*. Якщо вказана «кін. Адреса», то цим числом заповнюється область пам'яті. Приклад: I 22 = 55 P 0-40 = FF. Можливо виникнення неоднозначності при заповненні деяких елементів пам'яті. Наприклад, ко-

манда «PC = 23» буде сприйнята не як занесення числа 23 в пам'ять програм за адресою 0C, а як команда встановлення лічильника команд (PC) в значення 23. В цьому випадку потрібно явно вказати, що це адреса P0C = 23.

«Тип пам'яті» «адреса».«номер біта» = «число» – *встановити або скинути біт у пам'яті.* «Номер біта» може бути 7–0 (старший біт – 7). Приклад: I 20.6 = 1.

«Ім'я регістру».«номер біта» = «число» – *встановити або скинути біт у регістрі спеціального призначення* (A, B, P00-P03, IP, IE, TMOD, TCON, SCON). Приклад: TMOD.3 = 0.

M «тип пам'яті»«поч. адрес» – *встановити початкову адресу пам'яті у вікні.* Приклад: M I 20 M E 0FF M P 0.

G [«поч. адреса», [«кін. Адреса»]] – виконати програму «поч. адреса» до «кін. Адреса». Якщо «поч. адреса» не вказана, виконання починається з поточної команди (поточна команда виділена білим прямокутником). «кін. Адресу» можна не вказувати, якщо використовуються точки переривання. Програму, що виконується, можна зупинити натисненням будь-якої клавіші. G без параметрів можна ввести натисненням «Alt-F10». Можна вказати тільки кінцеву адресу, але кома повинна бути присутньою. Приклад: G 100-FF0 G, 2200.

T ON [,«файл. спец.»] – *включити трасування програми.* За умовчанням записи трасувань виводяться на принтер.

T OFF – *вимкнути трасування*.

INT «0/1» = «число» – *імітувати високий або низький рівень на входах INT0 або INT1*. Приклад: INT1 = 0.

BA = «адреса» – *встановити нову «точку відліку» для дизасемблювання*. Ця команда корисна при прогляданні таблиць, зашитих у пам'яті програм, коли при дизасемблюванні «назад» невідомо, звідки вести дизасемблювання.

RSTC – *скинути лічильник часу виконання програми.*

QUIT – *вихід в DOS.*

RST – *імітується скидання процесора.*

N – *очищення вмісту відладчика FD51.*

Відразу ж після запуску налагоджувач готовий до прийому команд користувача – курсор знаходиться в командному рядку.

Повноекранне редагування

Перехід у режим повноекранного редагування здійснюється натисненням клавіші [Enter] без введення команди. Тепер можна переміщати курсор по екрану за допомогою клавіш керування курсором і змінювати вміст регістрів, пам'яті і прапори набором чисел на клавіатурі. Можна змінити також початкову адресу дизасембльованого тексту (поточної інструкції) і початкові адреси вікон пам'яті (у перших рядках вікон). Повноекранне редагування можна проводити і при десятковому, і при двійковому поданні інформації на екрані. Під час редагування залишаються доступними всі команди, що вводяться функціональними клавішами. Щоб повернутися в командний рядок, натисніть [Enter] знову. Для швидкого переміщення курсора по екрану можна користуватися клавішами [Tab] і [Shift]-[Tab].

Режим асемблера

Для переходу в режим асемблера (введення команд налагоджуваної програми в мнемонічних позначеннях) потрібно в режимі повноекранного редагування помістити курсор у полі поточної інструкції завантаженої програми. Тепер наберіть мнемоніку (наприклад, «MOV A, # 45») і натисніть [Enter]. Якщо мнемоніка правильна, то відповідні їй коди заносяться в пам'ять програм, а вікно встановлюється на наступну адресу.

При асемблюванні підтримуються імена регістрів спеціального призначення. При виникненні неоднозначності слід числовим значенням передувати нулем.

Для виходу з режиму асемблера натисніть клавішу «Quit» або відведіть курсор з поля поточної інструкції.

Робота з точками переривання

Меню точок переривання викликається клавішею F5. Можна встановити одночасно 8 точок переривання.

Переривання (зупинка) виконуваної програми відбувається при досягненні вказаної в колонці «PC» адреси і виконанні умови «Counter» = = «Occur». «Counter» – це лічильник, значення якого визначає, скільки разів програма повинна пройти через указану адресу, щоб відбулась зупинка. «Occur» показує, скільки разів програма проходила через указану адресу.

Закінчивши редагування, поточні значення точок переривання можна зберегти на диску (клавіша F2). У користувача запрошується номер набору точок переривання (0–9). Інформація записується у файл з ім'ям FD51.BRK. Відновити картинку можна клавішею F1, також указавши її номер.

Для повернення в основне меню слід натиснути клавішу F5.

Визначивши точки переривання, можна запустити програму командою G без параметрів.

При зупинці програми за перериванням видається повідомлення з вказівкою номера точки переривання.

Детальніше робота налагоджувальника розглядається нижче в п. 3.3 (лабораторна робота 3. Складання і налагоджування програм на мові Асемблер).

2.3. Мікроконтролери фірми Atmel

Фірма Atmel виробляє такі серії мікроконтролерів: серію AT89, сумісну з фактичним промисловим стандартом MCS-51™ і серію RISC-мікроконтролерів AT90 власної архітектури [35]. Характерною особливістю мікроконтролерів фірми Atme1 є пам'ять програм, що базується на FLASH-пам'яті. Це забезпечує швидкість програмування і зменшення циклу розробки. Додатково передбачений спеціальний, послідовний інтерфейс, що дозволяє програмувати і налагоджувати мікроконтролери безпосередньо на монтажній платі.

Мікроконтролери серії AT89, сумісні з MCS-51™

Мікроконтролери з системою команд і архітектурою MCS-51™ представляти немає необхідності. Вони широко розповсюджені, безліч фірм випускає їх модифікації і програмне забезпечення для них. Випускає такі мікроконтролери і фірма Atme1 [35].

Відмінною особливістю цих мікроконтролерів є застосування FLASH-пам'яті програм. Ця особливість дозволяє практично миттєво змінювати програмний код мікроконтролера, що істотно скорочує цикл розробки.

Мікроконтролери в корпусі з 40/44 виводами повністю сумісні за виводами з контролерами 80С51 і забезпечують можливість використання напрацьованих програм і прямої заміни.

FLASH-пам’ять програм робить також можливою дистанційну зміну програмного коду вбудованих мікроконтролерів безпосередньо у замовника.

Мікроконтролери серії AT89 мають такі основні особливості:

- 8-розрядний процесор, оптимізований для керування;
- широкі можливості побітового оброблення;
- вбудовану FLASH -память програм;
- вбудовану оперативну пам’ять;
- двонаправлені лінії введення-виведення, що адресуються індивідуально;
- один або декілька 16-розрядних таймерів/лічильників;
- повнодуплексний UART;
- розгалужену структуру переривань;
- вбудований тактовий генератор;
- економічні режими: IDLE і POWER DOWN ;
- вбудовану пам’ять EEPROM (AT89S);
- послідовний інтерфейс SPI (AT89S);
- сторожовий таймер (AT89S).

Склад сімейства мікроконтролерів поданий в табл. 2.9 [35].

Розглянемо детальніше особливості архітектури мікроконтролерів.

Для забезпечення економії споживання енергії мікроконтролери мають два програмно-керовані режими роботи зі зниженою потужністю.

У режимі IDLE процесор вимкнений, тоді як оперативна пам’ять і вбудовані периферійні пристрої продовжують функціонувати. У цьому режимі споживання струму зменшується приблизно на 15 % від споживання повністю активного пристрою.

У режимі POWER DOWN Усі пристрої мікроконтролера вимкнені, проте дані в оперативній пам’яті продовжують зберігатися. В цьому режимі споживання мікроконтролера становить менше 15 мкА і у будь-якому випадку не більше 0,6 мкА. Крім того, мікроконтролери розроблені із застосуванням статичної логіки, яка не вимагає безперервної синхронізації. Тому частота тактового генератора може бути зменшена або він

може бути зупинений в очікуванні події, вимагаючої оброблення. Це також сприяє зниженому енергоспоживанню.

Мікроконтролери мають окремі адресні простори для пам'яті програми і даних. Це дозволяє звертатися до пам'яті даних з 8-бітовими адресами, чим забезпечується швидкість операцій, що виконуються 8-розрядним процесором.

Разом з тим за допомогою регістра ОРТК може згенерувати 16-бітову адресу даних. Таким чином, може бути адресовано до 64 Кбайт зовнішньої пам'яті, для якої контролер генерує сигнали читання і запису RD і WR. Для пам'яті програми забезпечується тільки читання. Безпосередньо адресується до 64 Кбайт пам'яті програм. Для читання зовнішньої пам'яті програм контролер генерує сигнал PSEN. Зовнішні пам'ять програми і пам'ять даних можуть бути об'єднані за логічним «І» для сигналів контролера RD і PSEN.

Після виконання процедури скидання виконання програми починається з адреси 0000Н. З адреси 0003Н розташовуються блоки оброблення переривань, що займають по 8 байт. Якщо процедура оброблення переривання займає не більше 8 байт, вона може розташовуватися в цьому блоці. Процедури оброблення переривань більшого розміру розміщуються в інших областях пам'яті програми, а керування передається їм з блоків оброблення переривань командами безумовного переходу. Якщо переривання в програмі не використовуються, адреси, зарезервовані під блоки оброблення переривань, можуть бути зайняті кодом програми.

Таблиця 2.9 – Основні характеристики МК фірми Atmel

Тип мікроконтролера	AT89 C51	AT89 LV5	AT89 C52	AT89L V52	AT89 C2051	AT89 C151	AT89 S852
Пам'ять програм	4	4	8	8	2	1	8
Пам'ять даних	128	128	256	256	128	64	256
Пам'ять EEPROM	–	–	–	–	–	–	2
Введення/виведення	32	32	32	32	15	15	32
16-бітові таймери / лічильники	2	2	3	3	2	1	3
UART	+	+	+	+	+	–	+
Джерела переривань	6	6	8	–	–	3	9
Біти захисту	3	3	3	3	8	6	3

Нижні адреси пам’яті програми можуть адресувати як вбудовану FLASH-пам’ять програми, так і зовнішню пам’ять залежно від того, чи сполучене виведення керування зовнішнім доступом EA з колом живлення або із загальним дротом відповідно. Зовнішня пам’ять програми може бути адресована 8-розрядною адресою з використанням порту введення-виведення P0 для організації мультиплексованої шини адреси даних або 16-розрядною адресою з використанням портів введення-виведення P0 і P2, причому останній передає старший байт адреси.

Зовнішня пам’ять даних може мати обсяг 64 Кбайт. Адресується вона також як і пам’ять програми з 8- або 16-розрядною адресою. Простір пам’яті даних розділений на 3 блоки: нижні – 128 байт, верхні – 128 байт і область регістрів спеціальних функцій (Specia1 Function Registers – SFR). Внутрішня пам’ять даних завжди адресується одним байтом, що відповідає максимальному обсягу пам’яті 256 байт. Проте застосування різних способів адресації дозволяє використовувати до 384 байт внутрішньої пам’яті. Це пов’язано з тим, що при прямій і непрямій адресації простору зверху FFH адресуються різні області пам’яті (верхні 128 байт і область *SFR*).

У нижніх 128 байт внутрішньої пам’яті перші 32 байт зайняті чотирма банками по 8 регістрів, що адресуються командами програми як R0...R7. Вибір банку, в якому адресуються регістри, забезпечується відповідною установленням двох бітів RS0 і RS1 у регістрі слова стану програми *(PSW)*. Така архітектура дозволяє ефективніше використовувати кодовий простір, оскільки команди звернення до регістрів коротші за команди прямої адресації пам’яті. Наступні після банків регістрів 16 байт внутрішньої пам’яті утворюють блок бітової адресації.

Система команд мікроконтролера включає широкий вибір команд бітової адресації, які можуть безпосередньо адресувати 128 біт в цій області. Біти, що адресуються, мають адреси 00H...7FH.

Усі байти в нижніх 128 байт можуть адресуватися прямим і непрямим методом адресації. До верхніх 128 байт, доступних тільки в контролерах з обсягом пам’яті 256 байт, можна звертатися тільки із застосуванням непрямої адресації. Простір *SFR* (регістрів спеціальних функцій) включає порти введення-виведення, регістри таймерів, регістри керування периферійними пристроями і т.ін. Ця область може адресуватися тіль-

ки прямою адресацією. Структура простору ідентична структурі аналогічного простору контролерів сімейства MCS-51™, проте є додаткові регістри. 16 адрес у просторі 8РК можуть адресуватися і побайтно, і порозрядно. Адреси регістрів, що розрядно адресуються, закінчуються трьома нулями. Адреси бітів цих регістрів мають значення 80H...FFH.

Система команд мікроконтролерів оптимізована для 8-розрядних застосувань керування, забезпечує ряд швидких способів адресації для доступу до внутрішньої оперативної пам'яті. Система команд забезпечує обширну підтримку для однобітових змінних як окремого типу даних, дозволяючи виконувати пряме розрядне маніпулювання в керуванні і логічних системах, які вимагають булевого оброблення.

Слово стану програми *(PSW)* містить біти стану, які відображають поточний стан процесора, і розміщується в просторі *SFR*. Слово стану програми містить біт перенесення *CY*, біт додаткового перенесення (для операцій із *BCD* – двійково-десятковим кодуванням) *AC*, біти вибору банку *RSO* і *RS1*, прапор переповнювання *OV*, біт контролю по парності *P* і два прапори стану, які визначаються користувачем.

Мікроконтролери мають команди прямої і непрямої адресації, регістрові команди і спеціальні команди для деяких регістрів. У останньому випадку код команди безпосередньо указує на регістр, з яким проводитиметься операція. При прямій адресації операнд визначений 8-розрядним полем адреси в команді. Цим методом може бути адресована тільки внутрішня оперативна пам'ять і простір SFR. При непрямій адресації в команді вказаний регістр, який містить адресу операнда. Таким методом може адресуватися як внутрішня, так і зовнішня оперативна пам'ять. Як регістр адреси для 8-розрядних адрес може бути або указівник вершини стека, або регістри R0, або R1 вибраного банку. Регістром адреси для 16-розрядних адрес може бути тільки 16-розрядний регістр-указівник даних DPTR. До банків регістрів, які містять регістри R0...R7, можна звертатися командами, чиї коди операції включають 3-розрядну специфікацію регістра. Команди, які звертаються до регістрів цим способом, забезпечують ефективне використання коду програми, оскільки при цьому в команді відсутній байт адреси. Банк, в якому поточною командою адресується регістр, вибирається відповідним установленням двох бітів у слові стану програми PSW.

Значення константи може прямувати за кодом операції в пам'яті програми. До пам'яті програми можна звертатися тільки через індексну адресацію. 16-розрядний базовий регістр (або *DPTR*, або лічильник команд PC) вказує на початок поточної команди.

Машинний цикл мікроконтролерів складається з 6 станів – S1...S6, кожен з яких займає два такти тактового генератора. Таким чином, тривалість машинного циклу становить 12 тактів тактового генератора або при тактовій частоті 12 МГц – 1 мкс. Команда програми може бути виконана протягом одного або декількох машинних циклів, наприклад, команда MOV X займає два машинні цикли.

Стандартно мікроконтролери мають 5 джерел переривань: 2 зовнішніх переривання, 2 переривання за таймером і переривання від послідовного порту. У деяких контролерах є додаткові джерела переривання відповідно до особливостей їх архітектури. Переривання за кожним з джерел може бути індивідуально дозволено або заборонено шляхом установленням або скидання відповідних бітів у регістрі дозволу переривань *IE*, розташованому в просторі SFR.

У процесі виконання програми стани прапорів переривань прочитуються в п'ятому стані машинного циклу і опитуються в наступному циклі. Для кожного з джерел переривань може бути запрограмований один з двох рівнів пріоритету шляхом установленням або скидання відповідного біта в регістрі пріоритетів переривань *IP*, розташованому в просторі *SFR*. Низькопріоритетне переривання може бути перервано високопріоритетним перериванням, але не іншим низькопріоритетним перериванням. Виконання процедури високопріоритетного переривання не може бути перервано ніяким перериванням. Якщо одночасно надійшли два запити на переривання з різними рівнями пріоритету, то спочатку виконується процедура високопріоритетного переривання. Під час вступу запитів на переривання з однаковим рівнем пріоритету порядок виконання процедур оброблення переривання визначається внутрішньою послідовністю опитування.

У процесі оброблення переривання апаратна процедура LCALL поміщає вміст лічильника команд *PC* в стек і завантажує початковою адресою відповідного блока оброблення переривання. Окрім лічильника команд, автоматично в стеку не зберігаються ніякі інші регістри. За збере-

ження інших необхідних регістрів відповідає програміст. У ряді випадків це дозволяє скоротити час оброблення переривання. В результаті багато функцій оброблення переривань, які є типовими в застосуваннях керування (перемикання виведення порту, перезавантаження таймера або читання буфера послідовного порту), можуть бути завершені швидше, ніж це було б можливо при іншій архітектурі. Багато застосувань вимагають більше двох рівнів пріоритетності переривань, які забезпечуються апаратними засобами мікроконтролерів. У такому разі можливе застосування простого програмного коду, за допомогою якого емулюється третій рівень пріоритетності переривання.

Послідовний порт мікроконтролерів – повнодуплексний, з буфером приймача. Доступ до регістрів прийому і передачі здійснюється через регістр SBUF в просторі SFR. При виконанні запису в цей регістр завантажується регістр передачі, читання забезпечує доступ до регістра прийому. При тактовій частоті мікроконтролера 12 МГц залежно від установленого режиму послідовний порт забезпечує швидкість обміну до 1 Мбод. Проте для забезпечення стандартних швидкостей обміну 1200 – 19200 бод необхідно тактувати мікроконтролер з частотою 11,059 МГц, при цьому для отримання необхідної швидкості обміну використовують один з таймерів.

Мікроконтролери мають два режими роботи зі зниженим споживанням по живленню – id1e і Power Down, перехід в які забезпечується встановленням відповідних бітів у регістрі PCON простору SFR.

У режимі id1e (IDL=1) тактовий генератор продовжує працювати, і забезпечується робота периферійних пристроїв – таймерів, блоку оброблення переривань і послідовного порту; при цьому процесор мікроконтролера зупиняється в очікуванні надходження переривання.

У режимі Power Down (PD =1) зупиняється тактовий генератор мікроконтролера, проте вміст вбудованої пам’яті і регістрів простору SFR зберігається.

Вихід зі стану Power Down можливий тільки при виконанні апаратного скидання. При скиданні переініціалізовуються всі регістри простору SFR, проте вміст внутрішньої пам’яті даних не змінюється. Апаратне скидання запускає також тактовий генератор мікроконтролера. На час перебування МК в режимі Power Down напруга живлення може бути знижена

на 2В. Проте напруга живлення не може бути знижена до того, як МК перейшов в режим Power Down, і повинна бути відновлена перед виконанням скидання. Сигнал скидання повинен бути достатньо тривалим (10 мс) для стабілізації роботи тактового генератора.

2.4. Мікроконтролери фірми Microchip

Мікроконтролери сімейства PIC [23] об'єднують усі сучасні технології мікроконтролерів: електрично програмовані користувачем ППЗУ, мінімальне енергоспоживання, високу продуктивність, добре розвинену RISC- архітектуру, функціональну закінченість і мінімальні розміри. Широка номенклатура виробів забезпечує використання мікроконтролерів у виробах, призначених для різноманітних сфер застосування. Ось короткий перелік виробів, побудованих на базі мікроконтролерів: мікро-АТС, автовідповідачі, АВНи, мобільні телефони, зарядні пристрої, системи сигналізації, вимірювальні прилади, лічильники води, газу і електроенергії, прилади сигналізації, регулятори температури, вологості, тиску і т. ін.

Такий широкий спектр їх застосування став можливий завдяки тому, що мікроконтролери сімейств PIC16/17 мають оптимальні для побудови даних систем архітектуру і параметри.

Нижче, в табл. 2.10 наводиться короткий перелік мікроконтролерів даного сімейства і їх характерні архітектурні особливості.

У табл. 2.10 використовуються такі умовні позначення: OTP – одноразово програмований кристал; ROM – пам'ять програм з масочним ПЗП; ІІ – електрично програмована пам'ять; WDT – сторожовий таймер; I^2C – послідовний інтерфейс; SPI – синхронний послідовний інтерфейс; USART – асинхронний послідовний інтерфейс; BROD – режим перезапуску при зменшенні напруги живлення; Комп – аналоговий компаратор;

Uref – програмоване джерело опорної напруги; АЦП, ЦАП – аналого-цифровий і цифро-аналоговий перетворювачі; Slave port – режим обміну з контроллером по паралельному каналу (DATA0.. .7, WR, RD, CS); *PWM* – широтно-імпульсний модулятор; *PCP* – робота в режимі компаратора/захоплення/шим-модулятора; *CAP* – режим таймера; *ICSP* – програмування на платі; *FLASH*-пам'ять програм;

Таблиця 2.10 – Сімейство мікроконтролерів PIC16/17

Назва	Пам'ять програм	OTP/ ROM	RAM	МГц	I/O	Таймер	CCP/ PWM	Послідовний інтерфейс	Особливості	Корпус
PIC16C177	12288x16	OTP	1024	40	33	3+WDT	2PWM	USART 1^2C/SPI	Slave port 8р., АЦП 8р. BX..BROD, ICSP, (поліп-шений такто-вий генератор)	40P, 40JW, 4LЬ, 44PT
PIC16C178	16384x16	OTP	1536	40	22	3+WDT	2PWM	USART 1^2C/SPI	Slave Port, 8р. АЦП 5BX.. BROD, ICSP, (поліпшений тактовий генератор)	28P, 28S0, 28JW
PIC16C179	16384x16	OTP	1536	40	33	3+WDT	2PWK	USART 1^2C/ SPI	Slave Port, 8р. АЦП 8EX.. .BROD, ICSP (поліпшений тактовий генератор)	40P, 40JW, 44L, 44PT
PIC16C185	8192x16	OTP	1024	40	50	4+WDT	3PWIK	USART 1^2C/SPI	Slave Port, 10р. АЦП 5BX..BROD, ICSP, (поліпшений тактовий генератор)	64SP, 68CL, 68L, в4PT

PIC16 – це сімейство мікроконтролерів, розроблене спеціально для застосування у вбудованих системах керування і контролю, де необхідні висока надійність і низька вартість. Для ілюстрації сказаного вище розглянемо молодшу групу даного сімейства – МК типу PIC16C5XX, подану в табл. 2.11.

Архітектура усіх мікроконтролерів сімейства PIC16/17 побудована на концепції роздільних шин областей пам'яті для даних і для команд (гарвардська архітектура). Шина даних і пам'ять даних (ОЗП) – мають ширину 8 біт, а програмна шина і програмна пам'ять (ПЗП) – 12 біт (PIC16C5XX).

Таблиця 2.11 – Мікроконтролери групи PIC16C5XX

Назва	Частота, Мгц	Виводи	Порти	ПЗП	ОЗП
PIC16C52	4	18	12	384 x 12 ППЗУ	25 x 8
PIC16C54	20	18	12	512 x 12 ППЗУ	25 x 8
PIC16C54A	20	18	12	512 x 12 ППЗУ	25 x 8
PIC16CR54	20	18	12	512 x 12 ПЗП	25 x 8
PIC16CR54A	20	18	12	512 x 12 ПЗП	25 x 8
PIC16CR54B	20	18	12	512 x 12 ПЗП	25 x 8
PIC16C55	20	28	20	512 x 12 ППЗУ	25 x 8
PIC16C56	20	18	12	1024 x 1 2 ППЗУ	25 x 8
PIC16CR56	20	18	12	1024 x 1 2 ПЗП	25 x 8
PIC16C57	20	28	20	2048 x 1 2 ППЗУ	72 x 8
PIC16CR57A	20	28	20	2048 x 12 ПЗП	72 x 8
PIC16CR57B	20	28	20	2048 x 1 2 ПЗП	72 x 8
PIC16СБИА	20	18	12	2040 x 1 2 ППЗУ	72 x 8
PIC16CR58A	20	18	12	2048 x 12 ПЗП	72 x 8
PIC16CR58B	20	18	12	2048 x 12 ПЗП	72 x 8

Така концепція забезпечує просту, але потужну систему команд, розроблену так, що бітові, байтові і регістрові операції працюють з високою швидкістю і з перекриттям за часом вибірок команд і циклів виконання. 12-бітова ширина програмної пам’яті забезпечує вибірку 12-бітової команди в один цикл. Двоступінчастий конвеєр забезпечує одночасну вибірку і виконання команди і дозволяє виконувати кожну команду за один цикл. Якщо команда змінює програмний лічильник (наприклад, GOTO), то для завершення команди потрібен другий цикл.

Синхронізація командних циклів і конвеєрне оброблення команд. Тактова частота з входу OSC1 ділиться на чотири при формуванні чотирьох фаз синхронізації, які не перетинаються: такти Q1, Q2, Q3, Q4. Програмний лічильник (PC) збільшується на кожному такті Q1. Вибірка команди із пам’яті програм і запис її в регістр виконуються на такті Q4. Команда декодується і виконується протягом наступних чотирьох тактів Q1... Q4.

Командний цикл складається з чотирьох тактів. Вибірка і виконання команд проводиться у конвеєрний спосіб, тобто в одному командному циклі декодується і виконується одна команда і вибирається наступна. Конвеєрна обробка дозволяє виконувати кожну команду за один цикл.

Якщо команда змінює програмний лічильник (наприклад, GOTO), то для завершення команди потрібен другий цикл. Цикл вибірки починається з інкрементування програмного лічильника протягом такту Q1. В циклі виконання протягом такту Q1 вибрана і записана команда знаходиться в регістрі команд (IR). Ця команда потім декодується і виконується протягом тактів Q2, Q3 і Q4. Дані із пам'яті зчитуються протягом такту Q2 (читання операнда), а запис виконується під час такту Q4 (запис результату). Усі команди виконуються за один цикл, крім команд розгалуження програми, які виконуються за два цикли, оскільки вибрана команда забирається із конвеєра, а нова команда вибирається і потім виконується. У PIC16/17 програмна пам'ять розташована в середині кристала. Виконувана програма може знаходитися тільки у вбудованому ПЗП.

Основні технічні характеристики мікроконтролерів PIC16C5XX:

- 33 команди і всі команди виконуються за один цикл, окрім команд переходу (2 цикли);
- робоча частота 4 … 20 МГц ;
- 12-бітові команди і 8-бітові дані;
- 7 (8) спеціальних апаратних регістрів SFR
- дворівневий апаратний стек;
- пряма, безпосередня, непряма і відносна адресація даних і команд;
- 12 (20) ліній введення-виведення з індивідуальною настройкою;
- максимальні вхідний/вихідний струми – 10 мА;
- 8-бітовий таймер/лічильник RTCC з 8-бітовим програмованим попереднім дільником;
- автоматичне скидання при включенні;
- таймер включення при скиданні;
- сторожовий таймер (WDT) з власним вбудованим генератором, що забезпечує підвищену надійність;
- EPROM біт секретності для захисту коду;
- біти ідентифікації;
- економічний режим SLEEP;
- вибрані користувачем біти для встановлення режиму збудження вбудованого генератора;
- RC генератор (RC);

- звичайний кварцевий резонатор (XT);
- високочастотний кварцевий резонатор (HS);
- економічний низькочастотний кристал (LP);
- вбудований пристрій програмування EPROM пам'яті програм (використовуються тільки два виводи);
- економічна високошвидкісна КМОП EPROM технологія;
- статичний принцип в архітектурі;
- широкий діапазон напруг живлення і температур;
- комерційний: 2,0...6,0 В; О...+70 °С;
- промисловий: 2,0...6,0 В; –40...+70 °С;
- низьке енергоспоживання;
- 2 мА (типово для 5 В, 4 МГц),
- 15 мкА (типово для 3 В, 32 кГц);
- 1 мкА (у режимі очікування при 3 В);

Регістри пам'яті даних (ОЗП) розділяються на дві функціональні групи: спеціальні регістри і регістри загального призначення. Спеціальні регістри включають регістр таймера/лічильника реального часу (TMRO), лічильник команд (PC), регістр стану (STATUS), регістри введення/виведення (PORT) і регістр непрямої адресації (PSR). Крім того, спеціальні регістри керують конфігурацією портів введення/виведення і режимів попереднього дільника. Регістри загального призначення використовуються програмою для зберігання змінних на розсуд користувача. У мікроконтролерах сімейства PIC16C5X існує пряма і непряма адресація всіх регістрів і елементів пам'яті. Усі спеціальні регістри і лічильник команд також відображаються на пам'яті даних.

Мікроконтролери PIC16C5X мають ортогональну (симетричну) систему команд, що дозволяє виконувати будь-яку операцію з будь-яким регістром, використовуючи будь-який метод адресації. Це полегшує програмування для них і значно зменшує час, необхідний на навчання роботи з ними.

У мікроконтролерах PIC16C5X є 8-розрядний арифметико-логічний пристрій (АЛП) і робочий регістр W. АЛП виконує складання, віднімання, зміщення, бітові і логічні операції.

У командах, що мають два операнди, одним з операндів є робочий ре-гістр W. Другий операнд може бути константою або вмістом будь-

якого регістра ОЗП. У командах з одним операндом операнд може бути вмістом робочого регістра або вмістом будь-якого регістра ОЗП.

Для виконання всіх операцій АЛП використовується робочий регістр W, який не може бути прямо адресований. Залежно від результату виконання операції можуть змінитися значення бітів перенесення C, десяткового перенесення DC і нуля Z в регістрі стану STATUS. При відніманні біти C і DC працюють як біти займу і десяткового займу відповідно. Регістр STATUS доступний для будь-якої команди так само, як і будь-який інший регістр. У його склад входять так само біти T0 і PD, які встановлюються апаратно і не можуть бути змінені програмно. Дані розряди встановлюються у відповідні стани при вмиканні живлення, скиданні і переході в режим SLEEP. Проводячи їх програмне опитування, можна визначити спосіб запуску програми.

Сімейство мікроконтролерів PIC16C5X містить дворівневий апаратний стек. При виконанні команди звернення до підпрограми у вершину стека завантажується лічильник команд, заздалегідь збільшений на одиницю. Одночасно старе значення з вершини стека копіюється в стек рівня 2.

Однією з основних особливостей портів введення/виведення є те, що програма може зчитувати і записувати дані в них аналогічно регістрам загального призначення. При читанні завжди прочитується дійсний стан виводів незалежно від того, запрограмовані окремі розряди як входи або як виходи. Після скидання всі розряди програмуються як входи. Виходи портів є защіпки, і їх стан не міняється до наступного запису в порт. Установлення режиму кожного розряду в усіх портах проводиться за допомогою встановлення відповідних розрядів в регістрах керування режимами портів TRIS (TRISA, TRISB або TRISC).

Модуль таймера (TMR0) у даних мікроконтролерах має такі особливості:

- 8-розрядний таймер/лічильник, доступний до читання і запису;
- 8-розрядний програмований попередній дільник, який може бути програмно-підключений або до таймера, або до таймера WDT;
- внутрішнє або зовнішнє тактування, при цьому може бути вибраний фронт тактуючого імпульсу.

Таймер має два режими роботи: режим таймера і режим лічильника. У режимі таймера TMR0 збільшується в кожному командному циклі

(якщо немає попереднього дільника). У режимі лічильника TMR0 збільшується за кожним перепадом 1/0 або 0/1 на виведенні TOCK1. Перепад, що збільшує значення TMR0, вибирається установленням відповідного біта в службовому регістрі OPTION, який програмно доступний за записом. У цьому ж регістрі проводиться встановлення режиму роботи попереднього дільника (TMR0/WDT) і значення коефіцієнта ділення.

Сімейство мікроконтролерів PIC16C5X має набір спеціальних функцій, призначених для розширення можливостей системи, мінімізації вартості, виключення навісних компонентів, забезпечення мінімального енергоспоживання і захисту коду програми від прочитування. До складу даних функцій входять:

- вибір типу генератора;
- таймер скидання (DRT);
- сторожовий таймер (WDT);
- режим зниженого енергоспоживання (SLEEP);
- захист коду програми від прочитування;
- біти ідентифікації.

Мікроконтролери сімейства PIC16C5X, окрім PIC16C52, мають вбудований сторожовий таймер WDT, який може бути вимкнений тільки через біт конфігурації, що задається при програмуванні. Для підвищення надійності він працює від власного RC- генератора. Таймер скидання ВКТ призначений для підтримки контролера в скинутому стані протягом 18 мс після ввімкнення живлення для стабілізації роботи генератора. Наявність цих таймерів дозволяє в багатьох застосуваннях відмовитися від схеми зовнішнього скидання.

Режим зниженого енергоспоживання SLEEP призначений для забезпечення дуже малого струму споживання в режимі очікування (менше 1мкА при вимкненому сторожовому таймері). Вихід з режиму SLEEP можливий за зовнішнім сигналом скидання або після закінчення витримки сторожового таймера. Можливість вибору типу генератора дозволяє ефективно використовувати мікроконтролери сімейства в різних застосуваннях. Використання RC-генератора дозволяє зменшити вартість системи, а LP-генератор на низькочастотному кварцевому резонаторі скорочує енергоспоживання. Захист коду програми і встановлення коду ідентифі-

кації проводяться шляхом установлення відповідних розрядів у слові конфігурації при програмуванні мікроконтролерів.

Для розробки і налагоджування програмних модулів вільно розповсюджується асемблер MPASM, емулятор MPSIM, інтегрована система налагоджування для Windows MPLAB, а також велика кількість добре документованих прикладів застосування мікроконтролерів PIC в різних сферах з вихідними текстами.

Для апаратної підтримки режиму програмування мікросхем є різні типи програматорів, що підключаються до комп'ютерів типу IBM PC. Існують типи програматорів, які можна підключати безпосередньо до робочої плати контролера, що значно прискорює налагодження. Програмування мікроконтролерів проводиться через послідовний канал, який використовує два розряди порту введення-виведення. Режим програмування задається шляхом установлення на виводі скидання мікроконтролера напруги + 12 В.

МК сімейства PIC16CXX в порівнянні з іншими 8-розрядними мікроконтролерами такого ж класу дозволяють зменшити програму 2:1, збільшити швидкодію 4:1 і ідеально підходять для дешевих додатків, що вимагають аналоговий інтерфейс.

PIC16X7XX – сімейство дешевих, високоефективних, 8-розрядних КМОП мікроконтролерів з вбудованим аналого-цифровим (analog-to-digital (A/D)) перетворювачем. Серед мікроконтролерів PIC16CXX дане сімейство займає середнє положення.

Для того щоб зменшити кількість зовнішніх компонентів і таким чином знизити вартість, підвищити надійність системи і зменшити споживання, сімейство мікроконтролерів PIC16X7XX має додаткові можливості.

У МК PIC16X7XX є чотири режими генератора:

- RC-генератор на одному контакті забезпечує низьку вартість;
- LP-генератор забезпечує мінімальне споживання;
- XT-генератор забезпечує стандартне рішення;
- HS-генератор служить для високочастотних застосувань.

Режим зупинки дозволяє різко зменшити споживання. Збудження з режиму зупинки може здійснюватися за допомогою зовнішніх і внутрішніх переривань і скидань.

Високонадійний сторожовий таймер зі своїм RC-генератором забезпечує захист від зациклення програми.

Варіант мікросхем CERDIR з ультрафіолетовим стиранням ідеально підходить для розробки і налагодження програми, тоді як одноразово програмовані варіанти рентабельні для випуску продукції в будь-якому обсязі. Мікросхеми з FLASH-пам'яттю програм дозволяють здійснювати розробку і налагодження програм на готовому пристрої і не вимагають заміни мікроконтролера після закінчення налагодження. Сімейство мікроконтролерів PIC16X7XX пристосоване для застосування у віддалених пристроях захисту і датчиках, для приладів керування і автомобілів.

Технологія програмованого ПЗП робить налагодження прикладних програм швидкою і надзвичайно зручною. Малогабаритні корпуси мікросхем роблять це сімейство мікроконтролерів досконалим для всіх додатків без обмежень.

Низька ціна, мала споживана потужність, висока ефективність, зручність при використанні і гнучкість I/O роблять PIC16X7XX універсальним навіть в областях, де використання мікроконтролерів раніше не розглядалося (наприклад, функції таймера, послідовний зв'язок, збирання і порівняння даних, функції ШІМ і застосування зі співпроцесором).

Сімейство мікроконтролерів PIC16X7XX є версією, розширеної архітектури МК PIC16C5X. Усі мікроконтролери сімейства PIC16CXX, що випускаються в однакових корпусах, сумісні за контактами.

Програма для PIC16C5X може бути легко перенесена в пристрої сімейства PIC16X7XX. Щоб перетворити програму, написану для PIC16C5X, в програму для PIC16X7XX, необхідно виконати такі кроки:

1) видалити всі операції вибору сторінок пам'яті програми (операції над бітами PA2, PA1, PAO) для команд CALL і GOTO;

2) повторно проглянути Усі обчислені операції для переходів (запис в PC, складання з PC і т. ін.), щоб переконатися, що біти сторінки встановлені правильно згідно з новою структурою;

3) виключити всі перемикання сторінок пам'яті даних, перевизначити змінні в даних, щоб перерозподілити їх;

4 перевірити всі записи в регістрах STATUS, OPTION і SFR, оскільки вони змінилися;

5) замінити вектор скидання, оскільки він став 0000h.

Варіанти пристроїв PIC16X7XX

Мікроконтролери сімейства PIC16X7XX випускають для різних частотних діапазонів і в різних варіантах корпусів. Залежно від застосування і промислових вимог, користуючись табл. 2.12, можна вибрати необхідний варіант пристрою.

Для сімейства PIC16X7XX існують декілька типів позначення пристроїв у номері:

1. C – як, наприклад, PIC16C74. Ці пристрої мають програмований ПЗП пам'яті програм і функціонують у стандартному діапазоні напруг живлення.

2. LC – як, наприклад, PIC16LC74. Ці пристрої мають програмований ПЗП пам'яті програм і функціонують в розширеному діапазоні напруг живлення.

3. CR – як, наприклад, PIC16CR72. Ці пристрої мають масочне ПЗП пам'яті програм і функціонують у стандартному діапазоні напруг живлення.

4. F – як, наприклад, PIC16F73. Ці пристрої мають FLASH-пам'ять програм, що допускає до 100 циклів стирання і запису.

Пристрої з ультрафіолетовим стиранням. Варіант пристрою з ультрафіолетовим стиранням, що випускається в корпусах CERDIP, оптимальний для розроблення макета і налагодження програмного забезпечення. Записаний в ПЗП варіант програми може бути знищений, а пристрій перепрограмований для будь-якого з режимів генератора.

Програматори PICSTART® Plus і PRO MATI® II здійснюють програмування усіх мікрокристалів PIC16X7XX.

Одноразово програмовані пристрої випускаються в пластмасових корпусах і дозволяють користувачу програмувати їх один раз. На додаток до пам'яті програми необхідно також програмувати слово конфігурації.

Пристрої з FLASH пам'яттю програм випускаються за ціною одноразово програмованих пристроїв. Їх доступність особливо корисна, коли необхідна гнучкість при частих модифікаціях програми і малих обсягах виробів.

Таблиця 2.12 – Мікроконтролери сімейства PIC16X7XX..

Прилад	Частота, МГц	Пам'ять програм			RAM, байт	Контакти I/O	Периферія						Особливості				Тип корпусу	Примітка
		PROM	ROM	FLASH			ADC	Vref	TMR	PWM	PCP	PORT	ICSP	BOR	PLVD	WDT		
PIC16C71	20	1K x 14	–	–	36	13	4 (8біт)	–	1-8 біт	–	–	–	+	–	–	+	18JW, 18P, 18SO	Струм через контакти I/O 25 мА
PIC16C710	20	512 x 14	–	–	36	13	4 (8біт)	–	1-8 біт	–	–	–	+	+	–	+	18JW, 18P, 18SO, 20SS	Струм через будь-який контакт I/O 25 мА
PIC16C711	20	1K x 14	–	–	68	13	4 (8біт)	–	1-8 біт	–	–	–	+	+	–	+	18JW, 18P, 18SO, 20SS	Струм через будь-який контакт I/O 25 мА
PIC16C712	20	1K x 14	–	–	128	13	4 (8біт)	–	2-8 біт, 1-16 біт	1	1	–	+	+	–	+	18JW, 18P, 18SO, 20SS	Струм через будь-який контакт I/O 25 мА
PIC16C715	20	2K x 14	–	–	128	13	4 (8біт)	–	1-8 біт	–	–	–	+	+	–	+	18JW, 18P, 18SO, 20SS	Струм через будь-який контакт I/O 25 мА
PIC16C716	20	2K x 14	–	–	128	13	4 (8біт)	–	2-8 біт, 1-16 біт	1	1	–	+	+	–	+	18JW, 18P, 18SO, 20SS	Струм через будь-який контакт I/O 25 мА
PIC16C717	20	2K x 14	–	–	256	16	6 (10біт)	1	2-8 біт, 1-16 біт	1	ECCP	SPI/MI^{2}C	+	PBOR	+	+	18JW, 18P, 18SO, 20SS	Внутрішній генератор 4 МГц
PIC16C72	20	2K x 14	–	–	128	22	5 (8біт)	–	2-8 біт, 1-16 біт	1	1	SPI / I^{2}C	+	+	–	+	28JW, 28SP, 28SO, 28SS, 28ML	Струм через будь-який контакт I/O 25 мА
PIC16CR72	20	–	2K x x 14	–	128	22	5 (8біт)	–	2-8 біт, 1-16 біт	1	1	SPI / I^{2}C	–	+	–	+	28SP, 28SO, 28SS	Струм через будь-який контакт I/O 25 мА

Продоіження табл. 2.12

Прилад	Частота, МГц	Пам'ять програм			RAM, байт	Контакти I/O	Периферія						Особливості				Тип корпусу	Примітка
		PROM	ROM	FLASH			ADC	Vref	TMR	PWM	PCP	PORT	ICSP	BOR	PLVD	WDT		
PIC16C73	20	4K x 14	–	–	192	22	5 (8біт)	–	2-8 біт, 1-16 біт	2	2	SPI / I^2C, USART	+	+	–	+	28JW, 28SP, 28SO, 28SS, 28ML	Струм через будь-який контакт I/O 25 мА
PIC16C74	20	4K x 14		–	192	33	8 (8біт)	–	2-8 біт, 1-16 біт	2	2	SPI / I^2C, USART, PSP	+	+	–	+	40JW, 40P, 44L, 44PQ, 44PT	Струм через будь-який контакт I/O 25 мА
PIC16C745	24	8K x 14	–	–	256	22	5 (8біт)	–	2-8 біт, 1-16 біт	2	2	USART, USB	+	+	–	+	28JW, 28SP, 28SO	Модуль USB, PLL на 4
PC16C765	24	8K x 14	–	–	256	33	8 (8біт)	–	2-8 біт, 1-16 біт	2	2	USART, USB, PSP	+	+	–	+	40JW, 40P, 44L, 44PT	Модуль USB, PLL на 4
PIC16C76	20	8K x 14	–	–	368	22	5 (8біт)	–	2-8 біт, 1-16 біт	2	2	SPI / I^2C, USART	+	+	–	+	28JW, 28SP, 28SO	Струм через будь-який контакт I/O 25 мА
PIC16C77	20	8Kx14	–	–	368	33	8 (86ит)	–	2-8 біт, 1-16 біт	2	2	SPI / I^2C, USART,	+	+	–	+	40JW, 40P, 44L, 44PQ, 44PT	Струм через контакти I/O 25 мА
PIC16C770	20	4Kx14	–	–	256	16	6 (12біт)	1	2-8біт, 1-16біт	1	ECCP	SPI/MI^{2}C	+	PBOR	+	+	20JW, 20P, 20SO, 20SS	Внутрішній генератор 4МГц
PIC16C771	20	4Kx14	–	–	256	16	6 (12біт)	1	2-8 біт, 1-16біт	1	ECCP	SPI/MI^{2}C	+	PBOR	+	+	20JW 20P, 20SO, 20SS	Внутрішній генератор 4МГц
PIC16C773	20	4Kx14	–	–	256	22	6 (126іт)	1	2-8 біт, 1-16 біт	2	2	SPI/MI^{2}C, AUSART	+	PBOR	+	+	28JW, 28SP, 28SO, 28SS	Струм через контакти I/O 25 мА
PIC16C774	20	4Kx14	–	–	256	33	10 (12біт)	1	2-8 біт, 1-16 біт	2	2	SPI/MI^{2}C, AUSART, PSP	+	PBOR	+	+	40JW, 40P,44L, 44PQ,44PT	Струм через контакти I/O 25 мА
PIC16C781	20	1Kx14	–	–	128	16	8 (8біт)	1	2-8біт, 1-16 біт	-	-	–	+	PBOR	+	+	20JW, 20P, 20SO, 20SS	2 компаратори, підсилювач, ЦАП

Закінчення табл. 2.12

Прилад	Частота, МГц	Пам'ять програм			RAM, байт	Контакти I/O	Периферія						Особливості				Тип корпусу	Примітка
		PROM	ROM	FLASH			ADC	Vref	TMR	PWM	PCP	PORT	ICSP	BOR	PLVD	WDT		
PIC16C782	20	2Кх14	–	–	128	16	8 (8біт)	1	2-8 біт, 1-16 біт	-	-	–	+	PBOR	+	+	20JW, 20P, 20SO, 20SS	2 компаратори, підсилювач, ЦАП
PIC16F72	20	–	–	2Кх14	128	22	5 (8біт)	–	2-8 біт, 1-16 біт	1	1	SPI / I^2C	+	+	–	+	28SP, 28SO, 28SS, 28ML	Струм через контакти I/O 25 мА
PIC16F73	20	–	–	4Кх14	192	22	5 (8біт)	–	2-8 біт, 1-16 біт	2	2	SPI / I^2C, USART	+	+	–	+	28SP, 28SO, 28SS, 28ML	Читання пам'яті програм
PIC16F74	20	–	–	4Кх14	192	33	8 (8біт)	–	2-8 біт, 1-16 біт	2	2	SPI/I^2C, USART, PSP	+	+	–	+	40P, 44L, 44PT	Читання пам'яті програм
PIC16F76	20	–	–	8Кх14	368	22	5 (8біт)	–	2-8 біт, 1-16 біт	2	2	SPI / I^2C, USART	+	+	–	+	28SP, 28SO, 28SS, 28ML	Читання пам'яті програм
PIC16F77	20	–	–	8Кх14	368	33	8 (8біт)	–	2-8 біт, 1-16 біт	2	2	SPI/I^2C, USART, PSP	+	+	–	+	40P, 44L, 44PT	Читання пам'яті програм

Умовні позначення:

PROM – *програмувальна пам'ять програм*
ROM – *масочне ПЗУ пам'яті програм*
RAM – *пам'ять даних*
I/O – *введення-виведення*
ADC – *аналого-цифровий перетворювач*
Vref – *джерело опорної напруги*
TMR –*таймери*
USART – *універсальний синхронно-асинхронний приємопередавач*
PWM – *широтно-імпульсний модулятор*
CCP – *модуль порівняння, накопичення й ШІМ*
ECCP – *модернізований* PCP
PORT – *порти введення-виведення*
SPI – *синхронний послідовний інтерфейс*
I^2C – *інтерфейс інтегральних схем*
MI^2C – *ведучий I^2C*
ICSP – *внутрішньосхемне послідовне програмування*
AUSART– *адресований USART*
PSP – *паралельний ведений порт*
USB – *універсальна послідовна шина*
BOR – *скидання при зниженні живлення*
PBOR – *програмувальне скидання при зниженні живлення*
PLVD – *програмувальний контроль напруги живлення*
WDT –*сторожовий таймер*
PLL – *схема множення частоти генератора*

Пристрої з масковим ПЗП програмуються заводом-виготовлювачем у процесі виробництва. Пристрої з масковим ПЗП виготовляються для користувачів, які випускають велику кількість продукції без додаткових операцій програмування, оскільки програма відпрацьована і не вимагає змін. Ці пристрої ідентичні одноразово програмованим, але вся пам'ять програм і слово конфігурації вже запрограмовані виготовлювачем.

Структурна схема мікроконтролерів P1C16C710/71/711/715 подана на рис. 2.13.

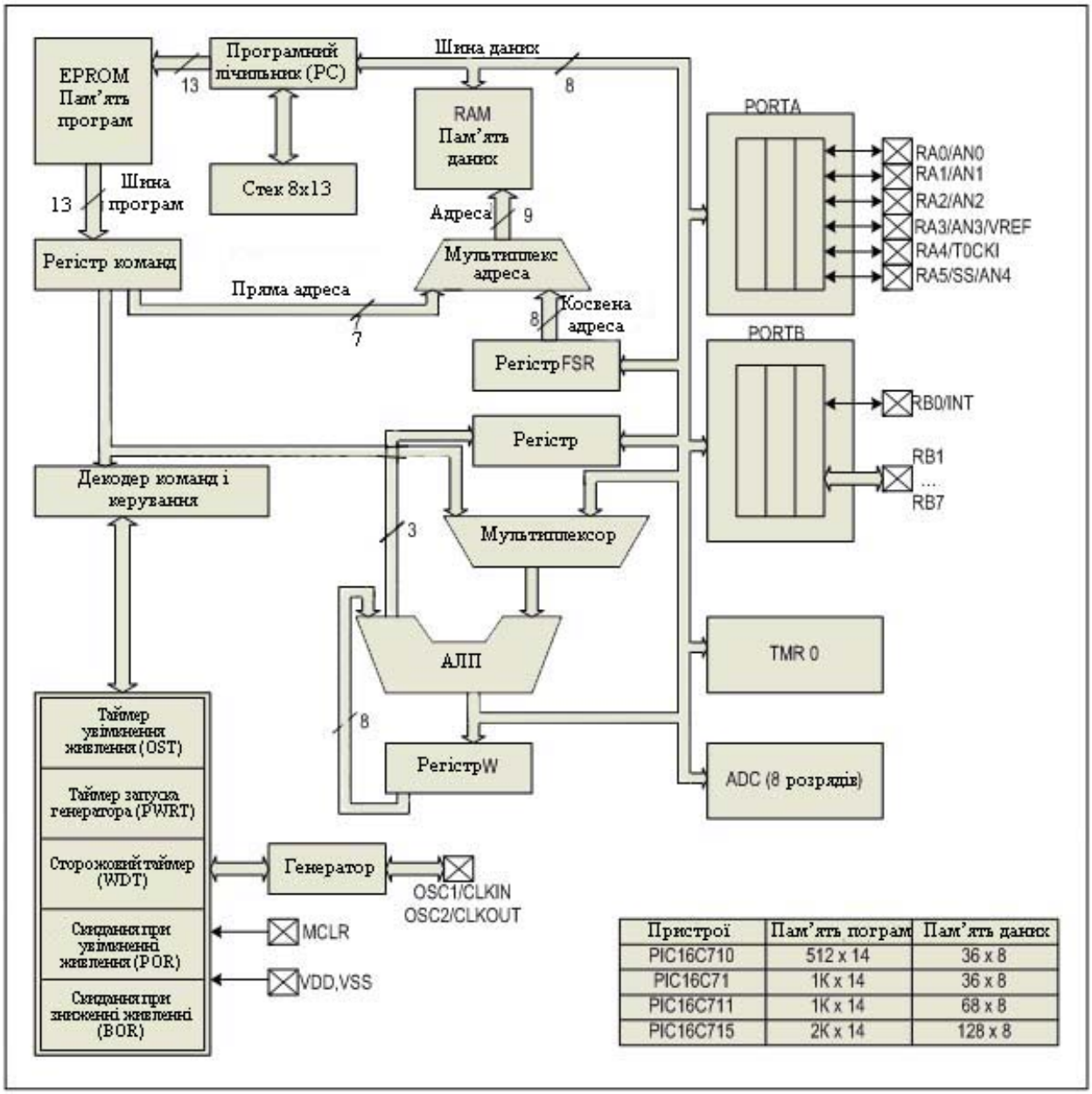


Рисунок 2.13 – Структурна схема PIC16C710/71/711/715

Пристрої PIC16C710/71 мають 36 байт пам'яті даних (RAM), PIC16C711 – 68 байт і PIC16C715 – 128 байт. У кожного з цих пристроїв по 13 контактів I/O (PORTA і PORTB). Пристрої PIC16C712/716 мають: 13 контактів I/O (PORTA і PORTB); два 8-розрядних таймери (TMR0 і TMR2) і один 16-розрядний (TMR1); модуль порівняння/накопичення ШІМ (CCPI); 8-розрядний АЦП з чотирма мультиплексованими вхідними каналами.

Крім того, до складу мікроконтролерів входять такі периферійні пристрої:

• 8-розрядний лічильник/таймер (TMR0);

• 8-розрядний A/D перетворювач з 4 мультиплексованими вхідними каналами.

Пристрої PIC16C717/770/771 (рис. 2.14) мають 256 байтів пам'яті даних, 16 контактів I/O (PORTA і PORTB). Вони містять такі периферійні пристрої:

• два 8-розрядних (TMRO і TMR2) і один 16-розрядний (TMR1) таймер;

• вдосконалений модуль порівняння/накопичення/ШІМ (ECCP), що дозволяє керувати силовими ключами, мостовою схемою (4 канали), напівмостовою схемою (2 канали) або одноканальною схемою з 10-розрядним дозволом на частоті 20 кГц;

• синхронний, послідовний порт, який може функціонувати як трипровідний послідовний периферійний інтерфейс (SPI) або двопровідна шина I^2C з апаратною реалізацією режиму ведучої шини (MI^2C);

• швидкодіючий A/D-перетворювач з 6 мультиплексованими вхідними каналами і внутрішнім керованим джерелом опорної напруги, причому PIC16C717 має 10-розрядний ADC, а PIC16C770/771 – 12-розрядний;

• вбудований RC-генератор тактових імпульсів, частота якого може бути задана при програмуванні 4 МГц або 37 кГц;

Система команд

Кожна команда PIC16X7XX є 14-розрядним словом, розділеним на код операції, який визначає тип команди, і один або більше операндів, над якими виконуються операції.

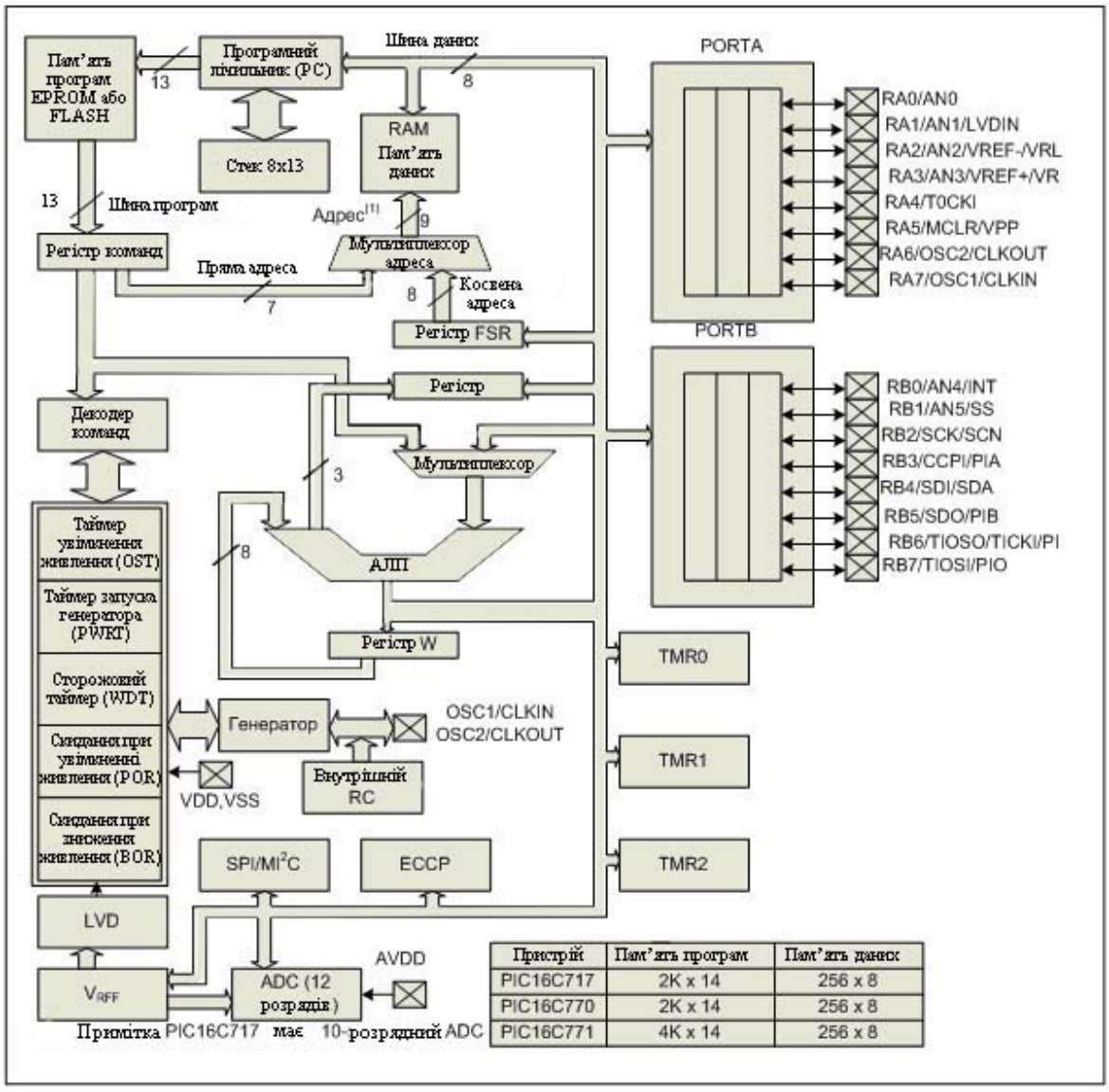


Рисунок 2.14 – Структурна схема PIC16C7117/770/771

Система команд високоортогональна і згрупована в три базові категорії: байтові операції, операції над розрядами, операції з константами (літералами) і операції керування. У байтових операціях «f» – указівник регістра, а «d» – указівник адресата результату. Указівник регістра визначає, який регістр повинен брати участь в операції. Указівник результату адресата визначає, куди повинен бути поміщений результат операції. Якщо «d» = 0, результат поміщається в регістр W. Якщо «d» = 1, результат поміщається в регістр, визначений в команді. У операціях над розрядами «b» – указівник розряду, над яким виконується операція, «f» – указівник регістру, в якому знаходиться операнд і куди поміщається результат.

У операціях з константами і командах керування «k» є восьми- або одинадцятирозрядне постійне число або значення літерала (мітка). Усі команди виконуються за один командний цикл. Але якщо результат перевірки умови істинний або лічильник програм змінений у результаті виконання команди, то в цьому випадку команди виконуються за два цикли, другий цикл виконується як операція NOP. Один цикл команди складається з чотирьох періодів генератора.

Усі команди виконуються за один цикл, крім команд розгалуження програми, які виконуються за два цикли, тому що вибрана команда вивантажується із конвеєра, а нова команда вибирається, а потім виконується. Перелік команд та їх структура наведені у додатку 3.

Докладний опис структурних схем, пристроїв які в них входять, особливостей функціонування в різних режимах, системи команд і застосування МК сімейства PIC16X7XX наведено в [23].

2.5. Мікроконтролери фірми Motorola

Мікроконтролери фірми Motorola [24] займають особливе місце серед аналогічної продукції інших фірм через низку обставин.

По-перше, фірма Motorola пропонує *найширшу в світі номенклатуру* мікроконтролерів, що охоплює практично всі області застосування і включає близько 300 моделей – від простих дешевих до високопродуктивних 32-розрядних мікроконтролерів з R1SC-ядром і потужною периферією [24]. Як наслідок, користувач має можливість вибрати для свого застосування оптимальну модель мікроконтролера щодо набору вбудованих функцій, мінімізуючи число компонентів у системі, і щодо економічних параметрів, що відповідають обсягу і особливостям виробництва.

Другою важливою особливістю мікроконтролерів (і решти продукції) фірми Motoro1a є їх *висока якість і надійність*. Будучи традиційним постачальником військово-промислового і аерокосмічного комплексів, а також автомобільної промисловості США, що висувають підвищені вимоги до надійності компонентів, ця фірма розробила і продовжує розвивати спеціальну програму підвищення якості продукції. Заслуги фірми в цій області відзначені національною нагородою США «За вищу якість продук-

ції», а також численними нагородами як кращому постачальнику від таких великих компаній, як General Motors, Ford, Chrysler, Bosch та ін. [24].

Ці дві важливі особливості, а також наявність широкого вибору програмних і апаратних засобів підтримки розробки, доступних як від фірми Motoro1a, так і від безлічі інших фірм, докладна довідкова література і ін-формація про застосування, зокрема доступна мережею Інтернет, забезпечують можливість отримання безкоштовних технічних консультацій і програмного забезпечення. Ці та інші переваги використання продукції, що стала фактично промисловим стандартом, забезпечують фірмі Motoro1a стійке перше місце з продажу мікроконтролерів у світі протягом ряду останніх років.

На жаль, мікроконтролери фірми процесорній архітектурі 6800 і відрізняються набором периферійних функцій. Це означає, що застосування будь-якого мікроконтролера цього сімейства відкриває користувачу можливість використовувати набутий досвід при створенні нових пристроїв як із застосуванням інших мікроконтролерів з обширного сімейства HC05, так і на основі продуктивнішого, але програмно-сумісного сімейства HC08.

До складу мікроконтролерів сімейства HC05 входять: ПЗП усіх типів, ОЗП, таймери, АЦП, ШІМ, контролери ЖКІ й інших дисплеїв, послідовні інтерфейси і багато інших пристроїв. Усі представники сімейства HC05 мають версії зі зниженим живленням і розширеним температурним діапазоном і випускаються в найрізноманітніших корпусах.

Сімейство HC08

Сімейство HC08 є наступним кроком у розвитку програми замовлених мікроконтролерів фірми Motoro1a для масових додатків і характеризується підвищеною в 5...10 разів продуктивністю процесорного ядра, сумісного за системою команд з ЦПУ HC05. Сімейство HC08 підтримує ефективні додаткові команди і методи адресації, а також такі нові функції, як прямий доступ до пам’яті, технологія нечіткої логіки і елементи цифрового оброблення сигналів.

При цьому статичне процесорне ядро повністю оптимізоване для роботи зі зниженою напругою живлення і дозволяє гнучко керувати споживанням за допомогою вбудованого синтезатора тактової частоти.

Сімейство HC08 є першим 8-розрядним сімейством *з визначуваною користувачем архітектурою* на базі набору стандартних модулів, що значно прискорює цикл розробки нового замовленого мікроконтролера.

Набір модулів у даний час включає різні типи ПЗП і ОЗП, таймери, послідовні інтерфейси, АЦП, контролер ЖКІ, контролер ПДП, силові і високовольтні ключі і т. ін. Перші представники цього сімейства з'явилися в 1994р., зараз до складу сімейства входять близько 20 моделей. Нова програма «Замовлений мікроконтролер за 7 днів», введена фірмою Motoro1a в 1996 р., дозволила радикально скоротити цикл розробки нових мікроконтролерів сімейства HC08, що, безумовно, приведе до його динамічного розвитку.

Сімейство HC11

На відміну від спеціалізованих мікроконтролерів «замовлених» сімейств, сімейство MC68HC11 містить близько 40 універсальних і високопродуктивних мікроконтролерів, орієнтованих як на масові ринки, так і на середнє і дрібне виробництво. *Процесорне ядро мікроконтролерів цього сімейства характеризується* підвищеною продуктивністю, відрізняється від HC05 ефективнішою архітектурою, системою команд, наявністю додаткових методів адресації і можливістю адресувати більший обсяг зовнішньої пам'яті. Мікроконтролери сімейства HC11 містять вбудовану пам'ять різних типів і конфігурацій.

Периферійні функції представлені багатофункціональними таймерами, АЦП (до 12 каналів і 10 розрядів), вбудованим співпроцесором, прискорюючим виконання множення і ділення на порядок, ШІМ і ЦАП; послідовними інтерфейсами, контролером ПДП, синтезатором тактової частоти та іншими функціями. Як і в інших сімействах, є велика різноманітність корпусів, а також версії зі зниженою напругою живлення і розширеним температурним діапазоном.

Сімейство HC05. Загальна характеристика і класифікація

Сімейство HC05 має в своєму складі близько 180 представників, що дає можливість використовувати найбільш широкий вибір варіацій вбудованої пам'яті і периферійних функцій. Це сімейство є основою програми фірми Motoro1a зі створення мікроконтролерів з характеристи-

ками, визначуваними користувачами, або таких «замовлених» мікроконтролерів. Ця програма направлена на створення мікроконтролерів, що поєднують низьку вартість при серійному виробництві з широкими функціональними можливостями і вимагають використання мінімального числа додаткових елементів.

Позначення мікроконтролерів містить символ, наступний в назві мікроконтролера безпосередньо за MC68HC05, що відносить його до однієї з підгруп у межах сімейства (або до серії), які відрізняються один від одного функціональними особливостями. З іншого боку, більшість мікроконтролерів сімейства HC05 створювалися під певні застосування, тому класифікацію зручно провести з урахуванням цих двох чинників одночасно.

Мікроконтролери загального призначення

Серія C характерна широкою різноманітністю вбудованої пам'яті і ліній паралельного введення-виведення. Асинхронний, послідовний інтерфейс (SCI) дозволяє організувати обмін даними із зовнішніми пристроями зі швидкістю до 131 кГц. Високошвидкісний синхронний послідовний інтерфейс (SPI) зручний для керування дисплеями і зовнішніми периферійними пристроями по 4-дротяній лінії. Усі мікроконтролери серії C мають у своєму складі 16-бітовий програмований таймер з функціями «схожої фіксації» і «вихідного порівняння» для одночасного вимірювання часових параметрів зовнішніх імпульсів і генерації імпульсного сигналу. Найбільш популярним (і, як наслідок, широкодоступним) представником серії C є мікроконтролер MC68HC705C8A з одноразово програмованою вбудованою пам'яттю, великою кількістю ліній введення-виведення, наявністю версії з подвоєною тактовою частотою і захистом коду від читання. Мікроконтролер MC68HC05CO є єдиним представником сімейства HC05, що не має вбудованого ПЗП і що адресує зовнішню пам'ять до 64 Кбайт.

Серія J включає недорогі 20-вивідні мікроконтролери, що містять ПЗП, багатофункціональний таймер і функцію переривання реального часу. Найбільш яскравою моделлю цієї серії є мікроконтролер MC68HC705J1A, наявність в якому програмованої пам'яті із захистом від

читання, виходів з високою здатністю навантаження, переривань від клавіатури, а також швидкісної версії і вельми недорогого комплекту налагоджувальних засобів роблять цей новий мікроконтролер все більш популярним.

Серія K містить найдешевші з тих, що випускаються фірмою Motoro1a 16-вивідні мікроконтролери, що включають пам'ять, таймер переривання реального часу, лінії з підвищеною здатністю навантаження і програмовану користувачем «ідентифікаційну» область пам'яті (8 байт) навіть у масовому варіанті.

Мікроконтролер MC68HC805K3 з вбудованою пам'яттю EEPROM (Flash) призначений для макетування і невеликих виробничих серій.

Серія P характерна наявністю вбудованого АЦП, різноманітністю варіантів вбудованої пам'яті (включаючи EEPROM), наявністю простого послідовного порту, а також невеликим 28-вивідним корпусом і низькою ціною. Найбільш популярним представником цієї серії є мікроконтролер MC68HC705P9.

Мікроконтролери для телекомунікацій

Серія F була спеціально створена для побудови абонентських телефонних апаратів та різних груп складності і терміналів. Головною особливістю мікроконтролерів цієї серії є *наявність в них генератора DTMF* для тонального набору номера і цифрової сигналізації. Ряд мікроконтролерів цієї серії додатково містять контролер ЖК-дисплея, а також великий обсяг ПЗП для реалізації складних алгоритмів сучасних ТА і незалежну пам'ять для зберігання номерів. Всі ці функції, разом з низьким споживанням, що дозволяє живитися від телефонної лінії або батарей, роблять мікроконтролери серії P привабливими для створення як масових телефонних апаратів, так і складного абонентського устаткування.

Серія L. Головними особливостями цієї серії є: наявність вбудованих контролерів алфавітно-цифрових і графічних ЖК-дисплеїв (від 32 до 40000 сегментів), наявність тонального генератора, годинника реального часу і низьке споживання енергії. Вони також широко використовуються при створенні різноманітного, особливо портативного, зв'язного устаткування: без-дротових телефонів, пристроїв персонального виклику (пейджерів), радіостанцій, цифрових блокнотів і т. ін.

Серія E включає мікроконтролери, що містять синтезатор тактової частоти з ФАПЧ для гнучкого керування споживанням, незалежну пам'ять, АЦП і інтерфейс I^2C, і використовується в засобах зв'язку як мікроконтролери загального призначення.

Серія C. Мікроконтролери загального призначення, також активно використовуються в комунікаційних пристроях, таких, як абонентські модулі АТС, системи цифрового ущільнення абонентських ліній і т. д.

Мікроконтролери для побутової електроніки

Серія L, що включає широкий вибір мікроконтролерів з контролерами ЖКІ, часто використовується в різноманітних побутових пристроях, що вимагають виведення на ЖКІ-дисплей.

Серія M містить вбудований контролер вакуум-флюоресцентного індикатора. Вбудований формувач забезпечує можливість керування по 24 лініях при напрузі 40 В. До складу мікросхем серії також входять 8-бітовий таймер, 6-канальний 8-бітовий АЦП, асинхронний, послідовний порт і великий обсягом вбудованого ПЗП.

Серія MC характеризується наявністю в 28-вивідному корпусі шви-дкісних каналів ШІМ і 6-канального АЦП та призначена для керування електродвигунами в «білій техніці»: холодильниках, пральних машинах, кухонних комбайнах і т. д.

Серія T спеціально призначена для застосування у відео- і телевізійній апаратурі і містить драйвер кольорового екранного дисплея, що дозволяє відображати на екрані ЕПТ символи і текстову інформацію. Вбудований АЦП може використовуватися для керування настройкою на певний канал телебачення. Канали ШІМ використовуються для керування гучністю звуку, яскравістю зображення і т. д. Інтерфейс I2C дозволяє керувати іншими підсистемами ТБ приймача, наприклад відеопроцесором.

Серія CC є продовженням серії T і має розширений драйвер екранного дисплея з можливістю секціонування даних.

Серії мікроконтролерів загального призначення, особливо найбільш дешеві серії з малою кількістю виведень (*K, J* і *RC*), використовують для вирішення простих задач керування в різноманітних побутових пристроях (наприклад, пульти дистанційного керування).

Мікроконтролери для автомобільної електроніки

Серія X включає мікроконтролери з вбудованим контролером локальної мережі (CAN-Controller Area Network) Європейського керуючого стандарту для побудови мультиплексної шини автомобіля. Мікроконтролери цієї серії використовують для локального керування збором даних у різних підсистемах автомобіля (приладова панель, склопідіймачі, підвіска, ABS, і т. ін.). Ряд моделей серії X містять розширений таймер, незалежну пам'ять, АЦП, ШІМ і розширений послідовний порт.

Серія V призначена для виконання функцій, аналогічних *серії X*, але орієнтована на інші стандарти побудови мультиплексної шини – MDLC (Message Data Link Control) або JL 850. Характерною особливістю мікроконтролерів цієї серії є поєднання декількох технологій: КМОП, високовольтної і силової, що дозволило створити «систему на кристалі», яка містить, крім стандартних блоків мікроконтролера, високовольтний регулятор напруги, трансівери мультиплексної шини, EEPROM, АЦП, ШІМ і інші пристрої.

Серії K, J, P загального призначення часто використовують для реалізації окремих функцій автомобіля, наприклад охоронної сигналізації (електронний ключ, центральний замок).

Мікроконтролери для промислового керування

Серія B поєднує великий обсяг вбудованого програмованого ПЗП, EEPROM, АЦП і ШІМ, а також таймер і розширений послідовний порт. Така конфігурація дозволяє використовувати мікросхеми серії в найрізноманітніших індустріальних застосуваннях.

Серія MC характеризується наявністю в 28-вивідному корпусі швидкісних каналів ШІМ і 6-канального АЦП і призначена для керування електродвигунами.

Серія X включає мікроконтролери з контролером локальної керуючої мережі, випускається в різних модифікаціях (ПЗП від 4 до 32К, корпус від 28 до 64 виводів, АЦП, ШІМ) і застосовується для побудови локальних вузлів збору/керування даних у розподілених системах керування технологічним устаткуванням. Мікроконтролери загального призна-

чення серій *K, J, P* також використовують при створенні розподілених систем керування/збору інформації, наприклад систем пожежної сигналізації, систем охорони доступу і т. ін. «Ідентифікаційна» програмована область пам'яті дозволяє будувати «адресні» датчики.

Мікроконтролери *серії L* з контролером ЖКІ також можуть використовуватися в таких системах, як вузли збирання/оброблення даних з індикацією. Низьке споживання мікроконтролерів цих серій дозволяє створювати системи з одночасною передачею живлення і даних по одній лінії.

ЦПУ і система команд сімейств HC05

ЦПУ сімейства HC05, що відрізняється простотою і зручністю програмування, має стандартну внутрішню тактову частоту 2 МГц, для деяких мікроконтролерів існують версії з тактовою частотою 4 МГц (цикл команди 250 нс). Програмна модель ЦПУ містить 5 регістрів (регістри ЦПУ не є частиною карти пам'яті).

Акумулятор (ACC) – 8-бітовий регістр загального призначення, в якому зберігаються операнди, результати арифметичних операцій, а також дані, з якими проводяться які-небудь операції. Акумулятор також використовується і для логічних операцій.

Індексний регістр (X) – використовується або при індексному режимі адресації, або як допоміжний акумулятор. Цей регістр може бути завантажений як безпосередньо, так і з пам'яті; може бути збережений в елементі пам'яті або порівняний з її вмістом.

Лічильник команд (PC) містить адресу команди, яка йде за виконуваною, або адресу операнда, що входить у код програми. Розрядність PC залежить від обсягу вбудованого ПЗП.

Указівник стека (SP) містить адресу наступної (вільної) комірки стека. Глибина стека мікроконтролерів сімейства HC05 становить 64 байт. Виклик підпрограми використовує 2 комірки стека, переривання – 5 комірок.

Регістр ознак (CC) містить 5 прапорів, що встановлюються залежно від результату виконання арифметичних і інших команд. Цими прапорами є: прапор напівперенесення (H), прапор негативного результату (S), прапор нульового результату (Z), біт маски переривань (I) і прапор перенесення (C).

Система команд мікроконтролерів включає 65 команд, що діляться на такі групи:

- команди переміщення даних (LDA, STA, CLR, LDX, STX...);
- команди передачі керування (JMP, JSR, RTI, переходи за умовами і бітами...);
- арифметичні команди (ADD, SUB, MUL...);
- логічні команди (AND, OR, COM, NEG...);
- команди роботи з бітами (BSET, BCLR, зміщення...);
- спеціальні команди (WAIT, STOP, SWI...).

Команди MC68HC05 виконуються, як правило, за 2...5 циклів внутрішньої тактової частоти, що становить 1...2,5 мкс при стандартній внутрішній тактовій частоті 2 МГц. Мікроконтролери сімейства MC68HC05 використовують вісім режимів адресації: неявна, безпосередня, пряма, розширена, індексна без зсуву, індексна з 8-розрядним зсувом, індексна з 16-розрядним зсувом, відносна.

Вбудована пам’ять мікроконтролерів сімейства HC05 може включати ПЗП (маскове, одноразово програмоване, програмоване з УФ-стиранням, програмоване з електричним стиранням) і ОЗП обсягом до 768 байтів. У карту пам’яті включені регістри портів паралельного введення-виведення (причому, як правило, ці адреси однакові для усіх моделей сімейства), а також адреси керуючих регістрів і регістрів даних периферійних пристроїв (таймера, послідовного інтерфейсу і т.д.). Область завантажувального ПЗП є масковою і дозволяє здійснювати «самозавантаження» даних у вбудовану пам’ять (включаючи програмування ППЗП) через паралельні або послідовні порти. Нарешті, вектори переривань, розташовані в ППЗП, визначають адреси переходів при перериваннях від периферійних підсистем (таймера, послідовних інтерфейсів, виведення зовнішнього переривання) і в інших випадках (RESET, програмне переривання).

Вбудовані підсистеми введення-виведення мікроконтролерів сімейства HC05 зустрічаються в найбільшій кількості мікроконтролерів даного сімейства. Основу блоку таймера становить 16-бітовий лічильник з попереднім дільником, що має можливість формувати переривання при переповненні і синхронізується внутрішньою тактовою частотою, яка ділиться

на 2. Більшість мікроконтролерів сімейства містить також пов'язані з лічильником підсистеми вхідної фіксації (ІС) і вихідного порівняння (ОС).

Система ІС служить для оброблення часових параметрів зовнішніх сигналів і дозволяє записувати в регістр ІС вміст лічильника при перепаді рівня зовнішнього сигналу з видачею відповідного переривання або установленням прапора.

Система ОС призначена для генерації імпульсного сигналу з програмованими часовими параметрами і дозволяє видавати в лінію порту «0» або «1» у момент рівності вмісту лічильника і вмісту регістра ОС.

У найбільш простих моделях сімейства функції ІС і ОС можуть бути відсутніми і замінюватися перериваннями реального часу (RTI) з програмованим інтервалом між перериваннями. Ще однією важливою системою, пов'язаною з таймером, є система стеження за виконанням програми (COP WatchDog). Ця система формує Reset, якщо періодично з певним проміжком часу не буде проведений запис у спеціальний регістр.

Послідовний інтерфейс зв'язку SCI є повнодуплексним асинхронним приймачем-передавачем, і може бути використаний для зв'язку з терміналом, PC (наприклад, за RS-232C) або іншими мікроконтролерами. Вбудований генератор частоти обміну дозволяє ділити внутрішню тактову частоту з отриманням 32 стандартних частот обміну – від 37,56 бод до 125 кбод. SCI так само підтримує такі функції, як програмована довжина посилання, виходу з режиму очікування приймача по вільній лінії або адресному маркеру, окремий дозвіл приймача і передавача, виявлення помилки кадру і шуму в лінії (з часовим дозволом 1/16 біт).

SCI може формувати 5 видів переривань (або встановлювати 5 прапорів) за такими умовами: «регістр даних передавача порожній», «передача завершена», «регістр даних приймача заповнений», «приймач переповнений і «лінія прийому вільна».

Послідовний периферійний інтерфейс SPI використовується для синхронної передачі інформації в послідовному коді на менші відстані, але зі значно більшою швидкістю і дозволяє МК взаємодіяти з різними периферійними пристроями, АЦП. SPI підтримує такі функції: повний дуплекс; режим провідного і веденого; 4 програмованих тактових частоти

до 1,05 МГц з програмованою полярністю і фазою; прапор переривання після закінчення передачі; захист від конфліктів на магістралі.

Аналого-цифровий перетворювач проводить перетворення зовнішньої напруги в діапазоні від Vss (нижня опорна напруга, що підключається до загальної шини) до Vrh (верхня опорна напруга) в 8-розрядний код від $00 до $FF відповідно.

АЦП використовує метод послідовних наближень, процес перетворення займає 32 цикли внутрішньої тактової частоти (16 мкс при 2 МГц). При тактовій частоті менше 1 МГц використовується вбудований RC-генератор частоти для АЦП (1,5 МГц). Вбудований мультиплексор дозволяє проводити перетворення по одному із зовнішніх аналогових входів (до 8), а також вимірювати Vss, Vrh, (Vrh+Vss)/2 для проведення контролю і юстирування. Підсистема АЦП містить регістр керування (задає режим роботи і запуск перетворення), регістр статусу (містить прапор закінчення перетворення) і регістр даних (результат перетворення). Характеристики МК представників сімейства HC05 наведені у табл. 2.13, в якій використовуються такі умовні позначення:

Ch – канал (channel);

EEPROM – електрично стираний програмований ПЗП;

i – вхід;

I/O – введення-виведення;

IC – вхідна фіксація (Input Capture), підсистема таймера для вимірювання тимчасових параметрів;

MFT – багатофункціональний таймер (Multifunction Timer);

o – вихід;

OC – вихідне порівняння (Output Capture), підсистема таймера для генерації імпульсу сигналу;

RTI – система переривань реального часу (Real Time Interrupt);

SCI – послідовний асинхронний інтерфейс зв’язку (Serial Communication Interface);

SCI+ – послідовний інтерфейс SCI, який може працювати також як SPI;

SIOP – послідовний порт введення-виведення (Simple serial input/ output port);

Таблиця 2.13 – Характеристики МК сімейства HC05

Назва	ПЗП	EEP ROM	Таймер	I/O	Послідовний інтерфейс	АЦП	ШІМ	Корпус	ОЗП	Примітка
68HC 705B5	6К	_	16 bit 2IC, 20C	94 8i	SCI+	8ch	2ch 8bit	56 - B	176	Загальне призначення АЦП + ШІМ +
68HC 705B16	15К	256	16 bit 2IC, 20C	32 20	SCI+	8ch •	2ch 8bit	52 - FN 64-FU	352	Вбуд. накачка захисту запису ЕСПЗУ, версія 4 МГц
68HC 705BD3	7,75К	—	MFT, PTI	24	1^2C	_	16ch 8bit	40 - P 42 - B	256	Процесор горизонтального і вертикального розгорнення
68HC70 5MC4	3,5К	_	16 bit 2IC, 20C MFT, RTI	22	SCI	6ch 8bit	2ch 8bit	28 - P 28 - DW 28 - S	176	Швидкісний ШІМ для керування двигуном
68HC 705P6A	4К	_	16 bit 1 IC, 1 0C	20	SIOP	4ch 8bit	–	28 - P 28 - DW	176	8KBI, 2 виходи по 15 мА

Короткий опис деяких представників сімейств HC05

МК загального призначення(MC68HC805K3), блок-схема якого подана на рис. 2.15, призначений для роботи в системах, що вимагають використання малогабаритного МК з мікроспоживанням і низькою вартістю: в охоронних системах, датчиках, побутових пристроях, портативних засобах зв'язку, пультах дистанційного керування і т. ін.

Коротка характеристика:

- ЦПУ HC05, множення 8 х 8;
- живлення від 1,8 В (запис EEPROM від 3В), низьке споживання;
- 920+16 байт EEPROM;
- переривання реального часу;
- переривання від клавіатури по 8 лініях (*);
- здатність навантаження 8 мА по чотирьох лініях (**);
- режими зниженого споживання STOP, WAIT; корпуси DIP-16, SOIC-16.

Рисунок 2.15 – Блок-схема мікроконтролера MC68HC805K3

Мікроконтролер загального призначення MC68HC705P6A, блок-схема якого подана на рис. 2.16, широко застосовується в різноманітних пристроях, що вимагають оброблення аналогового сигналу і обмежень до габаритів і вартості вживаного МК, наприклад, у пристроях контролю температури, в електронних вагах, локальних вузлах керування, пристроях сигналізації, побутовій техніці та ін.

Коротка характеристика:

- ЦПУ HCO5, множення 8 × 8, АЦП (4 канали, 8 розрядів);
- синхронний послідовний порт для підключення периферійних пристроїв, 16-розрядний таймер з функціями вхідного захоплення і вихідної фіксації, 4,6 кбайт програмованого ПЗП і 176 байтів ОЗП;
- 21 лінія введення-виведення, переривання від 8 ліній, 2 виходи зі струмом 15 мА; режими зниженого споживання STOP, WAIT, HALT, збереження даних в ОЗП, 28-вивідні корпуси DIP, SOIC.

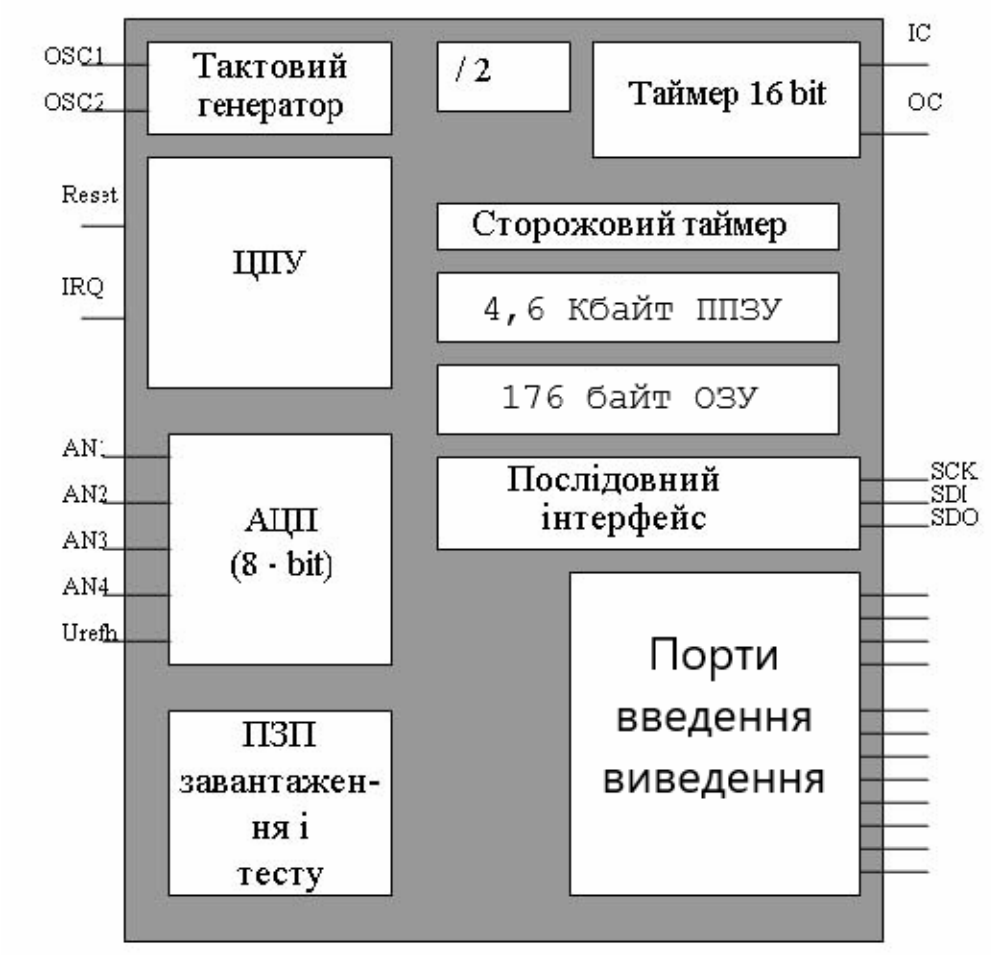


Рисунок 2.16 – Блок-схема мікроконтролера MC68HC705P6A

Мікроконтролер MC68HC705L1 наведений на рис. 2.17. Головними його особливостями є наявність вбудованого контролера ЖКІ на 64 сегменти і 6-канального 8-розрядного АЦП, а також робота від зниженої напруги живлення. Він широко застосовується в різних портативних вимірювальних приладах, засобах зв'язку, CD-плеєрах, радіоприймачах і т. ін.

Коротка характеристика:

- контролер ЖКІ до 64 сегментів (організація 3×12, 3×16, 4×12, 4×16);
- АЦП (6 каналів, 8 розрядів);
- 16-бітовий таймер з подвійними функціями IC і OC;
- лінія введення-виведення: 17 двонапр., 10 входів, 2 виходи;
- версії з живленням від 2,7В; корпус 56 –SDIP, 64 – QFP;

режими зниженого споживання STOP, WAIT.

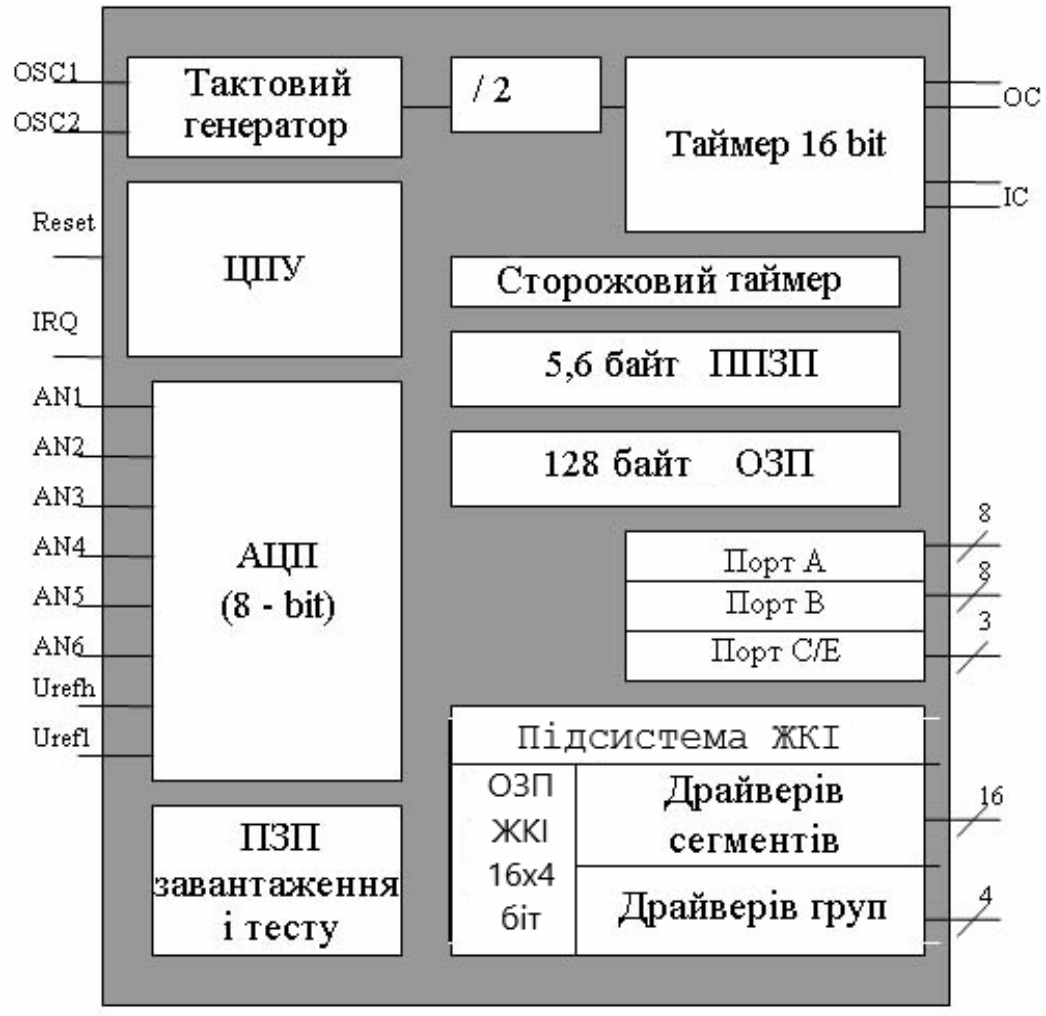


Рисунок 2.17 – Блок-схема мікроконтролера MC68HC705L1

Мікроконтролер MC68HC705MC4, блок-схема якого подана на рис. 2.18, оптимізований для керування електродвигунами з регульованою швидкістю обертання, які використовуються в побутовій техніці: компресорах холодильників, вентиляторах, пральних машинах, кухонних комбайнах та ін. Цей МК застосовується також для керування ключами в джерелах живлення, а також в інших пристроях.

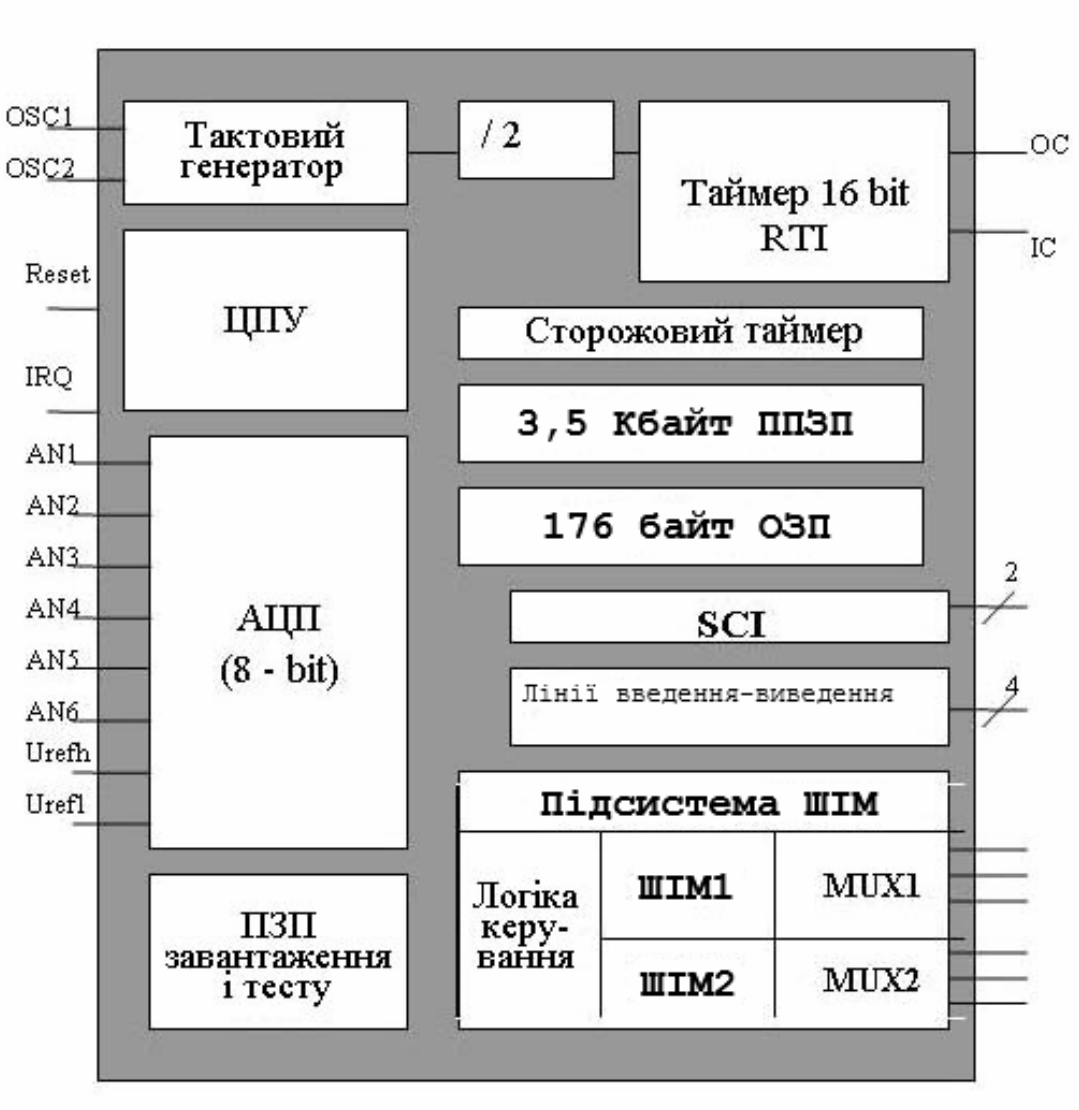


Рисунок 2.18 – Блок схема мікроконтролера MC68HC705MC4

Коротка характеристика:

- два 8-бітові канали ШІМ, незалежні або зв’язані;
- незалежні лічильники каналів;
- частота до 23 кГц;

- заповнення від 0 до 100 %, програмована полярність;
- програмоване перекриття;
- механізм одночасного оновлення регістрів ШІМ;
- механізм мультиплексування: 3 виходи з високою здатністю навантаження на кожен канал ШІМ;
- АЦП (6 каналів, 8 розрядів);
- 3,5 кбайт ППЗУ, 176 байт ОЗП, 16-бітовий таймер, RTI;
- асинхронний послідовний інтерфейс, 28-вивідні корпуси DIP, SOIC.

Сімейство 16-розрядних МК HC16 широко використовується в різних системах керування автомобілів, телекомунікаційному устаткуванні (сотових телефонах, телефонних комутаторах), побутовій електроніці (відеокамерах, телевізорах, цифрових аудіосистемах), офісній техніці (факсах, модемах, копіювальній техніці), медичному устаткуванні, робототехніці.

Ефективному використанню цих МК сприяє висока продуктивність 16-бітового CPU16 з частотою 25 МГц і потужна периферія.

Сімейство 32-розрядних МК 68300 є найбільш поширеним з високопродуктивних МК фірми Motorola. У ньому можна виділити *3 основних групи*, що принципово відрізняються за функціональним призначенням: *комунікаційну* (що містить комунікаційний співпроцесор), *промислового керування* (містить таймерний співпроцесор і застосовується в індустріальних системах керування, автомобільних контроллерах і т. д.), *загального призначення* (містить, крім центрального процесора, найбільш поширену універсальну периферію).

Докладніший опис архітектури (ЦПУ, системи команд, пам’яті і підсистем введення-виведення), зведених таблиць і структурних елементів та схем представників сімейств HC05, HC08, HC11, 16-розрядних HC12 і HC16, 32-розрядних 68300 мікроконтролерів фірми Motorola наведено у [24].

2.6. Мікроконтролери AVR Atmega 16

Загальні відомості

Серійне виробництво AVR почалося в 1996 р., а у теперішній час у серійному виробництві у Atmel знаходяться три сімейства AVR – «tiny», «classic» і «mega» [35].

Як і всі мікроконтролери AVR фірми «Atmel», микроконтролери сімейства Mega є 8-розрядними мікроконтролерами, призначеними для вбудовуваних застосувань. Вони виготовляються за малоспоживаючою КМОП-технологією, яка в поєднанні з вдосконаленою RISC-архітектурою дозволяє досягти якнайкращого співвідношення швидкодія/енергоспоживання. Мікроконтролери описуваного сімейства є найбільш розвиненими представниками мікроконтролерів AVR [35].

Відмітні особливості

До особливостей мікроконтролерів AVR сімейства Mega [35] належать:

- Flash ROM – обсяг незалежної пам’яті програм (кбайт);
- EEPROM – обсяг незалежної пам’яті даних (байт);
- RAM – обсяг статичної пам’яті даних (байт);
- External RAM – можливість підключення до мікроконтролера додаткової мікросхеми зовнішньої статичної пам’яті даних (кбайт);
- ISP – можливість програмування мікроконтролера в системі (на цільовій платі) при основній напрузі живлення;
- SPM – функція самопрограмування Flash ROM пам’яті мікроконтролера в системі без участі зовнішнього програматора;
- JTAG – вбудований JTAG – інтерфейс;
- I/O (pins) – максимальна кількість доступних ліній введення/виведення;
- Timer(s) 8/16 bit – кількість і розрядність таймерів/лічильників;
- USI – універсальний комунікаційний інтерфейс;
- AC – аналоговий компаратор;
- ADC (channels) – кількість каналів аналого-цифрового перетворення;
- Internal RC – наявність внутрішнього RC-кола для автономної роботи мікроконтролера (без зовнішнього джерела опорної частоти);

- WDT – сторожовий таймер;
- BDC – апаратний програмований блок захисту від перебоїв при раптовому (у тому числі і короткочасному) зникненні напруги живлення мікроконтролера;
- UART – асинхронний послідовний приймач-передавач;
- SPI – синхронний трипровідний послідовний інтерфейс;
- I2C – двопровідний послідовний інтерфейс;
- RTC – система реального часу;
- PWM (channels) – кількість незалежних каналів широтно-імпульсної модуляції;
- Command Set – кількість різних інструкцій в системі команд мікроконтролера;
- Vcc – діапазон робочої напруги живлення (В);
- Clock – діапазон робочих частот (МГц);
- Packages – типи корпусів, в які опресовується мікроконтролер, і за-гальна кількість виводів;
- FLASH-пам’ять програм обсягом 8.128 кбайт (число циклів стирання/запису не менше 1000);
- оперативна пам’ять (статичне ОЗП) обсягом 1,4 кбайт;
- пам’ять даних на основі ЭСППЗП (EEPROM) обсягом 512 байт – 4 кбайт (число циклів стирання/запису не менше 1000000);
- можливість захисту від читання і модифікації пам’яті програм і даних;
- можливість програмування безпосередньо в системі через послідовні інтерфейси SPI і JTAG;
- можливість самопрограмування;
- можливість внутрішньосхемного налагоджування відповідно до стандарту IEEE 1149.1 (JTAG);
- різні засоби синхронізації: вбудований RC-генератор з внутрішнім або зовнішнім RC-колом, що задає час, або із зовнішнім резонатором; зовнішній сигнал синхронізації;
- наявність декількох режимів зниженого енергоспоживання;
- наявність детектора зниження напруги живлення (brown-out detector, BOD);
- можливість програмного зниження частоти тактового генератора.

Характеристики процесора

Переважна більшість основних характеристик процесора мікроконтролерів сімейства Mega такі ж, як і у мікроконтролерів інших сімейств – Classic і Tiny:

- повна статична архітектура; мінімальна тактова частота, що дорівнює нулю;
- АЛУ підключено безпосередньо до регістрів загального призначення;
- більшість команд виконується за один машинний цикл;
- багаторівнева система переривань; підтримка черги переривань.

У той же час процесор мікроконтролерів сімейства Mega має ще ряд інших характеристик:

- найбільше число джерел переривань (до 27 джерел, з них до 8 зовнішніх);
- наявність програмного стека у всіх моделях сімейства;
- наявність апаратного помножувача.

Характеристики підсистеми введення/виведення

Усі характеристики підсистеми введення/виведення мікроконтролерів сімейства Mega такі ж, як і у мікроконтролерів інших сімейств:

- програмна конфігурація і вибір портів введення/виведення;
- виводи можуть бути запрограмовані як вхідні або як вихідні незалежно один від одного;
- вхідні буфери з тригером Шмітта на всіх виводах;
- можливість підключення до всіх входів внутрішніх підтягуючих резисторів.

Периферійні пристрої

Мікроконтролери сімейства Mega мають найбільш багатий набір периферійних пристроїв (ПП). Цими пристроями є:

- 8-розрядні таймери/лічильники (таймери T0 і T2). У ряді моделей ці таймери/лічильники можуть працювати як годинник реального часу (у асинхронному режимі);
- 16-розрядні таймери/лічильники (таймери T1 і T3);
- сторожовий таймер WDT;

- генератори сигналу з ШІМ розрядністю 8 біт (один з режимів роботи 8-розрядних таймерів/лічильників Т0 і Т2);
- одно-, дво- і триканальні генератори сигналу з ШІМ регульованою розрядністю (один з режимів роботи 16-розрядних таймерів Т1 і Т3). Дозвіл ШІМ-сигналу для різних моделей становить 8...10-бітний або 1...16-бітний;
- аналоговий компаратор;
- багатоканальний 10-розрядний АЦП як з несиметричними, так і з диференціальними входами;
- повнодуплексний універсальний асинхронний приймач-передавач (UART);
- повнодуплексний універсальний синхронний/асинхронний приймач-передавач (USART);
- послідовний синхронний інтерфейс SPI;
- послідовний двопровідний інтерфейс TWI.

Архітектура ядра

Ядро мікроконтролерів AVR сімейства Mega, як і ядро мікроконтролерів сімейства Classic і Tiny, виконане за вдосконаленою RISC-архітектурою. АЛП, що виконує усі обчислення, підключене безпосередньо до 32 робочих регістрів, об'єднаних в регістровий файл. Завдяки цьому АЛП виконує одну операцію (читання вмісту регістрів, виконання операції і запис результату назад в регістровий файл) за один машинний цикл. Практично кожна з команд (за винятком команд, у яких одним з операндів є 16-розрядна адреса) займає один елемент пам'яті програм.

У мікроконтролерах AVR реалізована гарвардська архітектура, яка характеризується роздільною пам'яттю програм і даних, кожна з яких має власні шини доступу до них. Така організація дозволяє одночасно працювати як з пам'яттю програм, так і з пам'яттю даних. Розділення шин доступу дозволяє використовувати для кожного типу пам'яті шини різної розрядності, причому способи адресації і доступу до кожного типу пам'яті також різні. Ще одним рішенням, направленим на підвищення швидкодії, є використання технології конвеєризації. Конвеєризація полягає в тому, що під час виконання поточної команди проводитися вибірка з пам'яті і дешифрування коду наступної команди. Основні параметри мікроконтролера ATmega 16 наведені в табл. 2.14.

Таблиця 2.14 – Основні параметри мікроконтролера ATmega 16

Позначення	Пам'ять програм (FLASH), Кбайт	Пам'ять даних (EEPROM), байт	Пам'ять даних (ОЗП), байт	Кількість ліній введення/ виведення	Напруга живлення, В	Тактова частота, МГц	Тип корпусу
ATmega 16	16	512	1К	32	4.5…5.5	0…16	DIP - 40 TQFP- 44

Архітектура мікроконтролерів сімейства Mega

Мікроконтролери AVR сімейства Mega є 8-розрядними мікроконтролерами з RISC-архітектурою [35]. Вони мають пам'ять програм (FLASH) і даних (EEPROM), що електрично стирається, а також різноманітні периферійні пристрої. Слід зазначити, що мікроконтролери сімейства Mega мають найбагатший набір периферійних пристроїв у порівнянні з мікроконтролерами інших сімейств. Структурна схема мікроконтролера наведена на рис. 2.19.

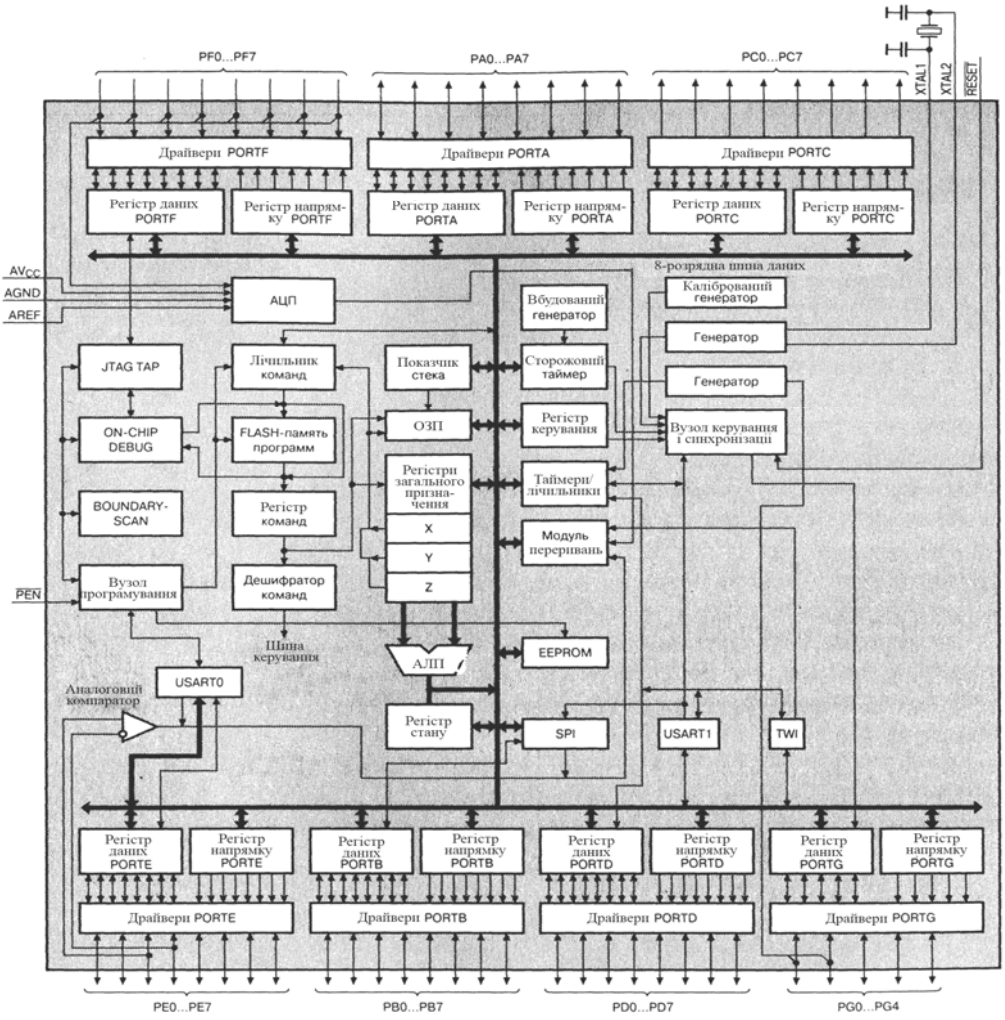


Рисунок 2.19. – Структурна схема мікроконтролерів сімейства Mega

Організація пам'яті

У мікроконтролерах AVR сімейства Mega реалізована гарвардська архітектура. Засоби адресації і доступу до цих областей пам'яті також різні. Узагальнена карта пам'яті мікроконтролера наведена в табл. 2.15.

Таблиця 2.15 – Загальна карта пам'яті

Модель	Пам'ять програм (Flash)		Пам'ять даних (ОЗП)		Пам'ять даних (EEPROM)	
	Верхня межа (F_END)	Обсяг, слів	Верхня межа (S_END)	Обсяг, Кбайт	Верхня межа (E_END)	Обсяг, байтів
ATmega 16x	$1FFF	8 K	$045F	1	$1FF	512

Така структура дозволяє центральному процесору працювати одночасно як з пам'яттю програм, так і з пам'яттю даних, що істотно збільшує продуктивність.

Кожна з областей пам'яті даних (ОЗП і EEPROM) також розташована в своєму адресному просторі.

Пам'ять програм

Пам'ять програм призначена для зберігання команд, що керують функціонуванням мікроконтролера.

Пам'ять програм також часто використовується для зберігання таблиць констант, не змінних під час роботи програми.

Пам'ять програм, яка електрично стирається ППЗП (FLASH-ПЗП). У зв'язку з тим, що довжина всіх команд кратна одному слову (16 біт) пам'ять програм має 16-розрядну організацію. Відповідно, ємність пам'яті мікроконтролерів сімейства Mega становить від 4К (4×I024) до 64К (64×I024) 16-розрядних слів.

Логічно пам'ять програм розділена на дві нерівні частини – область прикладної програми і область завантажувача. В останньому може розміщатися спеціальна програма (завантажувач), що дозволяє мікроконтролеру самостійно керувати завантаженням і вивантаженням прикладних програм. Якщо ж можливість самопрограмування мікроконтролера не використовується, прикладна програма може розташовуватися і в області завантажувача.

Для адресації пам'яті програм використовується лічильник команд (PC Program Counter).

Розмір лічильника команд становить 12...16 розрядів, залежно від обсягу пам’яті, що адресується.

За адресою $0000 пам’яті програм знаходиться вектор скидання.

Після скидання мікроконтролера виконання програми починається з цієї адреси (за цією адресою повинна розміщуватися команда переходу до частини ініціалізації програми). Починаючи з адреси $0002 пам’яті програм, розташовується таблиця векторів переривань.

При виникненні переривання після збереження в стеку поточного значення лічильника команд відбувається виконання команди, розташованої за адресою відповідного вектора. Тому за цими адресами розташовуються команди переходу до підпрограм оброблення переривань. У моделі ATmega16 використовуються команди абсолютного переходу (JMP).

Як відомо, пам’ять програм може використовуватися не тільки для зберігання коду програми, але також і для зберігання різних констант. Для пересилання байта з пам’яті програм у пам’ять даних є спеціальна команда LPM. При використовуванні команди LPM адреса, за якою здійснюється читання, визначається вмістом індексного регістра Z. При цьому старші 15 розрядів вмісту регістра визначатимуть адресу слова (0...32К), а молодший розряд визначатиме, який з байтів буде прочитаний: «0» – молодший байт, «1» – старший байт (рис. 2.20, *а*). При використанні цієї команди адреса слова визначатиметься розрядом RAMPZ0 регістра введення/виведення RAMPZ спільно зі старшими 15 розрядами вмісту регістра Z.

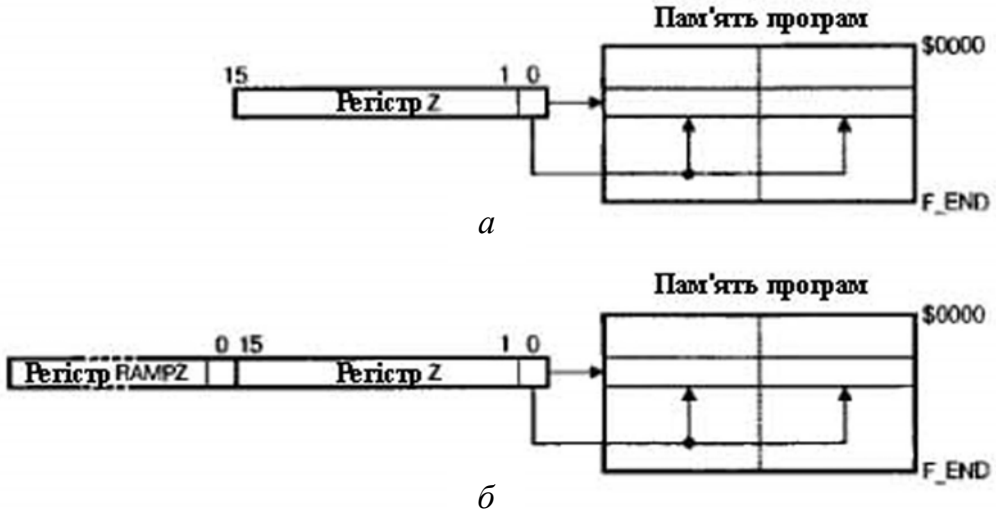


Рисунок 2.20 – Непряма адресація пам’яті програм при використанні команди LPM (*а*) і команди ELPM (*б*)

Молодший розряд регістра Z визначатиме, який з байтів слова буде прочитаний (рис. 2.20, *б*).

Регістр RAMPZ розташований за адресою $3B ($5B) в основному просторі РВВ мікроконтролерів ATmega16, а його формат показаний на рис. 2.21.

	7	6	5	4	3	2	1	0
$35	-	-	-	-	-	-	-	RAMPZ0
Читання (R)/ Запис (W)	R	R	R	R	R	R	R	R/W
Початкове значення	0	0	0	0	0	0	0	0

Рисунок 2.21 – Формат регістра RAMPZ

На закінчення слід зазначити, що FLAS-ПЗП, використовуваний в мікроконтролерах AVR, розрахований як мінімум на 1000 циклів стирання/запису.

Пам'ять даних

Пам'ять даних мікроконтролерів сімейства Mega розділена на три частини: регістрова пам'ять, оперативна пам'ять (статичне ОЗП) і энергонезалежне ЕСППЗП (EEPROM).

Регістрова пам'ять включає 32 регістри загального призначення (РЗП), об'єднані у файл, і службові регістри введення/виведення (РВВ). Під РВВ у пам'яті мікроконтролера відводиться 64 байт, а під ДРВВ – 160 байт. В обох областях регістрів введення/виведення розташовуються різні службові регістри (регістри керування мікроконтролером, регістр стану і т. ін.), а також регістри керування периферійними пристроями, що входять до складу мікроконтролера. Загальна кількість РВВ залежить від конкретної моделі мікроконтролера.

Для зберігання змінних програм, крім регістрів загального призначення, також може використовуватися статичне ОЗП обсягом від 512 байт до 4 Кбайт.

Для довготривалого зберігання різної інформації, яка може змінюватися в процесі функціонування готової системи (калібрувальні константи, серійні номери, ключі і т. ін.), в мікроконтролерах сімейства може використовуватися EEPROM-пам'ять. Її ємність становить для різних

моделей 512 байт...4 Кбайт. Ця пам'ять розташована в окремому адресному просторі, а доступ до неї здійснюється за допомогою визначених РВВ.

Статичне ОЗП

У мікроконтролерах AVR сімейства Mega використовується лінійна організація пам'яті. Обсяг статичного ОЗП для різних моделей сімейства становить 512 байт...4 Кбайт. У адресному просторі ОЗП також розташовані всі регістри мікроконтролерів, під них відведені молодші 96 (256) адрес. Решта адрес відведена під 512/1К/2К/4К...64К комірок статичного ОЗП.

Регістри загального призначення

Усі регістри загального призначення об'єднані в регістровий файл швидкого доступу, структура якого показана на рис. 2.22. У мікроконтролерах AVR всі 32 РЗП безпосередньо доступні АЛП, на відміну від мікроконтролерів інших фірм, в яких є тільки один такий регістр – робочий регістр W (акумулятор). Завдяки цьому будь-який РЗП може використовуватися практично у всіх командах і як операнд-джерело і як операнд-приймач. Таке рішення (у поєднанні з конвейєрним обробленням) дозволяє АЛП виконувати одну операцію за один машинний цикл.

Регістри	Адреса
R0	$00
R1	$01
R2	$02
R13	$0D
R14	$0E
R15	$0F
R16	$10
R17	$11
R26	$1A регістр X, мол. Байт
R27	$1B регістр X, ст. байт
R28	$1C регістр Y, мол. байт
R29	$1D регістр Y, ст. байт
R30	$1E регістр Z, мол. байт
R31	$1F регістр Z, ст. байт

Рисунок 2.22 – Структура регістрового файлу

Останні 6 регістрів файлу (R26...R31) можуть також об'єднуватися в три 16-розрядні регістри X, Y і Z, використовувані як указівники при непрямій адресації пам'яті даних.

Кожен регістр файлу має свою власну адресу в просторі пам'яті даних. Тому до них можна звертатися двома способами (як до регістрів і як до пам'яті), не дивлячись на те, що фізично ці регістри не є частиною ОЗП. Таке рішення є ще однією відмінною особливістю архітектури AVR, що підвищує ефективність роботи мікроконтролера і його продуктивність.

Регістри введення/виведення

Усі регістри введення/виведення умовно можна розділити на дві групи: службові регістри мікроконтролера і регістри, що належать до конкретних периферійних пристроїв (у тому числі регістри портів введення/виведення).

У всіх мікроконтролерах сімейства Mega регістри введення/виведення розташовуються в так званому просторі введення/виведення розміром 64 байти.

Розподіл адрес простору введення/виведення (як основного, так і додаткового) залежить від конкретної моделі мікроконтролера або, якщо точніше, від складу і можливостей периферійних пристроїв даної моделі.

До РВВ, розташованих в основному просторі введення/виведення, можна напряму звернутися за допомогою команд IN і OUT, що виконують пересилання даних між одним із 32 РОН і простором введення/виведення. У системі команд є також чотири команди порозрядного доступу, які використовують як операнди регістри введення/виведення: команди установлення/скидання окремого біта (SBJ і CBI) і команди перевірки стану окремого біта (SBIS і SBIC). На жаль, ці команди можуть звертатися тільки до однієї половини основних регістрів введення/виведення (адреси $00...$1F).

Крім безпосередньої адресації (за допомогою команд IN і OUT), до РВВ можна звертатися і як до комірок ОЗП за допомогою відповідних команд ST/SD/SDD і LD/LDS/LDD.

У першому випадку використовуються адреси РВВ, що належать основному простору введення/виведення ($00...$3F). У другому випадку адресу РВВ необхідно збільшити на $20. Серед усіх РВВ є один регістр,

що найчастіше використовується в процесі виконання програм. Цим регістром є регістр стану SREG. Він розташовується за адресою $3F ($5F) і містить набір прапорів, що показують поточний стан мікроконтролера. Більшість прапорів автоматично встановлюється в «1» або скидається в «0» при настанні певних подій (відповідно до результату виконання команд). Усі розряди цього регістра доступні як для читання, так і для запису; після скидання мікроконтролера усі розряди регістра скидаються в «0».

Формат цього регістра показаний на рис. 2.23, а його опис наведений в табл.2. 16.

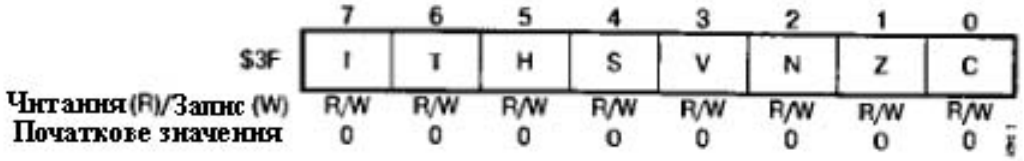


Рисунок 2.23 – Формат регістра стану SREG

Таблиця 2.16 – Розряди регістра стану SREG

Розряд	Назва	Опис
1	2	3
7	I	Загальний дозвіл переривань. Для дозволу переривань цей прапор повинен бути встановлений в «1». Дозвіл/заборона окремих переривань здійснюється установленням або скиданням відповідних розрядів регістрів масок (регістра керування перериваннями). Якщо прапор скинутий, то переривання заборонені незалежно від стану розрядів цих регістрів. Прапор скидається апаратно після входу в переривання і відновлюється командою RETI для дозволу.
6	T	Зберігання копійованого біта. Цей розряд регістра використовується як джерело або приймач команд копіювання бітів BLD і BST. Заданий розряд будь-якого РОН може бути скопійований в цей розряд командою BST або встановлений відповідно до вмісту даного розряду командою BLD
5	H	Прапор половинного перенесення. Цей прапор установлюється в «1», якщо відбулося перенесення з молодшої половини байта (з 3 розряду в 4) або займ зі старшої половини байта при виконанні деяких арифметичних операцій
4	S	Прапор знаку. Цей прапор дорівнює результату операції «ВИКЛЮЧАЮЧЕ АБО» (XOR) між прапорами N (негативний результат) і V (переповнювання числа в додатковому коді). Відповідно цей прапор установлюється в «1», якщо результат виконання арифметичної операції менше нуля

Закінчення табл. 2.16

1	2	3
3	V	Прапор переповнювання додаткового коду. Цей прапор установлюється в «1» при переповненні розрядної сітки знакового результату. Використовується при роботі зі знаковими числами, поданими в додатковому коді
2	N	Прапор негативного значення. Цей прапор установлюється в «1», якщо старший 7 розряд результату операції дорівнює «1». Інакше прапор дорівнює «0»
1	Z	Прапор нуля. Цей прапор установлюється в «1», якщо результат виконання операції дорівнює нулю
0	C	Прапор перенесення. Цей прапор установлюється в «1», якщо в результаті виконання операції відбувся вихід за межі байта

Використання зовнішнього ОЗП

Для дозволу роботи із зовнішнім ОЗП необхідно встановити в «1» розряд SRE регістра MCUCR.

Якщо робота із зовнішнім ОЗП дозволена, то при зверненні за адресою, що знаходиться поза межею внутрішнього ОЗП, автоматично відбувається звернення до зовнішнього ОЗП. Після формування на виведеннях порту A молодшого байта адреса виведення ALE змінює свій стан з логічної «1» на логічний «0» і залишається в цьому стані протягом усього циклу читання/запису. Звернення до внутрішнього ОЗП при дозволеній роботі із зовнішнім ОЗП також може привести до деякої активності на виведеннях портів A і C, проте це не впливає на роботу схеми, оскільки сигнали стробів читання (RD) і запису (WR) при цьому знаходяться в неактивному стані.

За відсутності звернення до зовнішньої пам’яті виведення порту A переводяться мікроконтролером у третій стан. Цього можна уникнути, якщо підключити до виходів порту внутрішні підтягуючі резистори або встановити в «1» розряд XMBK регістра SFIOR або XMCRB. При встановленому розряді на виведеннях порту A завжди зберігається останнє виведене значення.

Підключення зовнішнього ОЗП до мікроконтролера показано на рис. 2.24.

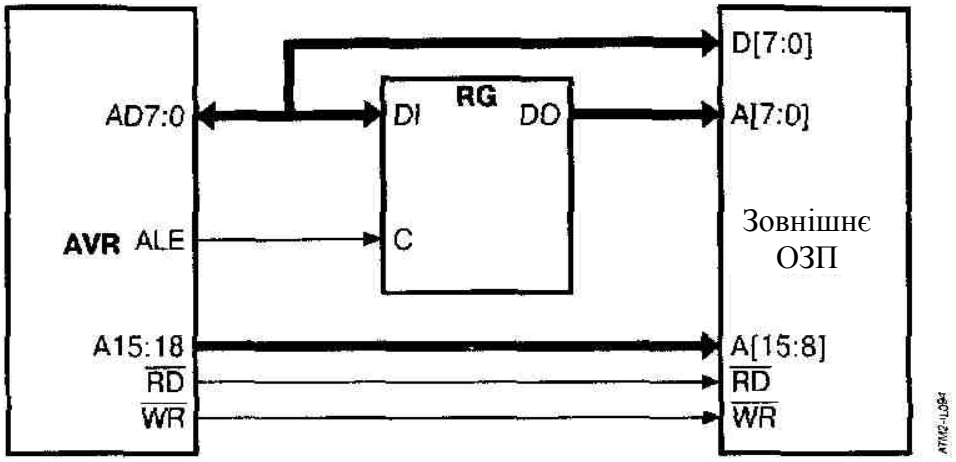


Рисунок 2.24 – Підключення зовнішнього ОЗП до мікроконтролера

Дані мікроконтролери мають такі можливості при роботі із зовнішньою пам’яттю:

- керування тривалістю циклу звертання до зовнішньої пам’яті;
- розділення зовнішньої пам’яті на два сектори з можливістю задання різної тривалості циклу звертання для кожного сектора;
- керування розрядністю шини адреси;
- утримання значень на шині даних для зменшення струмоспоживання.

Мікроконтролери сімейства Mega дозволяють використовувати мікросхеми із зовнішнім ОЗП з різним часом доступу. Підстроювання під різні мікросхеми здійснюється зміною тривалості циклу звертання до зовнішньої пам’яті. Більш того, є можливість розбити весь адресний простір зовнішнього ОЗП на два сектори, для кожного з яких може бути задана своя тривалість циклу звертання.

У загальному вигляді конфігурація зовнішнього ОЗП, підключеного до мікроконтролера, показана на рис. 2.25.

При роботі із зовнішнім ОЗП використовується 16-розрядна шина адреси, яка дозволяє адресувати 64К адрес. Для багатьох застосувань не вимагається такої великої ємності зовнішньої пам’яті. Число виведень порту С, задіяних під шину адреси, визначається вмістом розрядів XMM2…XMMO. Незадіяні виведення можуть використовуватися як лінії введення/виведення загального призначення.

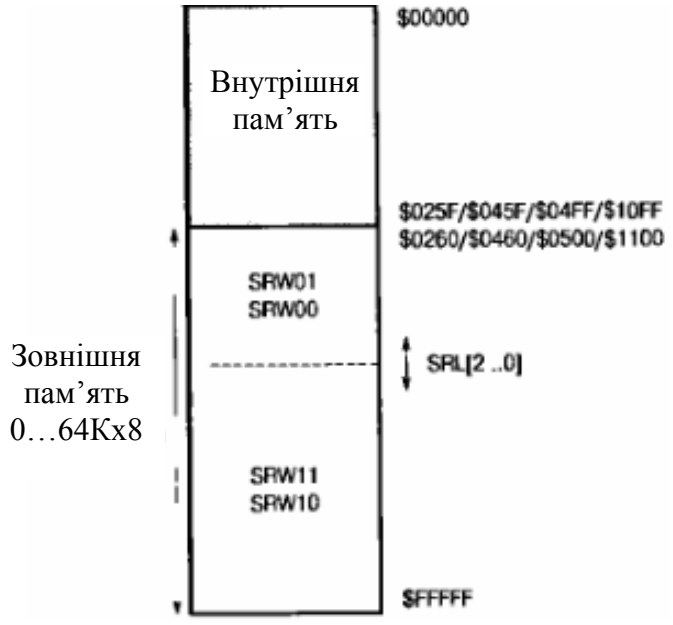


Рисунок 2.25 – Конфігурація зовнішнього ОЗП

За допомогою розрядів XMM2...XMMO можна також забезпечити повне використання місткості мікросхем зовнішнього ОЗП. За умовчанням молодші адреси зовнішньої пам'яті відображаються на адресний простір внутрішнього ОЗП і частина місткості зовнішнього ОЗП залишається незадіяним. Повністю використовувати місткість зовнішнього ОЗП можна, програмно керуючи старшими розрядами адреси. Для виконання цього «трюка» порт С повинен бути налаштований на вихід, а в защіпці порту повинно бути записано «$00». При маскуванні старших розрядів адреси на відповідні виведення порту С будуть виставлені «00» і відбудеться звертання до молодших адрес зовнішнього ОЗП починаючи з адреси $0000.

Необхідно пам'ятати, що в порівнянні зі звертанням до внутрішнього ОЗП звертання до зовнішнього ОЗП вимагає на 1, 2, 3 або 4 машинні цикли більше для кожного байта, оброблюваного командою. Таким чином, час виконання команд передачі даних (LD, SI, LDS, STS, PUSH і POP) збільшується на 1 (2, 3,4) цикли. Якщо стек розташований в зовнішньому ОЗП, то час переходу до оброблення переривань, виклику і повернення з підпрограм збільшується на 3 (5, 7, 9) машинних цикли. Це пов'язано з тим, що під час виконання вказаних операцій відбувається збереження або відновлення 16-розрядного лічильника команд і, крім того, при звертанні до зовнішньої пам'яті не використовується конвеєризація.

Способи адресації пам'яті даних

Мікроконтролери AVR сімейства Mega підтримують 8 засобів адресації доступу до різних областей пам'яті даних (РОН, РВВ, ОЗП).

Способів адресації всього два: пряма адресація і непряма. Проте кожен спосіб адресації має декілька різновидів залежно від того, до якої області пам'яті робиться звертання (при прямій адресації) або які додаткові дії виконуються над індексним регістром (при непрямій адресації).

Пряма адресація

При прямій адресації адреси операндів містяться безпосередньо в слові команди. Відповідно до структури пам'яті даних існують такі різновиди прямої адресації: пряма адресація одного РОН, пряма адресація двох РОН, пряма адресація РВВ, пряма адресація ОЗП.

Пряма адресація одного регістра загального призначення

Цей спосіб адресації використовується в командах, що оперують з одним з регістрів загального призначення. При цьому адреса регістра операнда (його номер) міститься в п'яти розрядах слова команд (рис. 2.26).

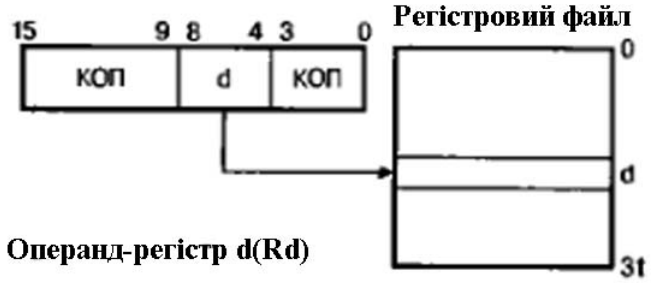


Рисунок 2.26 – Пряма адресація одного регістра загального призначення

Прикладом команд, що використовують цей спосіб адресації, є команди роботи зі стеком (PUSH, POP), команди інкремента (INC), декремента (DEC), а також деякі команди арифметичних операцій.

Пряма адресація двох регістрів загального призначення

Цей спосіб адресації використовується в командах, що оперують одночасно з двома регістрами загального призначення. При цьому адреса регістра джерела міститься в розрядах 9, 3...0 (5 розрядів), а адреса регістра приймача в разрядах 8...4 (5 розрядів) слова команди (рис. 2.27).

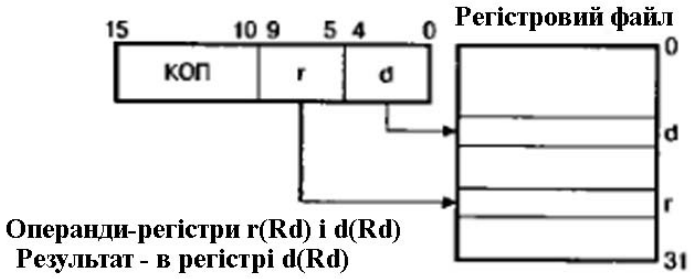


Рисунок 2.27 – Пряма адресація двох регістрів загального призначення

До команд, що використовують цей спосіб адресації, належать команди пересилання даних з регістра в регістр (MOV), а також більшість команд арифметичних операцій.

Пряма адресація регістра введення/виведення

Даний спосіб адресації використовується командами пересилання даних між регістром введення/виведення, розташованим в основному просторі введення/виведення, і регістровим файлом IN і OUT. В цьому випадку адреса регістра введення/виведення міститься в розрядах 10, 9, 3...0 (6 розрядів), а адреса РЗП – в розрядах 8...4 (5 розрядів) слова команди (рис.2.28). Положення розрядів r/d і P на рис. 2.28 показане умовно.

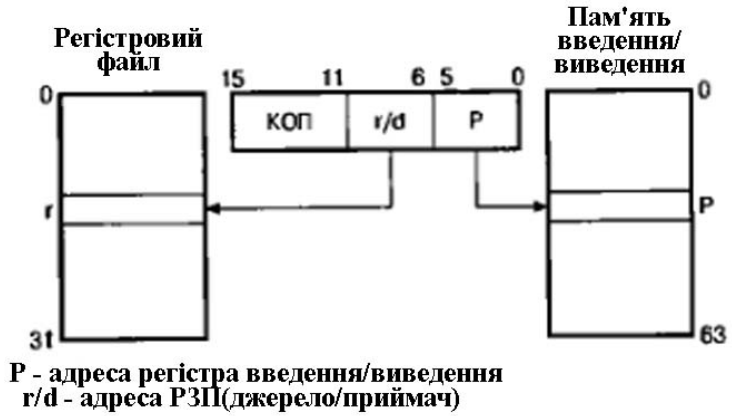


Рисунок 2.28 – Пряма адресація регістра введення/виведення

Пряма адресація ОЗП

Даний спосіб використовується для звертання до всього адресного простору пам’яті даних.

У системі команд мікроконтролерів сімейства є тільки дві команди, які використовують цей спосіб адресації. Це команди пересилання байта

між одним з РЗП і осередовищем ОЗП LDS і STS. Кожна з цих команд займає в пам'яті програм два слова (32 розряди).

У першому слові міститься код операції і адреса регістра загального призначення (у розрядах з 8 по 4).

У другому слові знаходиться адреса елемента пам'яті, до якої відбувається звертання (рис. 2.29).

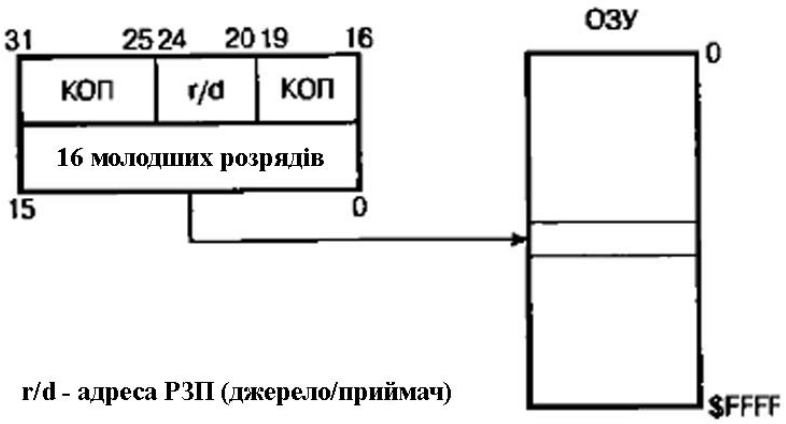


Рисунок 2.29 – Пряма адресація ОЗП

Непряма адресація

При непрямій адресації адреса елемента пам'яті знаходиться в одному з індексних регістрів X, Y і Z. Залежно від додаткових маніпуляцій, які здійснюються над вмістом індексного регістра, розрізняють такі різновиди непрямої адресації: проста непряма адресація, відносна непряма адресація, непряма адресація з переддекрементом і непряма адресація з постінкрементом.

Енергонезалежна пам'ять даних (EEPROM)

Усі мікроконтролери сімейства Mega мають у своєму складі енергонезалежну пам'ять (EEPROM-пам'ять). Обсяг цієї пам'яті коливається від 512 байт у моделях ATmega8x до 4 кбайт у старших моделях. EEPROM-пам'ять, розташована у своєму адресному просторі і так само, як і ОЗП, організована лінійно. Для роботи з EEPROM-памяттю використовуються три регістри введення/виведення: регістр адреси, регістр даних і регістр керування.

Регістр адреси

Регістр адреси EEPROM-пам'яті EEAR (EEPROM Address Register) фізично розміщується в двох РВВ EEARH:EEARL, розташованих за ад-

ресами $1F ($3F) і $1E ($3E) відповідно. У цей регістр завантажується адреса комірки, до якої вироблятиметься звертання. Регістр адреси доступний як для запису, так і для читання. При цьому в регістрі EEARH задіюються тільки молодші розряди. Незадіяні розряди регістра EEARH доступні тільки для читання і містять «0».

Регістр даних

Регістр даних EEPROM-пам'яті EEDR (EEPROM Data Register) розташований за адресою $ID ($3D). При записі в цей регістр завантажується дані, які повинні бути поміщені в EEPROM, а при читанні в цей регістр поміщаються дані, зчитані з EPROM.

Регістр керування

Регістр керування EEPROM-пам'яті EECR (EEPROM Control Register) розташований за адресою $IC ($3C). Цей регістр використовується для керування доступом до EEPROM-пам'яті. Формат цього регістра показаний на рис. 2.30., а його опис – в табл. 2.17.

Процедура запису одного байта в EEPROM-пам'ять складається з таких етапів:

1. Дочекатися готовності EEPROM до запису даних (чекати, поки скинеться прапор EEWE регістра EECR).

2. Дочекатися завершення запису в FLASH-пам'ять програм (чекати, поки не скинеться прапор SPMEN регістра SPMR).

3. Завантажити байт даних у регістр EEDR, а необхідну адресу – в регістр EEAR (за необхідності).

4. Установити в «1» прапори EEMWE регістра EECR.

5. Записати в розряд EEWE регістра EECR логічну «1» протягом 4 машинних циклів. Після встановлення цього розряду процесор пропускає 2 машинні цикли перед виконанням наступної інструкції.

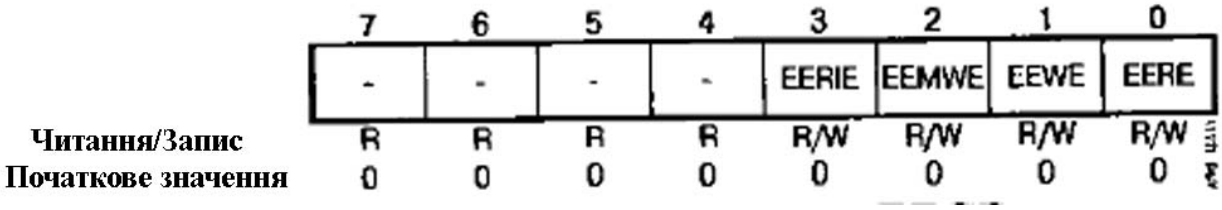


Рисунок 2.30 – Формат регістра EECR

Таблиця 2.17 – Розряди регістра EECR

Розряд	Назва	Опис
7…4	–	Не використовуються, читаються як «0»
3	EERIE	Дозвіл переривання від EEPROM. Цей розряд керує генерацією переривання, що виникає при завершенні циклу запису в EEPROM. Якщо цей розряд встановлений в «1», переривання дозволені (якщо прапор 1 регістра SREG також встановлений в «1»). При скинутому розряді EEWE переривання генерується постійно
2	EEMWE	Керування дозволом запису в EEPROM. Стан цього розряду визначає функціонування прапора дозволу запису EEWE. Якщо даний розряд встановлений в «1», то при записі в розряд EEWE «1» відбувається запис даних в EEPROM. Інакше встановлення EEWE в «1» не справляє ніякого враження. Після програмного встановлення розряд EEMWE скидається апаратно через 4 машинні цикли
1	EEWE	Дозвіл запису в EEPROM. При встановленні цього розряду в «1» відбувається запис даних в EEPROM (якщо EEMWE рівний «1»)
0	EERE	Дозвіл читання з EEPROM. Після встановлення цього розряду в «1» виконується читання даних з EEPROM. Після закінчення читання цей розряд скидається апаратно

Процес звернення до EEPROM-пам’яті контролюється внутрішнім RC-генератором. Відповідно тривалість циклу запису залежить від частоти цього генератора, напруги живлення, температури і становить 7.5...9.0 мс для ATmega16. Після закінчення циклу запису розряд EEWE апарата скидається, після чого програма може почати запис наступного байта.

Лічильник команд

Лічильник команд є регістром, в якому міститься адреса наступної виконуваної команди. Напряму з програми він недоступний. Розмір лічильника команд залежить від обсягу наявної пам’яті програм і становить 12 (ATmega8x)...16 (ATmegaI28x) розрядів.

При нормальному виконанні програми вміст лічильника команд автоматично збільшується на 1 або на 2 (залежно від виконуваної команди) в кожному машинному циклі. Цей порядок порушується при виконанні команд переходу, виклику і повернення з підпрограм, а також при виникненні переривань.

Після включення живлення, а також після скидання мікроконтролера в лічильник програм автоматично завантажується стартова адреса

$0000 або початкова адреса сектора завантажувача. Як правило, за цією адресою розташовується команда безумовного переходу до частини ініціалізації програми.

При виникненні переривання в лічильник команд завантажується адреса відповідного вектора переривання. Якщо переривання використовуються в програмі, за адресами векторів переривань повинні розміщуватися команди переходу до підпрограм оброблення переривань.

Команди умовного переходу

У цих командах здійснюється перевірка умови, результат якої впливає на стан лічильника команд. Якщо умова істинна, відбувається перехід за заданою адресою. Якщо ж умова помилкова, виконується наступна команда.

Команди умовного переходу мають обмеження щодо області дії. Насправді нове значення лічильника команд виходить добавленням до нього або відніманням з нього деякого зсуву. А оскільки під значення зсуву в слові команди відводиться всього 7 розрядів, максимальна величина переходу становить 63... 64 слів. Оскільки перехід за заданою адресою здійснюється завантаженням нового значення в лічильник команд, то у разі істинності при перевірці умови в конвеєрі виникає затримка тривалістю в один машинний цикл.

Команди безумовного переходу

Мікроконтролери сімейства Mega мають 3 команди безумовного переходу: команду відносного переходу RGMP, команду абсолютного переходу JMP, а також команду непрямого переходу IJMP.

Відносний перехід – команда RJMP

При виконанні команди відносного переходу вміст лічильника команд змінюється додаванням до нього або відніманням від нього деякого значення, що є операндом команди, як показано на рис. 2.31. Положення розрядів *k* на рисунку показано умовно. Оскільки під значення операнда в слові команди відводиться всього 11 розрядів, за допомогою цієї команди можна переходити тільки в межах 2047...+2048 слів.

У програмах у ролі операнда цієї команди замість констант використовуються мітки. Асемблер сам обчислює величину переходу і підставляє це значення в слово команди. Оскільки команда відносного переходу змінює вміст лічильника команд, вона виконується за 2 машинні цикли.

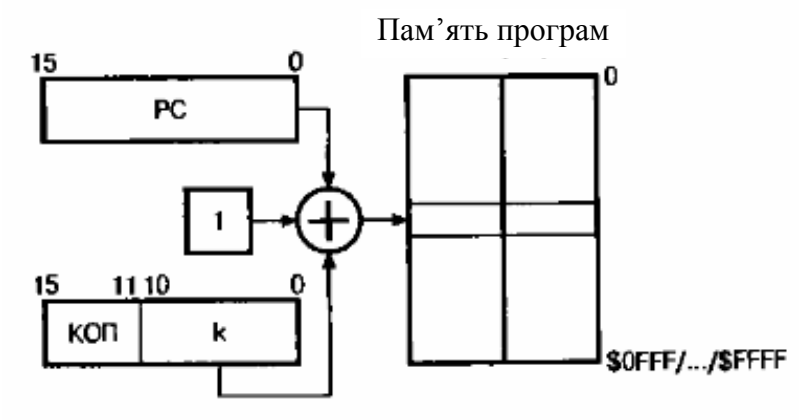


Рисунок 2.31 – Відносна адресація пам’яті програм

Абсолютний перехід – команда JMP

При виконанні команд абсолютного переходу в лічильник команд просто завантажується значення нової адреси, що є операндом команди (рис. 2.32). Положення розрядів *k* на рис. 2.32 показано умовно. За допомогою цієї команди можна здійснювати перехід у межах усього адресного простору мікроконтролерів AVR.

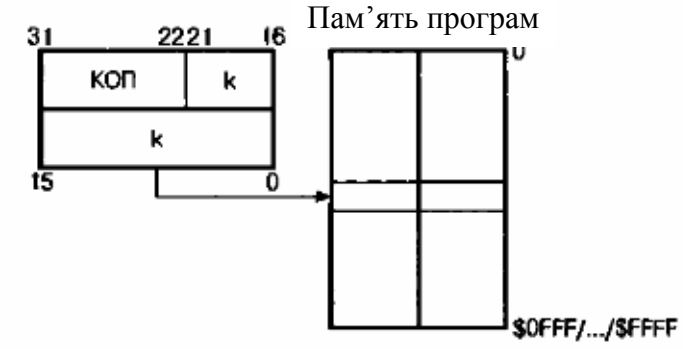


Рисунок 2.32 – Абсолютна адресація пам’яті програм

Команда абсолютного переходу виконується за 3 машинних цикли. Додатковий цикл пов’язаний з більшою довжиною коду команди (2 слова).

Непрямий перехід – команда IJMP

При виконанні цієї команди здійснюється перехід за адресою, яка знаходиться в індексному регістрі Z. Відповідно процес виконання команди зводиться до завантаження вмісту індексного регістру в лічильник команд (рис. 2.33).

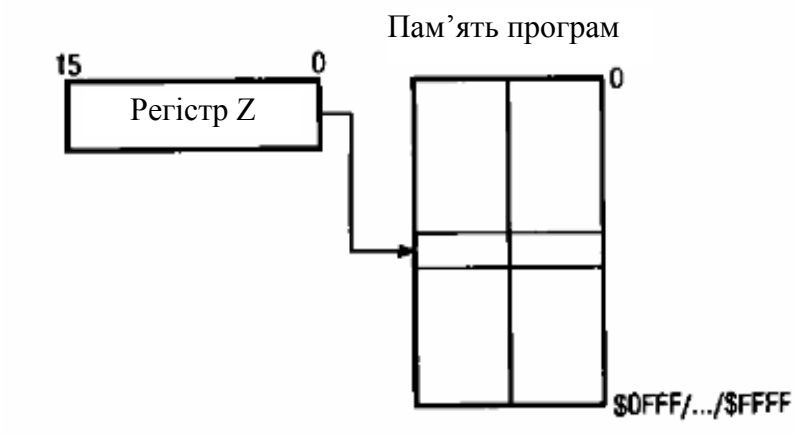


Рисунок 2.33 – Непряма адресація пам’яті програм

Як і команда абсолютного переходу, дана команда не має обмежень щодо області дії. Дійсно, оскільки індексний регістр розрядний, максимально можлива величина переходу становить 64 кслів (128 Кбайт)

Команди виклику підпрограм

Для виклику підпрограм в мікроконтролерах сімейства Mega є три команди: команда відносного виклику RCALL, команда абсолютного виклику CALL, а також команда непрямого виклику ICALL.

Відносний виклик підпрограми – команда RCALL

Спочатку команда RCALL зберігає в стеку значення лічильника команд. Потім вміст лічильника команд збільшується або зменшується на деяке значення, що є операндом команди (див. рис. 2.31). Оскільки операнд є 12-розрядним числом, максимальна величина переходу становить 2047... +2048 слів.

У програмах у ролі операндів цієї команди, як і у разі команди RJMP, використовуються мітки. Асемблер сам обчислює величину переходу і під-ставляє це значення в слово команди.

Абсолютний виклик підпрограми – команда CALL

При виконанні команди після збереження в стеку поточного значення лічильника команд в останній завантажується число, що є операндом команди (див. рис. 2.32). За допомогою цієї команди можна здійснювати виклик у межах усього адресного простору. Команда абсолютного виклику підпрограм виконується за 4 машинних цикли.

Непрямий виклик підпрограми – команда ICALL

Команда ICALL спочатку зберігає в стеку значення лічильника команд. Потім у лічильник команд завантажується вміст індексного регістра. Максимально можлива величина переходу становить 64К слів (128 Кбайт), тому данна команда також не має обмежень щодо області дії. Як і команда RCALL, команда непрямого виклику підпрограм виконується за 3 машинних цикли.

Команди повернення з підпрограм

У кінці кожної підпрограми обов'язково повинна знаходитися команда повернення з неї. У системі команд мікроконтролерів є дві такі команди. Для повернення зі звичної підпрограми, що викликається командою RCALL, використовується команда RET. Для повернення з підпрограми оброблення переривання використовується команда RETI.

Обидві команди відновлюють зі стека вміст лічильника команд, збережений там перед переходом до підпрограми. Команда повернення з підпрограми RETI додатково встановлює в «1» прапор загального дозволу переривань I регістра SREG, що скидається апаратно при виникненні переривання.

На виконання кожної з команд повернення з підпрограми потрібно 4 машинних цикли.

Стек

У всіх мікроконтролерах сімейства Mega стек реалізований програмно. Він розміщується в пам'яті даних, і його глибина визначається тільки розміром вільної області пам'яті програм. Як указівник стека у всіх моделях використовується пара регістрів введення/виведення SPH:SPL, розташованих за адресами $3E ($5E) і $3D ($5D) відповідно. Оскільки після подачі напруги живлення (або після скидання) в регістрах міститься нульове значення, на самому початку програми указівник стека необхідно проаналізувати, записавши в нього значення верхньої адреси пам'яті даних.

При виклику підпрограм адреса команди, розташованої за командою виклику, зберігається в стеку. Значення указівника стека при цьому зменшується на 2, оскільки для зберігання лічильника команд потрібно 2 байти. При поверненні з підпрограми ця адреса витягується із стека і завантажується в лічильник команд. Значення указівника стека відповідно

збільшується на 2. Те ж відбувається і під час переривання. При генерації переривання адреса наступної команди зберігається в стеку, а при поверненні з підпрограми оброблення переривання він відновлюється зі стека.

Для роботи зі стеком у наборі команд є дві команди: команда занесення в стек (PUSH) і команда витягання зі стеку (POP).

Тактування, режими зниженого енергоспоживання і скидання

Загальні відомості

З мікроконтролерами сімейства Mega можуть використовуватися різні джерела тактового сигналу. Першим використовується вбудований кварцовий генератор із зовнішнім резонатором, що підключається. Також як тактовий може використовуватися простий RC-генератор як внутрішній (що калібрується), так і із зовнішнім RC-колом. Крім того, як тактовий може використовуватися зовнішній сигнал синхронізації.

Усі мікроконтролери сімейства Mega мають декілька (до 6) режимів зниженого енергоспоживання – так звані «сплячі» режими. Кожний з цих режимів дозволяє скоротити енергоспоживання мікроконтролера в періоди його бездіяльності. Вхід в будь-який з цих режимів виконується за командою SLEEP. При виході мікроконтролера зі «сплячого» режиму здійснюється його скидання.

Скидання мікроконтролера відбувається не тільки при його «пробудженні», але і при настанні ряду інших подій. Цими подіями є: поява на виході RESET сигналу НИЗЬКОГО рівня, включення напруги живлення, зниження напруги живлення нижче мінімально допустимого рівня, спрацьовування сторожового таймера, а також отримання команди скидання за інтерфейсом JTAG.

Тактовий генератор

Спрощений пристрій синхронізації мікроконтролерів сімейства Mega подано на рис. 2.34.

Як видно з рис. 2.34, на основі системного тактового сигналу формуються додаткові сигнали, які використовуються для тактування різних модулів і блоків мікроконтролера:

- $clka_{cup}$ – тактовый сигнал центрального процесора, використовується для тактування блоків мікроконтролера, які відповідають за роботу з ядром мікроконтролера (регістровий файл, пам’ять даних). При вимкненні цього сигналу ЦПК залишається, усі розрахунки завершуються;

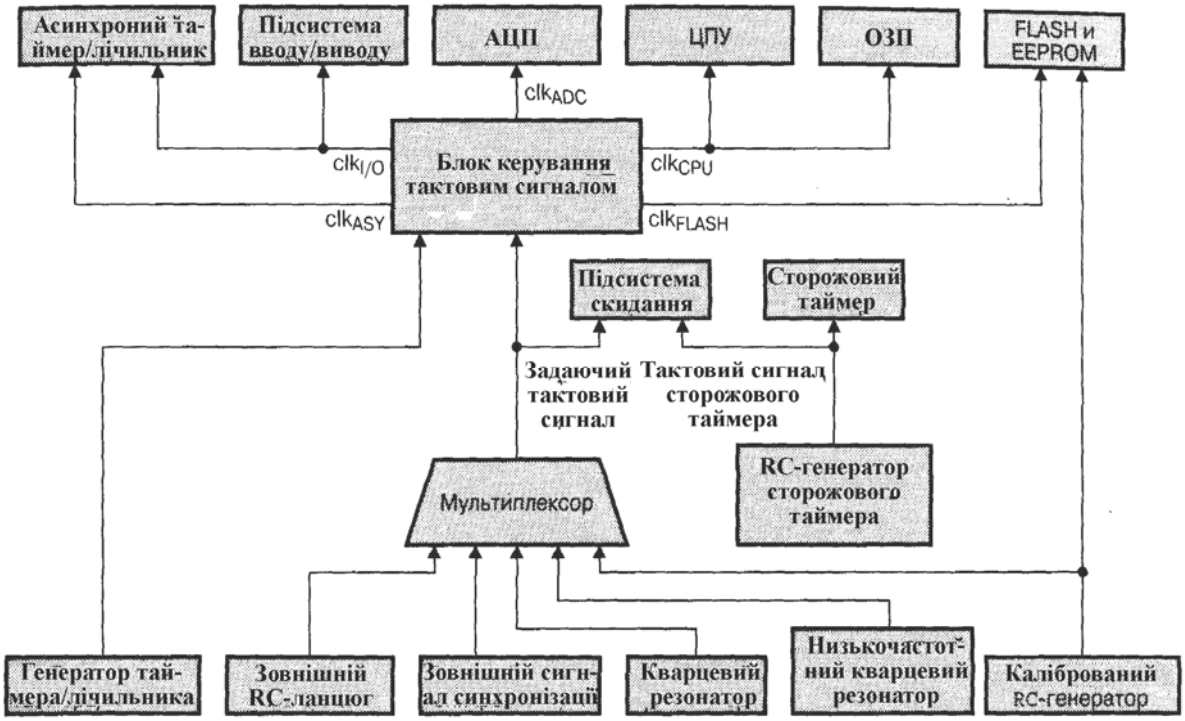


Рисунок 2.34 – Пристрій синхронізації

- $clka_{I/0}$ – тактовий сигнал підсистеми введення/виведення, використовується більшістю периферійних пристроїв, таких, як таймери/лічильники і интерфейсні модулі. Цей сигнал використовується також підсистемою зовнішніх переривань, але ряд зовнішніх переривань можуть генеруватися і за його відсутності;
- $clka_{FlASH}$ – тактовий сигнал для керування FLASH-пам’яттю програм. Як правило, цей сигнал формується одночасно з тактовим сигналом центрального процесора;
- clk_{ASY} – тактовий сигнал асинхронного таймера/лічильника. Тактування здійснюється безпосередньо від зовнішнього кварцового резонатора. Наявність цього сигналу дозволяє використовувати відповідний таймер/лічильник як годинник реального часу навіть при знаходженні мікроконтролера в «сплячому» режимі;
- clk_{ADC} – тактовий сигнал модуля АЦП. Наявність цього тактового сигналу дозволяє здійснювати перетворення при зупиненому ЦПК і підсистемі введення / виведення. При цьому значно зменшується рівень перешкод, що генеруються мікроконтролером, точність перетворення збільшується.

Тактовий генератор МК сімейства Mega може працювати із зовнішнім кварцовим резонатором, зовнішнім або внутрішнім RC-колом, також із зовнішнім сигналом синхронізації. Можливість використання того або іншого джерела тактового сигналу залежить від моделі мікроконтролера (табл. 2.18).

Таблиця 2.18 – Джерела тактового сигналу

Джерело тактового сигналу	ATmega16x
Генератор із зовнішнім резонатором	+
Генератор із зовнішнім RC-колом	+
Генератор з внутрішнім RC-колом	+
Зовнішній сигнал синхронізації	+

Режими зниженого енергоспоживання

Різні моделі мікроконтролерів сімейства підтримують 3...6 режимів зниженого енергоспоживання (табл. 2.19). Режими відрізняються числом периферійних пристроїв мікроконтролера, функціонуючих у «сплячому» режимі, і ступенем зменшення енергоспоживання.

Таблиця 2.19 – Режими зниженого енергоспоживання

Режим зниженого енергоспоживання	ATmega16x
Idle	+
ADC Noise Reduction	+
Power Down	+
Power Save	+
Standby	+
Extended Standby	+

У цілому для керування «сплячим» режимом у мікроконтролерах сімейства використовується три або чотири (залежно від моделі) розряди регістрів введення/виведення.

Перемикання в будь-який з режимів зниженого споживання здійснюється командою SLEEP. При цьому прапор SE повинен бути встановлений в «1». Щоб уникнути ненавмисного перемикання мікроконтролера в «сплячий» режим, рекомендується встановлювати цей прапор безпосе-

редньо перед виконанням команди SLEEP. Режим, в який перейде мікроконтролер після виконання команди SLEEP, визначається станом розрядів SM2...SMO.

Вихід зі «сплячого» режиму може бути здійснений в результаті переривання і скидання.

При генерації переривання мікроконтролер переходить в робочий режим, зупиняється на 4 машинних циклах, виконує підпрограму оброблення переривання і відновлює виконання програми з інструкції, наступної за командою SLEEP. Вміст РОН, ОЗП і РВВ при цьому не змінюється.

Після переходу мікроконтролера в робочий режим керування передається за адресою вектора скидання.

Скидання

«Скидання» переводить мікроконтролер в певний стійкий стан. Скидання може бути викликане такими подіями:

- включення напруги живлення мікроконтролера;
- подача сигналу низького рівня на виведення RESET (апаратне скидання);
- тайм-аут сторожового таймера;
- падіння напруги живлення нижче заданої величини;
- скидання інтерфейсом JTAG.

При настанні будь-якої з перерахованих подій у всі регістри введення/виведення заносяться їх початкові значення, а в лічильник команд завантажується значення адреси вектора скидання. За цією адресою повинна знаходитися команда безумовного переходу на початок програми. Якщо ж переривання в програмі не використовуються, то вона може починатися безпосередньо з адреси вектора скидання.

Структурна схема підсистеми скидання наведена на рис. 2.35.

Логіка схеми скидання всіх мікроконтролерів сімейства Mega така. При настанні події, що приводить до скидання мікроконтролера, формується внутрішній сигнал скидання. Одночасно запускається таймер формування затримки скидання. Після закінчення певного проміжку часу внутрішній сигнал скидання знімається і починається виконання програми.

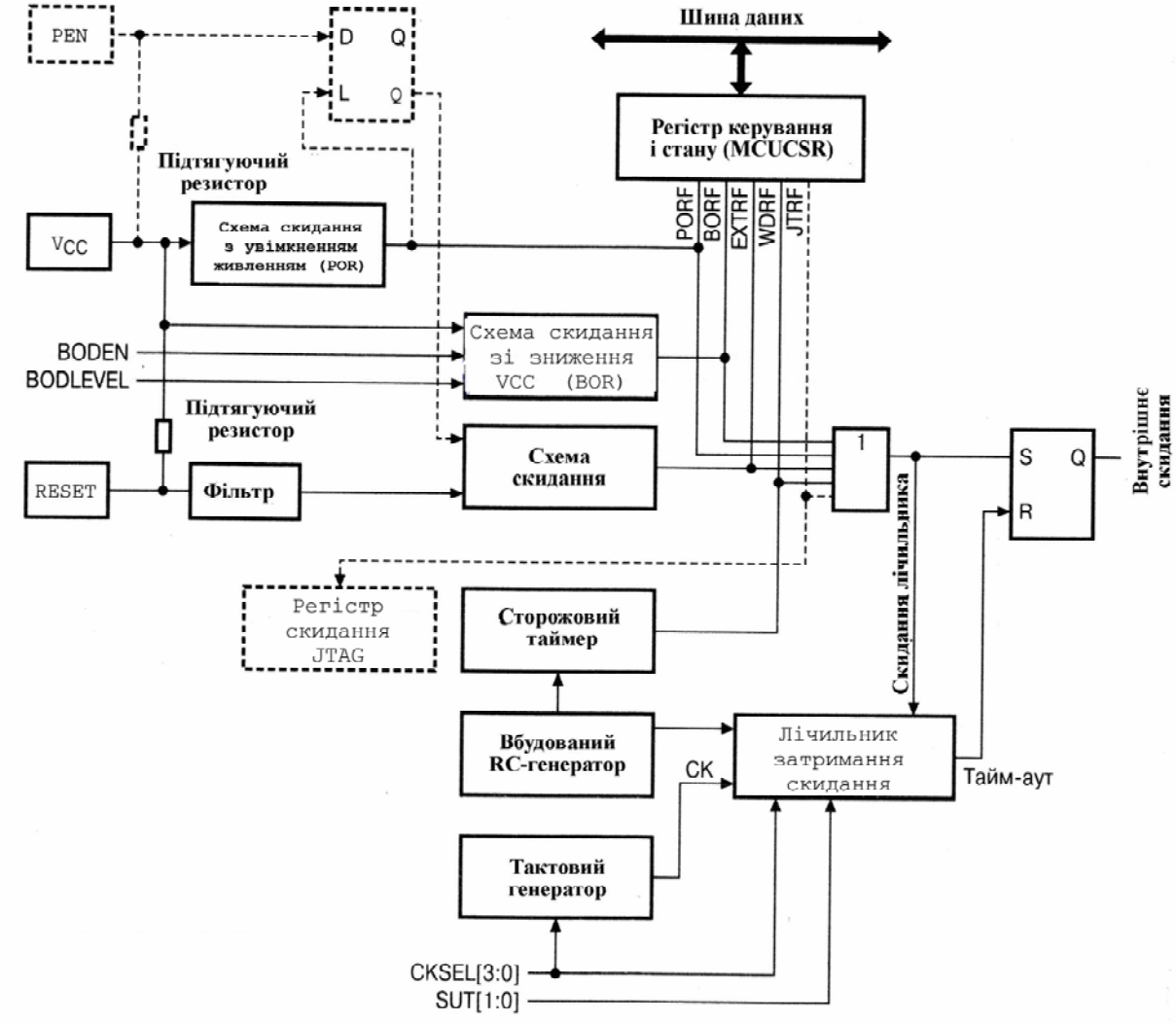


Рисунок 2.35 – Структурна схема підсистеми скидання мікроконтролерів сімейства Mega

Переривання

Загальні відомості

Переривання припиняє нормальний хід програми для виконання пріоритетної задачі, що визначається внутрішньою або зовнішньою подією мікроконтролера. При виникненні переривання мікроконтролер зберігає в стеку вміст лічильника команд PC і завантажує в нього адресу відповідного вектора переривання. За цією адресою, як правило, знаходиться команда безумовного переходу до підпрограми оброблення переривання. Останньою командою підпрограми оброблення переривання повинна бути команда RETI, яка забезпечує повернення в основну програму і відновлення заздалегідь збереженого лічильника команд.

Оскільки джерелами переривань, зокрема, є різні периферійні пристрої мікроконтролерів, кількість переривань залежить від конкретної моделі.

Таблиця векторів переривань

Як і всі мікроконтролери AVR, мікроконтролери сімейства Mega мають багаторівневу систему пріоритетних переривань. Молодші адреси пам'яті програм, починаючи з адреси $0002, відведені під таблицю векторів переривання. Кожному перериванню відповідає адреса в цій таблиці, яка завантажується в лічильник команд при виникненні переривання. Положення вектора в таблиці також визначає і пріоритет відповідного переривання: чим менше адреса, тим вище пріоритет переривання. Розмір вектора переривання залежить від ємності пам'яті програм мікроконтролера і становить 2 байт для моделей ATmega16X. Відповідно для переходу до підпрограм оброблення переривань використовують команди JMP.

Практично в усіх мікроконтролерах сімейства Mega положення векторів переривань може бути змінене. Таблиця може розташовуватися не тільки на початку пам'яті програм, а також і на початку області завантажувача. Причому переміщення таблиці може бути здійснене безпосередньо в ході виконання програми.

Керування розміщенням таблиці переривань здійснюється двома молодшими розрядами регістрів GICR: IVSEL (перший розряд) і IVCE (нульовий розряд). Стан прапора IVSEL визначає положення таблиці в пам'яті програм. Якщо прапор скинутий в «0», таблиця векторів переривань розташовується на початку пам'яті програм, а якщо прапор встановлений в «1», – на початку області завантажувача. Конкретне значення початкової адреси області завантажувача залежить від установлення конфігураційних середовищ BOOTSZ1 і BOOTSZ0. Розряд IVCE призначений для зміни прапора IVSEL.

Для зміни положення таблиці векторів переривань необхідно встановити розряд IVCE в «1» і потім протягом наступних чотирьох машинних циклів занести необхідне значення в розряд IVSEL. При цьому розряд IVCE скидається в «0». Інакше розряд IVCE буде скинутий після закінчення чотирьох машинних циклів, забороняючи подальшу зміну прапора IVSEL.

На час виконання описаної послідовності переривання автоматично забороняються і дозволяються тільки після скидання прапора IVCE. Стан прапора I регістра SREG при цьому не міняється.

Обробка переривань

Для глобального дозволу/заборони переривань призначений прапор I регістра SREG. Для дозволу переривань він повинен бути встановлений в «1», а для заборони – скинутий в «0». Індивідуальний дозвіл або заборона (маскування) переривань виконується встановленням/скиданням відповідних розрядів регістрів масок переривань.

При виникненні переривання прапор I регістра SREG апаратно скидається, забороняючи тим самим обробку наступних переривань. Проте в підпрограмі оброблення переривання цей прапор можна знову встановити в «1» для дозволу вкладених переривань. При поверненні з підпрограми оброблення переривання (при виконанні команди RETI) прапор I встановлюється апаратно.

Усі наявні переривання можна розділити на два типи. Переривання першого типу генеруються при настанні деякої події, в результаті якої встановлюється прапор переривання. Потім, якщо переривання дозволене, в лічильник команд завантажується адреса вектора відповідного переривання. При цьому прапор переривання апаратно скидається. Він також може бути скинутий програмно шляхом запису лог.1 в розряд регістра відповідного прапора.

Переривання другого типу не мають прапорів переривань і генеруються протягом усього часу, поки є умови, необхідні для генерації переривання. Відповідно, якщо умови, що викликають переривання, зникнуть до дозволу переривання, генерації переривання не відбудеться.

Слід пям’ятати, що при виклику підпрограм оброблення переривань регістр стану SREG не зберігається. Тому користувач повинен самостійно запам’ятовувати вміст цього регістра при вході в підпрограму оброблення переривання (якщо це необхідно) і відновлювати його значення перед викликом команди RETI.

Мікроконтролери сімейства Mega підтримують чергу переривань, яка працює таким чином: якщо умови генерації одного або більше переривань виникають у той час, коли прапор загального дозволу переривань скинутий (усі переривання заборонені), відповідні прапори встановлюються в «1» і залишаються в цьому стані до встановлення прапора загаль-

ного дозволу переривань. Після дозволу переривань виконується їх оброблення у порядку пріоритету.

Найменший час відгуку для будь-якого переривання становить 4 машинних цикли, протягом яких відбувається збереження лічильника команд в стеку. Якщо переривання відбудеться під час виконання команди, що триває декілька циклів, то генерація переривання відбудеться тільки після виконання цієї команди. Якщо ж переривання відбудеться під час знаходження мікроконтролера в «сплячому» режимі, час відгуку збільшується ще на 4 машинних цикли.

Повернення в основну програму займає 4 машинних цикли, протягом яких відбувається відновлення лічильника команд зі стека. Після виходу з переривання процесор завжди виконує одну команду основної програми, перш ніж обслужити будь-яке відкладене переривання.

Порти введення/виведення

Загальні відомості

Кожен порт мікроконтролерів складається з певної кількості виведень, через які мікроконтролер може здійснювати прийом і передачу цифрових сигналів. Завдання напряму передачі даних через будь-який контакт введення/виведення може бути виконане програмно у будь-який момент часу.

Вихідні буфери усіх портів, маючи симетричні характеристики навантажень, забезпечують високу здатність навантаження при будь-якому рівні сигналу. Здатності навантаження достатньо для безпосереднього керування світлодіодними індикаторами.

Вхідні буфери всіх виведень побудовані за схемою тригера Шмітта. Для усіх входів є можливість підключення внутрішнього підтягаючого резистора між входом і шиною живлення V_{cc}.

Відмінною особливістю портів мікроконтролерів сімейства Mega при використанні їх як цифрових портів введення/виведення загального призначення є реалізація істинної функціональності операції «читання/модифікація/запис». Завдяки цьому можна виконувати операції над будь-якими виведеннями (за допомогою команд SBI і CBI), не впливаючи на інші виведення порту. Це стосується зміни режиму роботи контакту введення/виведення, зміни стану вихідного буфера (для виходів) і зміни стану внутрішнього підтягуючого резистора (для входів).

ATmegal6x мають чотири 8-розрядні порти введення/виведення (порти A, B, C, D), 32 контакти введення/виведення.

Регістри портів введення/виведення

Звертання до портів виконується через регістри введення/виведення. Під кожен порт в адресному просторі введення/виведення зарезервовано по 3 адреси, за якими розміщені такі регістри: регістр даних порту PORTx, регістр напряму даних DDRx і регістр виведень порту PINx. Дійсні назви регістрів утворюються підстановкою назви порту замість символу «x», відповідно регістри порту A називаються PORTA, DDRA, PINA, порту B – PORTB, DDRB, PINB і т. д.

Оскільки за допомогою регістрів PINx здійснюється доступ до фізичних значень сигналів на виведеннях порту, вони доступні тільки для читання, тоді як решта двох регістрів доступна і для читання, і для запису.

Таймери

Мікроконтролери сімейства залежно від моделі мають у своєму складі від двох до чотирьох таймерів/лічильників загального призначення (табл. 2.20).

Таймер/лічильник T0 має мінімальний набір функцій, який залежить від моделі мікроконтролера. Він може використовуватися для відліку і вимірювання тимчасових інтервалів як лічильник зовнішніх подій, можливість генерації сигналів з широтно-імпульсною модуляцією (ШІМ) фіксованої розрядності.

Таймер/лічильник T1 використовують для відліку тимчасових інтервалів і як лічильник зовнішніх подій. Крім того, він може виконувати запам’ятовування свого стану за зовнішнім сигналом. Як і таймер/лічильник T0, він може працювати як широтно-імпульсний модулятор, але вже змінної розрядності і до того ж багатоканально.

Таблиця 2.20 – Таймери/лічильники загального призначення

Таймер/лічильник	ATmega16x
Таймер/лічильник T0 (8-розрядний)	+
Таймер/лічильник T1 (16-розрядний)	+
Таймер/лічильник T2 (8-розрядний)	+
Таймер/лічильник T3 (16-розрядний)	–

Таймер/лічильник Т2 повністю аналогічний таймеру/лічильнику Т0. За наявності в мікроконтролері обох таймерів/лічильників один з них може працювати в асинхронному режимі, а інший – як лічильник зовнішніх подій.

Таймер/лічильник Т3 за функціональними можливостями ідентичний таймеру/лічильнику Т1.

У складі всіх мікроконтролерів сімейства є також сторожовий таймер. Цей таймер дозволяє уникнути несанкціонованого зациклення програми, що виникає з тих або інших причин.

Сторожовий таймер

Сторожовий таймер призначений для захисту мікроконтролера від перебоїв у процесі роботи. Структурна схема сторожового таймера наведена на рис. 2.36.

Сторожовий таймер має незалежний тактовий генератор, тому він працює навіть під час знаходження мікроконтролера в будь-якому з режимів спокою. Типове значення частоти цього генератора дорівнює 1 МГц при V_{cc} = 5.0 В.

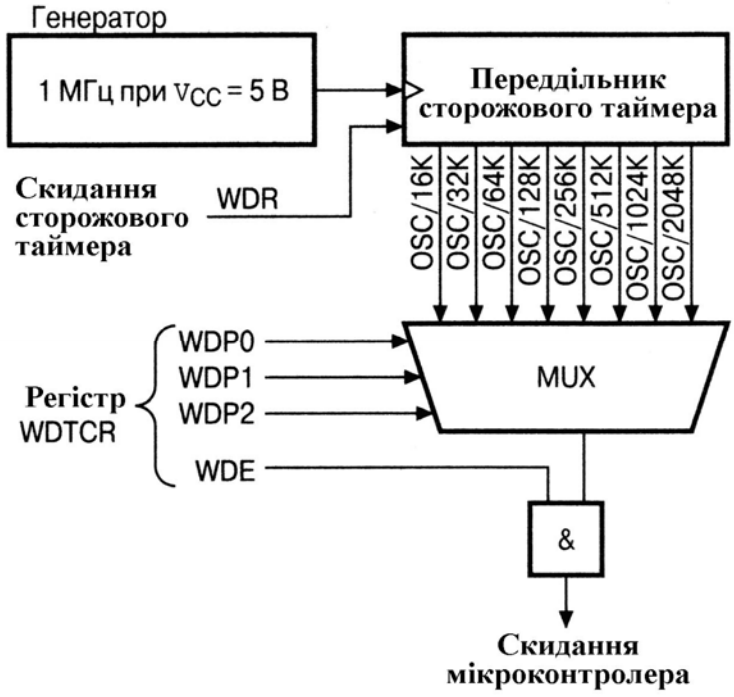


Рисунок 2.36 – Структурна схема сторожового таймера

Фактична частота генератора залежить від напруги живлення пристрою, температури, технологічних чинників. Якщо сторожовий таймер включений, то через проміжки часу, рівні його періоду, він виконує скидання МК. Щоб уникнути скидання при нормальному виконанні програми, сторожовий таймер необхідно регулярно скидати через проміжки часу, менші його періоду. *Скидання сторожового таймера* здійснюється командою WDR. *Для керування сторожовим таймером* призначений регістр WDTCR. *Для включення/вимкнення сторожового таймера* використовуються два розряди регістра WDTCR: WDE і WDTOE. Якщо розряд WDE встановлений в «1», сторожовий таймер включений, якщо скинутий в «0» – вимкнений. Безпосередньо перед включенням таймера рекомендується виконувати його скидання командою WDR. У Atmegal6x для вимкнення сторожового таймера необхідно однією командою записати лог. 1 в розряди WDE і WDTOE, а протягом наступних чотирьох машинних циклів записати лог. 0 в розряд WDE. Зміна періоду тайм-ауту може бути виконана у будь-який момент часу без жодних обмежень.

Аналоговий компаратор

Модуль аналогового компаратора входить до складу всіх без винятку сімейства Mega. Будучи включеним, компаратор дозволяє порівнювати значення напруг, присутніх на двох виводах мікроконтролера. Результатом по-рівняння є логічне значення, яке може бути прочитане з програми. За результатом порівняння можна згенерувати переривання, а також здійснити захоплення стану таймера/лічильника Т1. Остання функція дозволяє, зокрема, вимірювати тривалості аналогових сигналів. Використовувані компаратором виводи є контактами портів введення/виведення загального призначення (табл. 2.21).

Щоб указані виводи могли використовуватися аналоговим компаратором, вони повинні бути сконфігуровувані як входи (відповідний розряд регістра DDRx встановлений в «1»).

Таблиця 2.21 – Виводи, використовувані аналоговим компаратором

Назва	ATmega 16x	Призначення
AIN0	PB2	Неінвертуючий вхід
AIN1	PB3	Інвертуючий вхід

Крім того, необхідно відключити внутрішні підтягаючі резистори записом лог. 0 у відповідний розряд регістра PORTx. Структурна схема аналогового компаратора наведена на рис. 2.37.

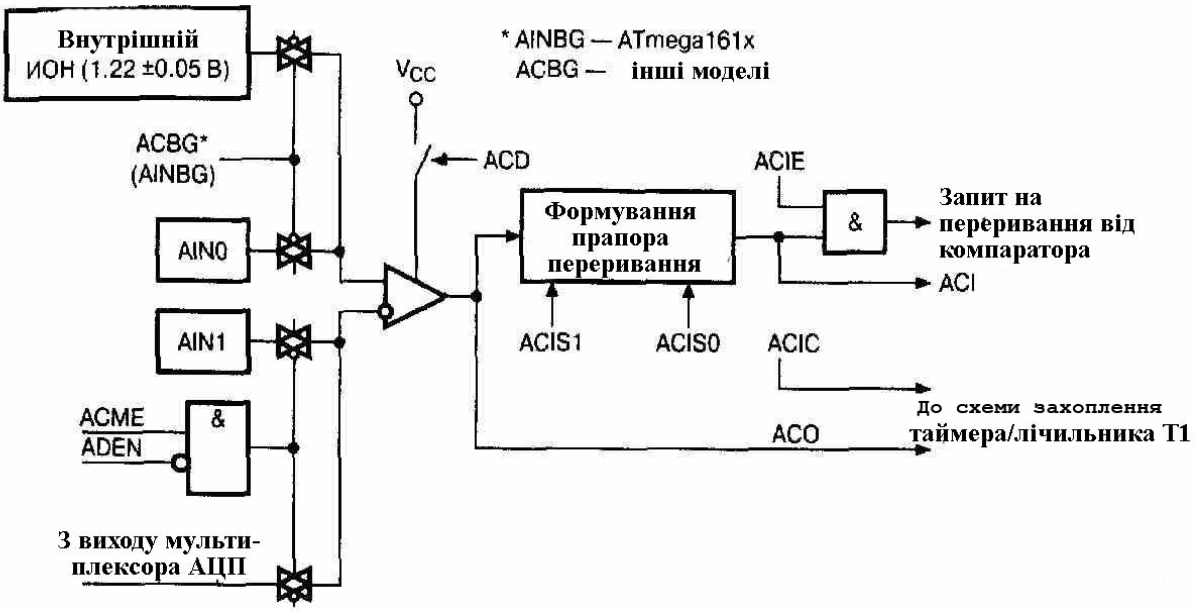


Рисунок 2.37 – Структурна схема аналогового компаратора

Аналогово-цифровий перетворювач

Структурна схема модуля АЦП наведена на рис. 2.38.

Основні параметри АЦП такі:

- абсолютна похибка ± 2 МЗР;
- інтегральна нелінійність ± 0.5 МЗР;
- швидкодія до 15 тис. виборок/с.

На вході модуля АЦП є 8-канальний аналоговий мультиплексор, що надає в розпорядження користувача 8 каналів з несиметричними входами.

У моделях Atmega16x входи АЦП можуть також об’єднуватися попарно для формування в цілому до 13 каналів з диференціальним входом. Два канали при цьому мають можливість 20- і 200-кратного попереднього посилення вхідного сигналу.

При коефіцієнтах посилення lx і 20x дійсна роздільна здатність становить 8 розрядів, а при коефіцієнті посилення 200x – 7 розрядів.

Джерелом опорної напруги для АЦП може бути як напруга живлення мікроконтролера, так і внутрішнє або зовнішнє джерело опорної напруги.

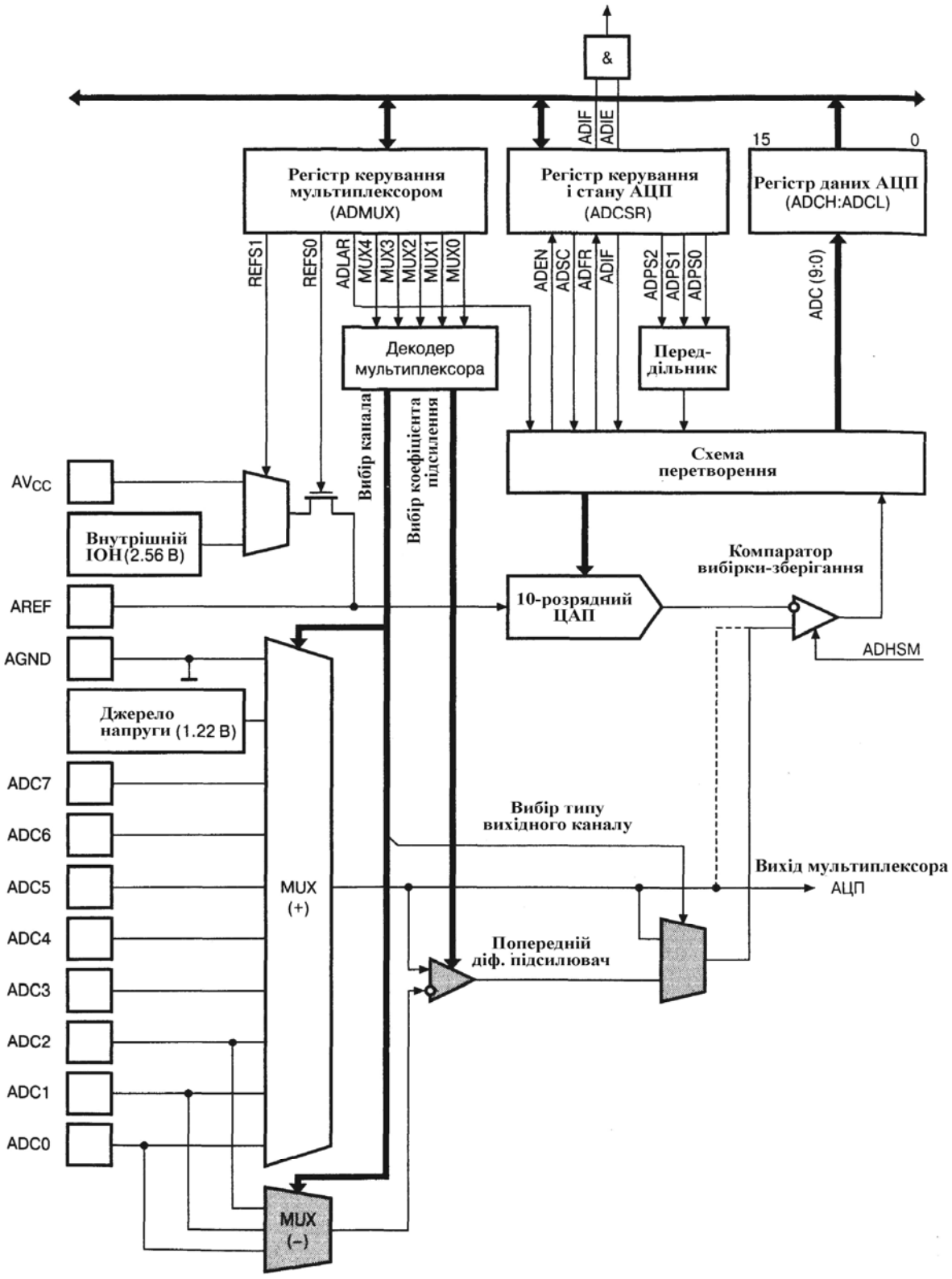


Рисунок 2.38 – Структурна схема модуля АЦП

У процесі роботи АЦП може функціонувати в двох режимах:

- режим одиночного перетворення, коли запуск кожного перетворення ініціюється користувачем;
- режим безперервного перетворення, коли запуск перетворень виконується безперервно через певні інтервали часу.

2.7. Робота з інтегрованим середовищем для розробки та налагодження програм Bascom AVR

Після запуску Bascom AVR відкривається вікно інтегрованого середовища, на робочому полі якого розміщується багатовіконний текстовий редактор, де розміщається лістинг програми, як показано на рис. 2.39.

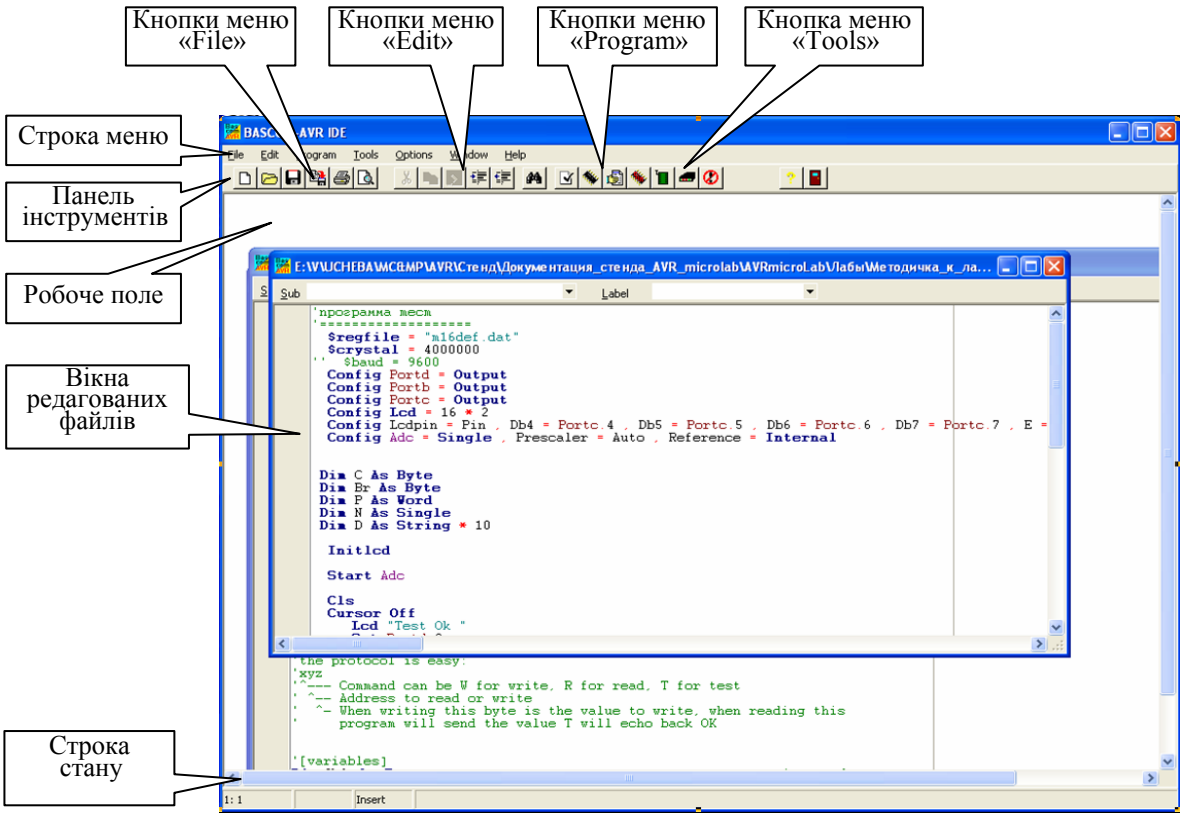


Рисунок 2.39 – Вікно середовища Bascom AVR і його секції

Перед початком роботи треба налагодити вікно редактора таким чином, щоб на екрані комп'ютера програми мали той же вигляд, що й у прикладах, для чого треба в меню *Options* ініціалізувати команду *Invirontment* і у вікні, що при цьому відкриється, вибрати опцію *Editor* і там налагодити таблицю опцій за зразком, наведеним на рис. 2.40.

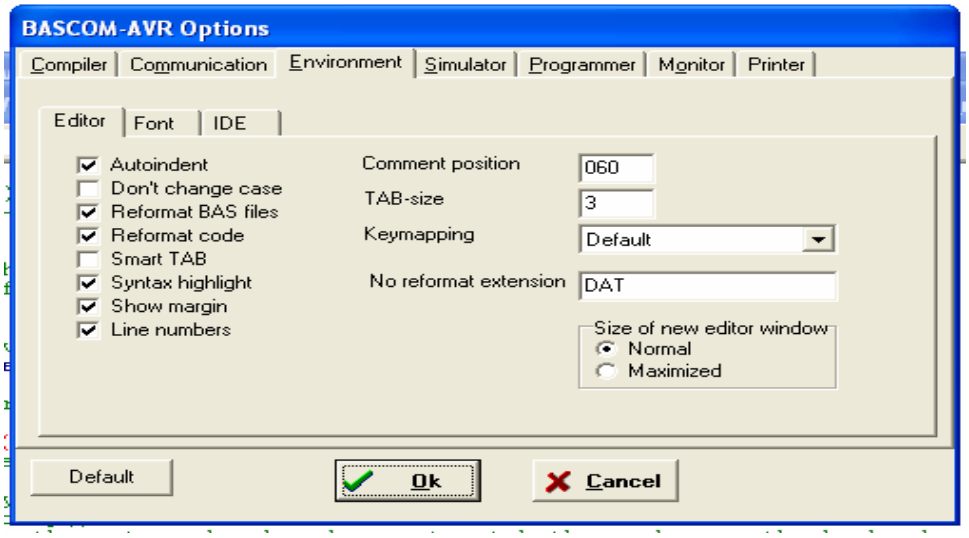


Рисунок 2.40 – Налагодження вікна редактора, опція *Editor*

При виборі опції *Font* таблицю опцій треба налагодити таким чином, як показано на рис. 2.41. Наступним кроком є налагодження опції IDE, для чого треба натиснути кнопку *IDE* і налагодити таблицю, як показано на рис. 2.42.

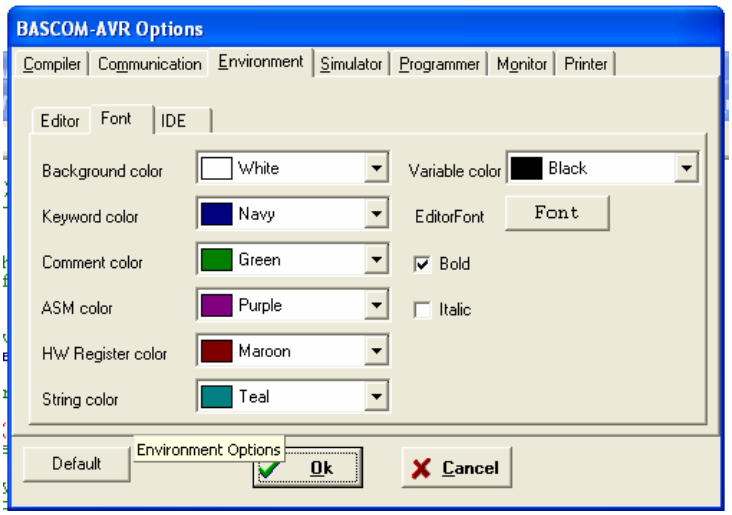


Рисунок 2.41 – Налагодження опції *Font* редактора

У цьому ж вікні треба вказати, де у файловій системі комп'ютера ви збираєтесь розміщувати свої файли. Для цього в діалогове вікно *File location* треба записати шлях до папки, де будуть міститься ваші файли за замовчуванням.

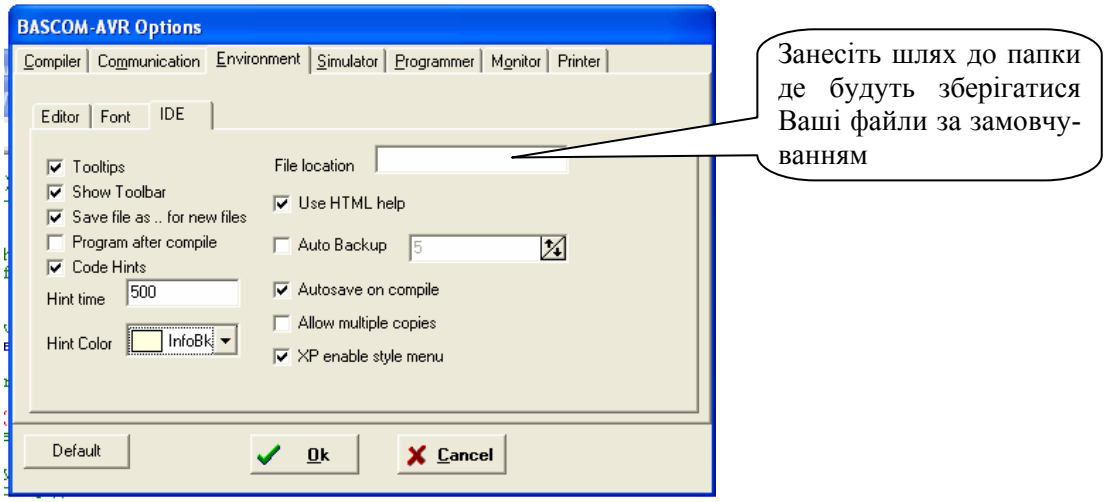


Рисунок 2.42 – Налагодження опції *IDE* редактора

При розробці лістингу програми найбільш корисними будуть спеціальні команди меню «Edit», які ініціалізуються по натисканню вказаних нижче кнопок.

Команда зсуву виділеного блоку тексту лістингу вправо; альтернативний шлях: CTRL+SHIFT+I

Команда зсуву виділеного блоку тексту лістингу вліво; альтернативний шлях: CTRL+SHIFT+U

Команда «Знайти текст» – пошук у тексті лістингу окремих слів, змінних, виразів тощо.

Наступним кроком після розробки програми є компіляція джерельного модуля програми, але перед компіляцією треба налагодити компілятор, для чого треба знову звернутися до меню *Options* і ініціалізувати команду *Compiler* і у вікні, що при цьому відкриється, налагодити таблицю опцій за зразком, наведеним на рис. 2.43.

Дуже важливим для цього є вказівка компілятора про тип мікрококнтролера, для якого розроблюється програма, оскільки кожен тип мікроконтролеру має свої особливості стосовно системи команд, і ця вказівка є основою коректної компіляції програми.

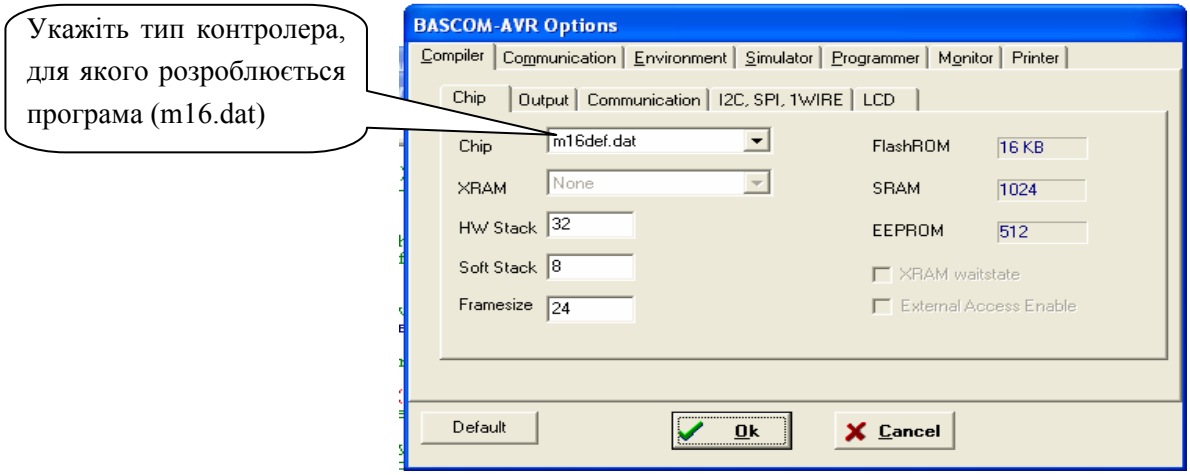


Рисунок 2.43 – Налагодження компілятора опції *Compiler*

Компіляція програми

Після розробки програми у вікні вбудованого текстового редактора файл програми має бути збереженим у якій-небудь папці файлової системи комп'ютера. Після цього файл має бути компільованим. Запуск компілятора може бути виконаним одним з трьох способів:

1) натисканням функціональної клавіши F7;

2) вибором команди *Compile* з меню *Program*;

3) натисканням кнопки  на інструментальній панелі.

Компілятор у результаті оброблення джерельного модуля програми в активній папці з джерельним модулем з розширенням .BAS автоматично генерує ще 7 файлів, призначення яких зазначено в табл. 2.22.

Таблиця 2.22 – Призначення файлів, що генеруються компілятором

Файл	Призначення файлу
xxx.BIN	Бінарний файл(Binary File) – програмний код, який має бути записаним у пам'ять програм мікроконтролера
xxx.DBG	Файл налагодження (Debug file), який використовується емулятором
xxx.OBJ	Об'єктний файл (Object file) для емуляції у вбудованому емуляторі. Він також може бути використаним у стимуляторі AVR Studio
xxx.HEX	16-ковий файл у форматі Intel hexadecimal file, який може бути використаний в інших програмах
xxx.ERR	Файл помилок (Error file). Генерується тільки при знайденні помилок
xxx.RPT	Файл рапорту (Report file)
xxx.EEP	Файл образу EEPROM (EEPROM image file)

У разі випадку серйозних помилок на екран виводиться діалогове вікно з повідомленням про помилки, що не можуть бути визначені автоматично, і процес компілювання завершується.

Прості помилки виводяться у вигляді повідомлення у нижній частині вікна редактора над рядком статусу.

Якщо «клікнути» на рядку з інформацією про помилку, то редактор переведе курсор у рядок джерельного модуля програми з помилкою. Цей рядок в лістингу виділяється червоним кольором.

Після виправлення помилки файл має бути рекомпільованим.

Пошук синтаксичних помилок у програмі

За бажанням користувача до компіляції програми може бути задіяний пошук синтаксичних помилок, який також виконує компілятор, але в цьому випадку він генерує тільки файл помилок, якщо вони знаходяться.

Ця опція може бути виконаною одним з трьох шляхів:

1) натисканням функціональних клавіш CTRL + F7;

2) вибором команди *Syntax check* з меню *Program*;

3) натисканням кнопки  на інструментальній панелі.

Емуляція програми

Після компіляції програми вона може бути записана в пам’ять програм мікропроцесора або емульована яким-небудь емулятором.

Емуляція – моделювання на комп’ютері роботи мікроконтролера при виконанні ним програми, де моделюється робота усіх функціональних пристроїв мікроконтролера – регістрів мікропроцесора, пам’яті і т. ін.

Емулятор дозволяє налагодити програму на «апаратному» рівні – знайти помилки в алгоритмі, в той час як компілятор, перетворюючи джерельний модуль програми в модуль, що виконується, знаходить тільки синтаксичні помилки.

Емулятор дозволяє бачити, як змінюються дані у всіх основних функціональних пристроях мікроконтролера під час виконання програми.

Дане середовище може використовувати або свій вбудований емулятор, або емулятори інших інтегрованих середовищ, наприклад, AVR Studio, за вибором користувача.

Емулятор можна активізувати одним з трьох шляхів:

1) натисканням функціональної клавіші F2;

2) вибором команди *Simulate* з меню *Program*;

3) натисканням кнопки  на інструментальній панелі.

Емулятором використовуються .OBJ і .DBG-файли.

На рис. 2.44 показано вигляд вікна вбудованого емулятора, де виносками пояснене призначення полів вікна і кнопок керування процесом емуляції.

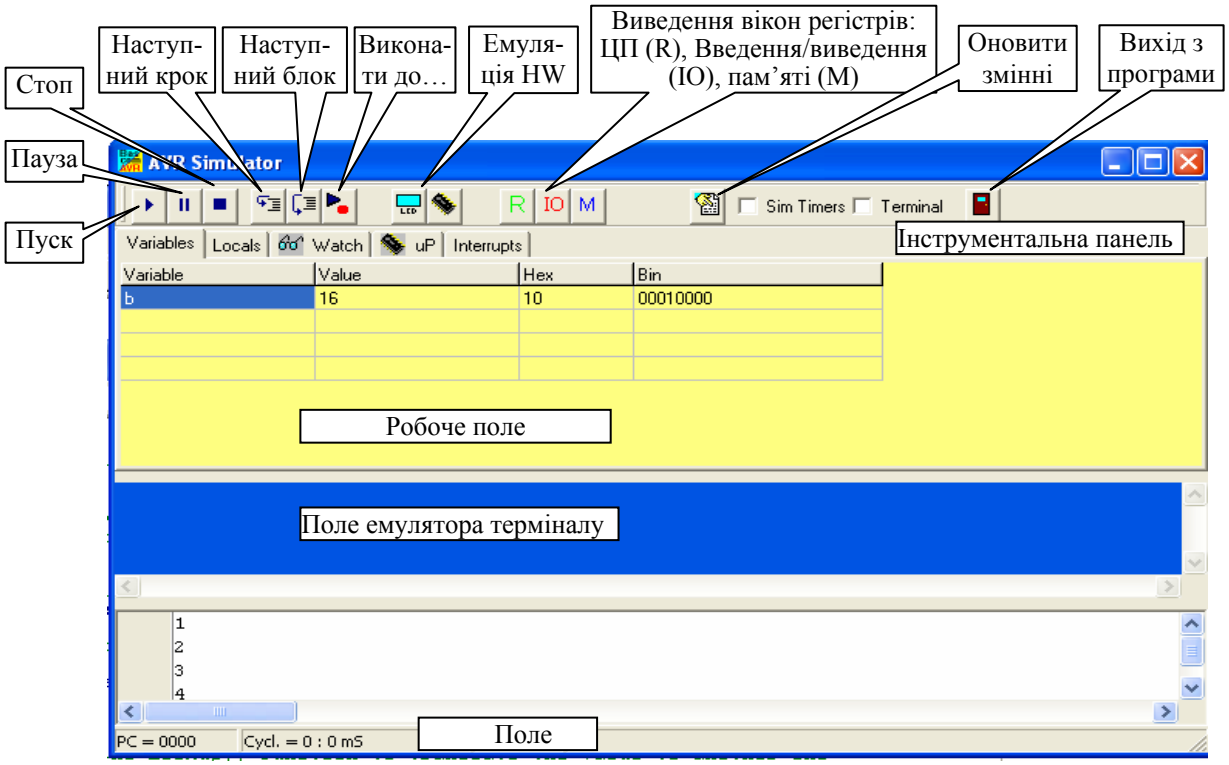


Рисунок 2.44 – Вигляд вікна вбудованого емулятора

Для повноти опису можливостей емулятора зупинимось на розгляді функцій полів вікна і кнопок інструментальної панелі.

Вікно емулятора поділене на декілька полів:

- поле інструментальної панелі;
- робоче поле;
- поле емулятора терміналу;
- поле статусу.

Інструментальна панель

 Це кнопка «ПУСК» емулятора (альтернативний пуск – по F5).

 Це кнопка «ПАУЗА» в роботі емулятора. Після натискання цієї кнопки доцільним є покрокове виконання емуляції, що виконується натисканням функціональної клавіші F8 або кнопки «ПУСК».

 Це кнопка «ЗУПИНКА» роботи емулятора. Після зупинки емуляції усі змінні набувають початкового стану і продовження емуляції можливе тільки спочатку – за командою пуску.

 Це кнопка «КРОК». Натисканням цієї кнопки (або F8) буде емулюватись тільки один рядок програмного коду BASIC програми. Емулятор перейде до стану ПУСК, а після виконання рядка програми перейде у стан ПАУЗА. Якщо натиснути кнопку F8 ще раз і утримувати її довго, то почнеться неперервне емулювання програми до повторного натискання F8, тоді емулятор перейде у стан ПАУЗА.

 Це кнопка НАСТУПНИЙ КРОК (альтернатива SHIFT+F8). Ця кнопка має той же ефект, що і кнопка КРОК, але підпрограми будуть виконуватись повністю. Емулятор не може працювати в режимі КРОК підпрограми.

 Це кнопка ПУСК ДО. Після натискання цієї кнопки емулятор буде виконувати емулювання програми до рядка програми, де стоїть курсор. Цей рядок програми має бути виконуваним. Перед натисканням цієї кнопки необхідно заздалегідь поставити курсор у заданий рядок програми, де має бути зроблена зупинка емулювання.

 Це кнопка виведення вікна стану регістрів процесора.

У цьому вікні значення регістрів виводяться у шістнадцятковому форматі. Для зміни значення регістру треба клацнути мишкою у колонці *Val* таблиці, де з клавіатури задати нове значення регістру.

 Це кнопка виведення вікна стану регістрів введення/виведення процесора.

Робота з вікном введення/виведення аналогічна роботі з вікном стану регістрів. Пусті рядки означають, що за цима адресами немає регістрів введення/виведення. (Ці регістри можуть бути видалені пізніше).

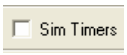
 Натискання цієї кнопки виводить вікно стану пам’яті.

Значення комірок пам’яті можуть бути змінені аналогічно вікну стану регістрів.

При переміщенні від комірки до комірки пам’яті у поле статусу виводиться змінна, яка записана за цією адресою.

Вкладка SRAM показує вміст внутрішньої пам’яті й пам’яті XRAM. Вкладка EEPROM показує вміст пам’яті СППЗП.

 Кнопка відновлення змінних відновить усі змінні на шині (F5). Коли використовують емулятор апаратного забезпечення, якщо включили цю опцію, LEDS відновлять свій стан. Використання цієї опції сповільнить емуляцію. Саме тому це – опція. Коли використовують F8 для покрокового налагодження коду, не слід включати цю опцію, оскільки змінні обновляються після кожного кроку.



Якщо буде потреба емулювати внутрішні таймери мікропроцесором, необхідно включити цю опцію. Моделювання таймерів використовує багато процесорного часу, тому не слід включати цю опцію в більшості випадків. Коли ви налагоджуєте код таймера, корисно емулювати таймери.

Емулятор підтримує основні режими таймера. Через те що є багато нових мікросхем з новими режимами таймера, можливо, що емулятор не під-тримує усі режими. Якщо буде потреба емуляції таймера, кращим варіантом може бути використання останньої версії AVR Studio і завантаження BASCOM Object file.

Навіть AVR Studio має деякі недоліки, тому найкращою залишається перевірка коду на реальній мікросхемі.

Ця опція дозволяє використати термінал емулятора для імітації послідовної передачі даних.

Звичайно емулятор виводить послідовний сигнал вихідного пристрою в синьому вікні, і також можна ввести дані, які потрібно послати послідовному порту.

Коли ви включаєте опцію *Terminal*, дані відсилаються фактично послідовному порту, і буде відображено отримання даних послідовним портом.

Під секцією інструментальної панелі є вкладка з безліччю сторінок.

Ця вкладка дозволяє переглядати значення змінних програми. Ви можете підсумувати змінні, двічі клацнувши по колонці *Variable*. Зявиться список, з якого можна вибрати змінну.

Щоб переглянути змінну типу масив, наберіть ім’я змінної з індексом.

Під час емуляції ви можете змінити значення змінних у колонкці *Value*, *Hex*-колонці або *Bin*-колонці. Ви повинні натиснути ENTER, щоб зберегти зміни.

Щоб видалити змінну, треба нажати CTRL+DEL.

Щоб ввести більше змінних, натисніть клавішу DOWN, і новий ряд стане видимим.

Також можна відображати змінну, вибираючи її у вікні коду і натискаючи ENTER. Вона буде додана до списку змінних автоматично.

Зверніть увагу, що відновлення змінних вимагає часу. Тому видаліть змінні, які не повинні відображатися, для прискорення емуляції.

Вікно LOCALS показує змінні, знайдені в SUB або FUNCTION. Відображаються тільки локальні змінні. Ви не можете підсумувати змінні в секції LOCALS.Зміна значення локальних змінних працює так само, як на вкладці *Variable*.

Вкладка *Watch* може використовуватися, щоб увести вираз, який буде оцінено під час емуляції. Якщо вираз правильний, емуляція призупиняється.

Щоб ввести новий вираз, наберіть вираз у полі тексту нижче кнопки *Remove* і натисніть кнопку *Add*. Коли ви натиснете кнопку *Modify*, емулятор вибере вираз зі списку і тоді замінить значення в полі тексту. Щоб видалити вираз, виберіть необхідний вираз зі списку і натисніть кнопку *Remove*. Під час емуляції, коли вираз стає правильним, відображаються вирази, що були обрані на вкладці *Watch*.

Ця вкладка показує значення регістра-статусу мікропроцесора (SREG). Прапори можуть бути змінені шляхом кліку по прапорцю.

Показує стек програмного забезпечення, стек апаратного забезпечення і значення указівника структури. Також показує мінімальне або максимальне значення, досягнуті під час емуляції. Коли одна із цих областей даних вводиться або накладається на іншу, відбувається переповнення стека або структури.

Про це буде повідомлено паузою й прапорцем.

Натискання кнопки *snapshot* збереже знімок значень регістра струму і створить копію в пам’яті. Зверніть увагу, що кнопка *Snapshot* зміниться на *Stop.* Тепер виконайте який-небудь код, натискаючи F8 і натисніть кнопку *Snapshot* знову. З’явиться вікно, що покаже Усі змінені адреси розміщень.

Це може допомогти визначити, що регістр або пам’ять використовує значення.

Коли ви записуєте ISR (Програма оброблення переривання) з опцією NOSAVE, ви можете використовувати це рішення, регістри якої використовують і потім зберігають тільки зміни регістрів.

Interrupts Ця вкладка показує джерела переривань. Коли жодна програма оброблення переривання не запрограмована, всі кнопки будуть неактивними.

Коли ви записали ISR (використовуючи *ON INT*...), кнопка для відповідного переривання буде активна. Тільки використовувані переривання будуть активні. Клацніть по кнопці переривання – буде виконуватися відповідна програма оброблення переривань. Цим ви емулюєте переривання. Включення *«Sim Timers*» може також запустити подію.

Генератор імпульсів може використовуватися для генерації імпульсів таймеру, коли використовується зустрічний режим. Спочатку виберіть необхідний контакт із блоку зниженої напруги. Залежно від мікросхеми доступні один або більше контактів. Більшість мікросхем має 2 лічильники, тому звичайно 2 вхідних контакти. Потім виберіть число імпульсів і час затримки між імпульсами, після чого натисніть кнопку *Pulse* для генерації імпульсів. Час затримки необхідний, щоб були оброблені й інші завдання. Опція *Sim timers* повинна бути обрана при емуляції таймерів/лічильників.

Вікно термінала

Під вікном із вкладками розташоване вікно емулятора термінала. Це темно-синє поле.

У програмі, при використанні PRINT, вихідний сигнал буде відображатися в цьому вікні. При використанні INPUT у програмі необхідно встановити центр вікна термінала й тип у кращі значення. Можна також вихідний сигнал відправити безпосередньо на COM-порт. Перевірте опцію *Terminal*, щоб увімкнути цю функцію.

Параметри термінала емулятора будуть використовуватися для швидкості в бодах і COM-порту. Будь-які дані, отримані COM-портом, будуть також відображатися у вікні термінала емулятора.

Вікно Джерел

Під вікном термінала розміщується вікно джерел.

Воно містить вихідний текст програми, що емулюється. Усі лінії, які містять виконуваний код, ліворуч означені жовтою крапкою. Ви можете встановити контрольну крапку на цих лініях, вибравши лінію і натиснувши F9. Підводимо курсор миші до імені змінної – у рядку стану з'явиться значення змінної. Якщо ви виберете змінну й натиснете ENTER, то вона буде додана до вікна *Variable*. Щоб використати функціональні клавіші (наприклад, F8 для покрокового налагодження), центр повинен бути встановлений у вікні джерел. Синя стрілка вказує на рядок, що виконається наступним. Натиснувши кнопку емуляції апаратного забезпечення  , побачимо показані нижче вікна.

Верхня секція – віртуальний LCD-дисплей. Він показує код у режимі PIN і шинному режимі. Для шинного режиму емулятор підтримує тільки 8-бітний режим.

Нижче LCD-дисплея – область світлодіодів, що дають візуальну індикацію щодо портів. Клацніть по світлодіодам для перемикання.

PA означає PORT A, PB означає PORT B, і т. д.

IA означає PIN A, IB означає PIN B і т.д.

Порти будуть відображатися залежно від обраного типу мікропроцесора.

Праворуч від контактів світлодіодів розташована відслідковуюча панель. Вона використовується для емуляції вхідної напруги, застосовуваної при аналого-цифрових перетвореннях. Зверніть увагу: не всі мікросхеми мають АЦП. Ви можете встановити значення для кожного каналу, вибираючи необхідний канал нижче панелі, що відслідковує.

Поруч із панеллю, що відслідковує, цифрова клавіатура. Ця допоміжна клавіатура використовується для емуляції GETKBD () функції.

При емуляції клавіатури важливо пам'ятати, що натискати/клацати кнопку клавіатури необхідно перед емулюванням getkbd () рядка.

Для емуляції компаратора, встановіть рівень вхідної напруги, використовуючи IN0.

Включення емуляції апаратного забезпечення

Клацнувши кнопкою  , ви можете емулювати фактично вбудовані порти мікропроцесора. Використовувана мікросхема мікропроцесора

повинна мати послідовний порт. Для емуляції апаратного забезпечення необхідно скомпілювати файл basmon.bas.

Приклад компіляції

Переконайтесь, що використовується DT006 simmstick і мікросхема Mega16 AVR. Відкрийте файл basmon.bas і змініть рядок $REGFILE = “xxx” на $REGFILE = «m16def.dat». Скомпілюйте програму і запрограмуйте мікросхему. Краще встановлювати лок-біти так, щоб монітор не записував поверх, при випадковому натисканні F4.

Емуляція апаратного забезпечення працює, коли установка має послідовний порт. З’єднайте кабелем COM-порт вашого PC і DT006. Імовірно буде одне з’єднання. Звичайно воно використається для відправлення даних емулятору за командою PRINT.

Програма моніторингу компілює програму для швидкості 19200 бод.

У параметрах Options Communication повинна вказуватися та ж швидкість у бодах.

Ті ж самі налаштування для програми моніторингу використаються й для термінала емулятора, так що виберіть вхід COM і швидкість у бодах – 19200.

Включіть або скиньте DT006. Вона, ймовірно, вже включена з тих пір, як скомпілювали програму basmon.bas і запам’ятали її в m16.

Коли ви натисните кнопку емуляції апаратного забезпечення, емулятор відішле або одержить дані, якщо порт, контакт або регістр DDR змінюють значення. Це дозволяє емулювати рідкокристаличний дисплей апаратного забезпечення, або щось простіше, наприклад, світлодіод. У папці SAMPLES знаходиться програма DT006. Ви можете скомпілювати програму й натиснути F2. Коли ви покроково виконуєте програму, світлодіоди змінюються.

Усі команди можуть бути емульовані таким чином, але вони повинні мати змогу використати статичну синхронізацію. Це означає, що однопровідне з’єднання не буде працювати, через те що воно залежить від синхронізації. I2C має статичну шину, й тому буде працювати.

Коли програма зависає, це означає, що щось пішло не так з послідовною передачею даних. Єдиний спосіб уникнути подібного полягає

в тому, щоб натиснути кнопку Real hardware Simulation знову. Real hardware Simulation – ефективний спосіб тестувати апаратне забезпечення.

 Кнопка відновлення змінних обновить усі змінні під час роботи (F5).

Якщо включено цю опцію, світлодіоди тільки обновлять свій стан при використанні емулятора апаратного забезпечення. Використання цієї опції сповільнить емуляцію.

Емуляція сторожового таймера

Більшість мікросхем AVR мають внутрішній сторожовий таймер. Цей таймер синхронізований із внутрішнім генератором. Частота – приблизно 1 Мгц. Напруга й температурні коливання впливають на сторожовий таймер. Це не дуже точний таймер. У такий спосіб при відновленні/ скиданні сторожового таймера необхідно враховувати деякий допуск. Емулятор попередить, коли відбудеться переповнення сторожового таймера, але тільки якщо сторожовий таймер включений.

Рядок стану

Рядок стану показує ПЛ (програмний лічильник) і число циклів. Можна скинути цикли, підвівши курсор миші на рядок стану й клацнувши правою клавішею. Відкриється контекстне меню з опцією скидання циклів. Його можна використати, щоб визначити, скільки часу необхідно команді програми. Показаний час може також залежати від значення змінної.

2.8. Мікроконтролери сімейства MSP430

Питання вибору мікроконтролера має принципове значення, оскільки його результат багато в чому визначає не тільки сукупність можливих технічних характеристик майбутньої системи, але й весь спектр потенційних проблем, пов’язаних із процесом розробки, виробництва, реалізації й можливих доробок у майбутньому.

За даними дослідницького центру Harbor Research, Inc., отриманим в результаті анкетування, значущість факторів, що впливають на вибір сімейства мікроконтролерів для нової розробки, подана на рис. 2.45.

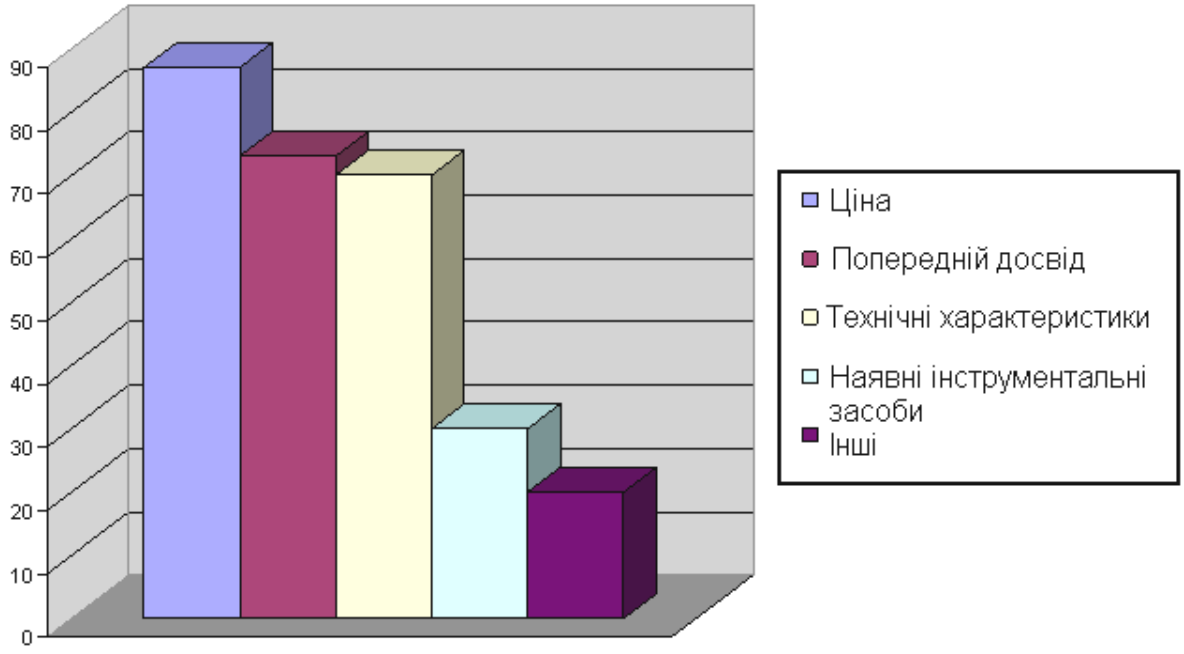


Рисунок 2.45 – Значущість факторів, які впливають на вибір МК

Із усього різноманіття мікроконтролерів, від 4- до 32-розрядних, одним із лідерів є сімейство 16-розрядних мікроконтролерів MSP430 із украй низьким енергоспоживанням, вироблене корпорацією Texas Instruments (TI) [36].

В останні роки накреслилася тенденція до використання в мікроконтролерах замість традиційних ПЗП (масочних, однократно програмувальних або перепрограмувальних) пристроїв флеш-пам'яті. Переваги цієї технології очевидні – можливість багаторазового перепрограмування в цільовій системі, нагромадження й відновлення даних безпосередньо із прикладної програми. У випадку Texas Instruments дана тенденція виявилася в розширенні популярного сімейства мікроконтролерів MSP430 рядом пристроїв із флеш-пам'яттю (MSP430F) [36].

Короткий перелік основних характеристик мікроконтролерів MSP430F:

– 16-розрядна RISC-архітектура, виконання регістрових інструкцій за один машинний цикл;

– робоча частота від 0 до 8 Мгц;

– напруга живлення від 1,8 В до 3,6 В;

– украй низьке споживання: від 1,3 до 2,5 мкА при частоті 4 кГц і напрузі живлення 2,2 В; від 160 до 280 мкА при частоті 1 Мгц і напрузі живлення 2,2 В;

– п’ять режимів економії енергії;

– споживання в черговому режимі (standby mode) від 0,7 до 1,6 мкА;

– споживання в режимі відключення зі збереженням умісту ОЗП (режим сну) – 0,1 мкА;

– пробудження із чергового режиму за 6 мкс;

– можливість програмування в цільовій системі через послідовний інтерфейс;

– досить широкий набір інтегрованої периферії;

– діапазон робочих температур від –40 до +85 °С.

2.8.1. Архітектура MSP430

Архітектура MSP430 подана на рис. 2.46 [36]. Мікроконтролери сімейства MSP430 містять 16-розрядне RISC CPU, периферійні модулі й гнучку систему тактування, з’єднані через нейманівську загальну адресну шину (MAB) пам’яті й шину пам’яті даних (MDB). Поєднуючи сучасне CPU з відображуваними в пам’яті аналоговими й цифровими периферійними пристроями, сімейство MSP430 [36] пропонує рішення для застосувань зі змішаними сигналами:

а) сімейство MSP430 має такі особливості:

– архітектура з ультранизьким споживанням, що збільшує час роботи при живленні від батарей:

– для збереження вмісту ОЗП необхідний струм не більше 0,1 мкА;

– модуль тактування реального часу споживає 0,8 мкА;

– струм споживання при максимальній продуктивності становить 250 мкА;

б) високоякісна аналогова периферія для виконання точних вимірів:

– вбудовані модулі 12-розрядного або 10-розрядного АЦП швидкістю 200 ksps;

– температурний датчик і джерело опорної напруги VRef;

– здвоєний 12-розрядний ЦАП;

– таймери, керовані компаратором для виміру резистивних елементів;

– схема спостереження (супервізор) за напругою живлення;

– 16-розрядне RISC CPU, що допускає нові застосування до фрагментів коду;

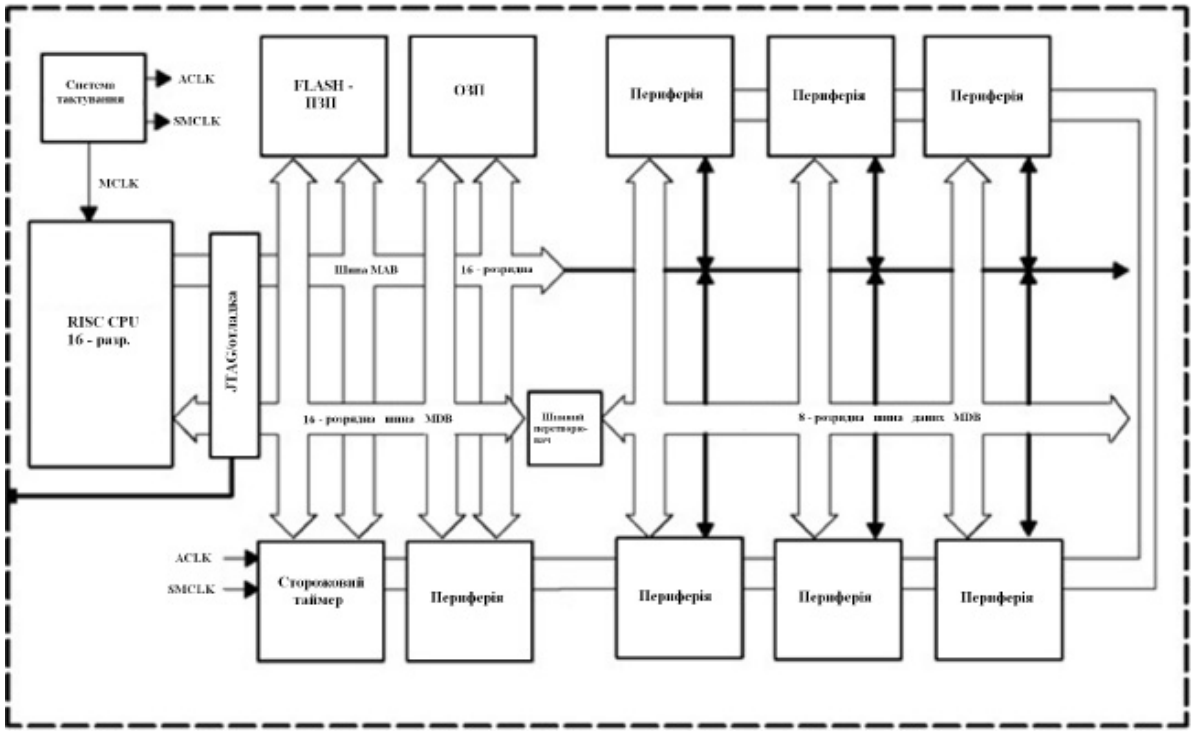


Рисунок 2.46 – Архітектура MSP430

– великий регістровий файл знімає проблему «вузького файлового горлечка»;

– компактне ядро має знижене енергоспоживання і вартість;

– ядро оптимізовано для сучасного високорівневого програмування;

– набір команд складається з 27 інструкцій, підтримується сім режимів адресації;

– розширені можливості векторних переривань.

Можливість внутрішньосхемного програмування flash-пам’яті дозволяє гнучко змінювати й обновляти програмний код, робити реєстрацію даних.

Система тактування розроблена спеціально для використання в застосуваннях з живленням від батарей. Допоміжна низькочастотна система тактування (ACLK) працює безпосередньо від звичайного 32 кГц годинникового кристала. Модуль ACLK може використатися як фонова система реального часу з функцією самостійного «пробудження». Інтегрований високошвидкісний осцилятор із цифровим керуванням (DCO) може бути джерелом основного тактування (MCLK) для ЦПУ й високошвидкісних периферійних пристроїв. Модуль DCO стає активним і ста-

більним менш чим через 6 мкс після запуску. Рішення на основі архітектури MSP430 дозволяють ефективно використати високопродуктивне 16-розрядне RISC CPU у дуже малі проміжки часу: низькочастотна допоміжна система тактування забезпечує роботу мікроконтролера в режимі ультранизького споживання потужності; активізація основного високошвидкісного модуля тактування дозволяє виконати швидке оброблення сигналів.

Спеціальна вбудована логічна підсистема емуляції перебуває безпосередньо в пристрої й доступна через JTAG без використання додаткових системних ресурсів. Вигоди вбудованої емуляції полягають у такому:

– можлива фонова розробка й налагодження на повній робочій швидкості виконання програми;

– підтримується використання контрольних точок і покрокове виконання програми;

– об’єкт внутрішньосхемної розробки має ті ж характеристики, що й у кінцевому пристрої;

– зберігається цілісність змішаних сигналів, на яку не впливають перешкоди кабельного розведення.

Сімейство MSP430 має нейманівську архітектуру з єдиним адресним простором для регістрів спеціального призначення (SFR), периферії, ОЗП й flash-пам’яті програм. Конкретний розподіл пам’яті можна довідатися з довідкових даних на пристрій, що цікавить. Доступ до програмного коду виконується завжди за парними адресами. Дані можуть бути доступні як байти або як слова.

Загальна ємність адресної пам’яті становить 64 Кбайт, з урахуванням передбачуваного розширення.

Початкова адреса flash-пам’яті залежить від ємності наявної пам’яті й відрізняється для різних пристроїв. Кінцева адреса flash-пам’яті завжди 0FFFFh. Flash-пам’ять може використатися як для програмного коду, так і для даних. Байти або слова таблиць даних можуть зберігатися й використовуватися безпосередньо у flash-пам’яті, що виключає необхідність копіювати ці таблиці в ОЗП перед подальшим використанням.

Таблиця векторів переривань займає верхні 16 слів адресного простору flash-пам’яті, при цьому вектор переривання з найвищим пріоритетом знаходиться в самому верхньому адресному слові flash-пам’яті (0FFFEh).

ОЗП починається з адреси 0200h. Кінцева адреса ОЗП залежить від ємності поданої пам'яті й розрізняється для кожного конкретного пристрою. ОЗП може використатися як для програмного коду, так і для даних.

Периферійні модулі відображені в адресному просторі. Адреси з 0100 до 01FFh зарезервовані для 16-розрядних периферійних модулів. Вони будуть доступні за допомогою команд-слів. Якщо використовуються однобайтові команди, допустимі тільки парні адреси, при цьому старший байт результату завжди буде містити «0».

Адресний простір з 010h по 0FFh зарезервовано для 8-розрядних периферійних модулів. Ці модулі доступні за допомогою однобайтних команд. Читання байтів модулів за допомогою команд-слів призведе до появи в старшому байті непередбаченого вмісту.

Якщо в байт модуля будуть записуватися дані у вигляді слова, то в регістрі периферійного модуля збережеться тільки молодший байт цього слова, старший буде знехтуваний.

Деякі функції периферії конфігуруються в регістри спеціального призначення (SFRs). Регістри спеціального призначення розташовані в нижчих 16-ти байтах адресного простору і організовані у вигляді байтів. Звертання до регістрів SFRs виконується тільки з використанням однобайтних команд. Призначення окремих бітів регістрів SFRs описано в технічних рекомендаціях на кожний конкретний пристрій.

Байти розташовані в парних або непарних адресах. Слова розташовуються тільки в парних адресах. При роботі з командами-словами повинні використовуватися тільки парні адреси. Молодший байт слова завжди розташований за парною адресою. Старший байт – у наступній непарній адресі. Наприклад, якщо слово даних розташоване за адресою xxx4h, то молодший байт слова даних буде мати адресу xxx4h, а старший байт слова адресу xxx5h.

2.8.2. Центральний процесорний пристрій MSP430

Центральний процесорний пристрій (ЦПП, англ. CPU) зображений на блок-схемі, рис. 2.47 [36]. Він включає можливості, спеціально створені для сучасних технологій програмування, що зазначаються як оброблення таблиць і використання мов високого рівня, подібних мові *C*.

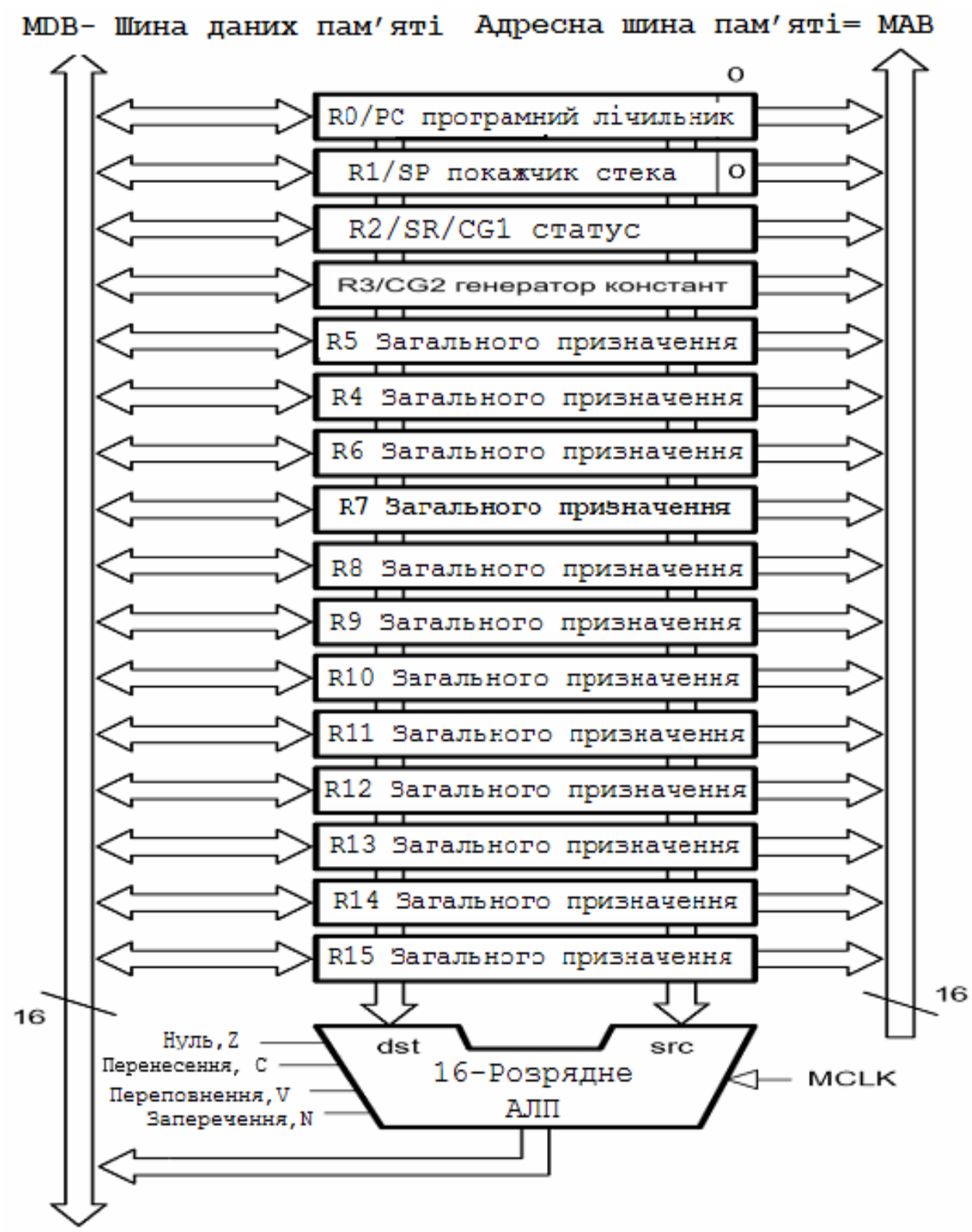


Рисунок 2.47 – Блок-схема ЦПП MSP430

Центральний процесорний пристрій може виконувати адресацію в повному адресному діапазоні без використання сторінок пам’яті [36].

ЦПП має такі можливості:

– RISC-архітектура з 27 командами й 7 режимами адресації;

– ортогональна архітектура, при якій кожна команда придатна для кожного режиму адресації;

– повний доступ до всіх регістрів, включаючи програмний лічильник, регістри статусу й указівник стека;

– однотактні регістрові операції;

– великий 16-розрядний регістровий файл, що зменшує кількість звертань до пам’яті;

– 16-розрядна адресна шина, що забезпечує прямий доступ і розгалуження у всьому діапазоні пам’яті;

– 16-розрядна шина даних, що дозволяє прямо маніпулювати параметрами шириною в слово;

– генератор констант негайно надає шість найбільше використовуваних значень, зменшуючи розмір коду;

– прямий обмін між комірками пам’яті без проміжного запису в регістр;

– команди й адресація у форматах «слово» і «байт».

ЦПП включає шістнадцять 16-розрядних регістрів. Регістри R0, R1, R2 і R3 мають спеціальне призначення. Регістри з R4 по R15 є робочими регістрами загального призначення.

16-розрядний програмний лічильник (PC/R0) указує на наступну команду, що буде виконуватися. Кожна команда складається з парного числа байтів (два, чотири або шість), тому PC інкрементується відповідно.

Команди доступу в адресному просторі 64 Кбайт виконуються до меж слів, тому PC вирівнюється до парних адрес.

Програмний лічильник PC може бути адресований всіма командами і у всіх адресних режимах.

Указівник стека (SP/R1) використовується ЦПП для зберігання адрес повернення з підпрограм і переривань.

Стек оснований на передінкрементній-постінкрементній схемі.

Крім того, указівник стека SP може використовуватися з усіма командами і у всіх адресних режимах.

Указівник стека SP ініціалізується в ОЗП користувачем і вирівнюється до парних адрес.

2.8.3. Набір команд MSP430

Повний набір команд сімейства MSP430 [36] містить 27 команд ядра і 24 емульовані команди, що подані в табл. 2.23.

Таблиця 2.23 – Набір команд MSP430

Мнемоніка		Опис		V	N	Z	C
ADC(.B)*	dst	Додавання біта C з одержувачем	dst + C → dst	*	*	*	*
ADD(.B)	src, dst	Додавання джерела з одержувачем	src + dst → dst	*	*	*	*
ADDC(.B)	src, dst	Додавання джерела й біта C з одержувачем	src + dst + C → dst	*	*	*	*
AND(.B)	src, dst	Операція «І» джерела й одержувача	src .and. dst → dst	0	*	*	*
BIC(.B)	src, dst	Очищення бітів в одержувачі	.not.src .and. dst → dst	–	–	–	–
BIS(.B)	src, dst	Установлення бітів в одержувачі	src .or. dst → dst	–	–	–	–
BIT(.B)	src, dst	Перевірка бітів в одержувачі	src .and. dst	0	*	*	*
BR*	dst	Перехід за призначенням	dst → PC	–	–	–	–
CALL	dst	Виклик одержувача	PC + 2 → stack, dst → PC	–	–	–	–
CLR(.B)*	dst	Очищення одержувача	0 → dst	–	–	–	–
CLRC*		Очищення біта C	0 → C	–	–	–	0
CLRN*		Очищення біта N	0 → N	–	0	–	–
CLRZ*		Очищення біта Z	0 → Z	–	–	0	–
CMP(.B)	src, dst	Порівняння джерела й одержувача	dst – src	*	*	*	*
DADC(.B)*	dst	Десяткове додавання біта C з одержувачем	dst + c → dst (десяткове)	*	*	*	*

Продовження табл. 2.23

Мнемоніка		Опис		V	N	Z	C
DADD(.B)	src, dst	Десяткове додавання джерела й біта C з одержувачем	src + dst + C → dst (десяткове)	*	*	*	*
DEC(.B)*	dst	Декремент одержувача	dst – 1 → dst	*	*	*	*
DECD(.B)*	dst	Подвійний декремент одержувача	dst – 2 → dst	*	*	*	*
DINT*		Заборона переривань	0 → GIE	–	–	–	–
EINT*		Дозвіл переривань	1 → GIE	–	–	–	–
INC(.B)*	dst	Інкремент одержувача	dst + 1 → dst	*	*	*	*
INCD(.B)*	dst	Подвійний інкремент одержувача	dst + 2 → dst	*	*	*	*
INV(.B)*	dst	Інвертування одержувача	.not.dst → dst	*	*	*	*
JC/JHS	label	Перехід, якщо C установлений / перехід, якщо найвищий або такий же		–	–	–	–
JEQ/JZ	label	Перехід, якщо дорівнює / перехід, якщо Z установлений		–	–	–	–
JGE	label	Перехід, якщо більше або дорівнює		–	–	–	–
JL	label	Перехід, якщо менше		–	–	–	–
JMP	label	Перехід	PC + 2 * зсув → PC	–	–	–	–
JN	label	Перехід, якщо N установлений		–	–	–	–
JNC/JLO	label	Перехід, якщо З не встановлений / перехід, якщо нижчий		–	–	–	–
JNE/JNZ	label	Перехід, якщо не дорівнює, перехід, якщо Z не встановлений		–	–	–	–
MOV(.B)	src, dst	Пересилання джерела в одержувач	src → dst	–	–	–	–
NOP*		Нема операції		*	*	*	*
POP(.B)*	dst	Зняття елемента зі стека в одержувач	@SP → dst, SP + 2 → SP	*	*	*	*

Закінчення табл. 2.23

Мнемоніка		Опис		V	N	Z	C
PUSH(.B)	src	Розміщення джерела в стек	SP – 2 → SP, src → @SP	*	*	*	*
RET*		Повернення з підпрограми	@SP → PC, SP + 2 → SP	0	*	*	*
RETI		Повернення з переривання		*	*	*	*
RLA(.B)*	dst	Арифметична ротація вліво		*	*	*	*
RLC(.B)*	dst	Ротація вліво через C		*	*	*	*
RRA(.B)	dst	Арифметична ротація вправо		*	*	*	*
RRC(.B)	dst	Ротація вправо через C		0	*	*	*
SBC(.B)*	dst	Віднімання not (C) з одержувача	dst + 0FFFFh + C →dst	*	*	*	*
SETC*		Встановлення C	1 → C	–	–	–	1
SETN*		Встановлення N	1 → N	–	1	–	–
SETZ*		Встановлення Z	1 → Z	–	–	1	–
SUB(.B)	src, dst	Віднімання джерела з одержувача	dst + .not.src + 1 → dst	*	*	*	*
SUBC(.B)	src, dst	Віднімання джерела й not (C) з одержувача	dst + .not.src + C → dst	*	*	*	*
SWPB	Dst	Обмін байтів		–	–	–	–
SXT	Dst	Поширення знака		0	*	*	*
TST(.B)*	Dst	Перевірка одержувача	dst + 0FFFFh + 1	0	*	*	1
XOR(.B)	src, dst	Виключаюче «АБО» джерела й одержувача	src .xor. dst → dst	*	*	*	*

Команди ядра – це команди, що мають унікальний код операції, що декодуються ЦПП. Емульовані команди являють собою інструкції, що полегшують читання і написання коду, але не мають власного коду операції, тому Асемблер автоматично змінює їх на еквівалентні команди ядра. Використання емульованих команд не приводить до збільшення обсягу коду або зниження продуктивності.

Існує три формати команд ядра:

1) з подвійним операндом ;

2) з одиночним операндом ;

3) команди переходу.

Усі команди з одним і двома операндами можуть бути командами для роботи з байтами або командами для роботи зі словами, що використовують відповідно розширення «.B» або «.W». Байтові команди використовуються для доступу до даних байта або до байта периферійного пристрою. Команди-слова використаються для доступу до даних слова або до слова периферійного пристрою. Якщо ніяке розширення не використається, команда є командою-словом.

2.8.4. Система синхронізації

Система синхронізації мікроконтролерів MSP430 значною мірою визначає їхні можливості щодо досягнення гранично малого енергоспоживання. На рис. 2.48 подана спрощена структурна схема модуля синхронізації, загальна для всіх пристроїв MSP430F1xxx (генератор XT2 є присутнім не у всіх пристроях) [36]. Модуль системної синхронізації виробляє три тактових сигнали – базову частоту MCLK, допоміжну базову частоту SMCLK і допоміжну частоту ACLK. Сигнал MCLK використається для тактування ЦП, у той час як SMCLK і ACLK у різних комбінаціях використовуються для тактування периферійних пристроїв. Застосування трьох сигналів синхронізації дозволяє розроблювачеві прикладної системи досягти найкращого балансу між продуктивністю і споживаною потужністю.

Як видно з рис. 2.48, первинними джерелами сигналів синхронізації служать один, два або три генератори. Генератори LFXT1 і XT2 можуть працювати з керамічним резонатором із частотою від 450 кГц до 8 Мгц, кварцовим резонатором із частотою від 1 до 8 Мгц або із зовнішнім генератором частоти. Крім цього, генератор LFXT1 може працювати від годинникового кристала на 32768 Гц. Усі аналогові компоненти для 32 кГц кварцу інтегровані в LFXT1 – із зовнішніх компонентів потрібен тільки кристал. Якщо ж використовуються високочастотні резонатори, необхідність використовування навантажувальних конденсаторів зберігається. DCO являє собою RC-генератор, фундаментальна частота якого задається внутрішнім або зовнішнім резистором. Частота DCOCLK може програм-

но змінюватися в широких межах. Використовуючи DCO із внутрішнім резистором, можна побудувати систему, що не використовує ніяких зовнішніх частотозадавальних компонентів. При цьому не слід забувати, що, як і у випадку будь-якого іншого RC-генератора, частота DCO піддана впливу температури, напруги живлення й міняється від пристрою до пристрою, однак є можливість її цифрового підстроювання. Розроблювач має можливість використовувати будь-які комбінації генераторів і синхросигналів виходячи із критеріїв вартості, споживаної потужності й продуктивності розроблювального пристрою.

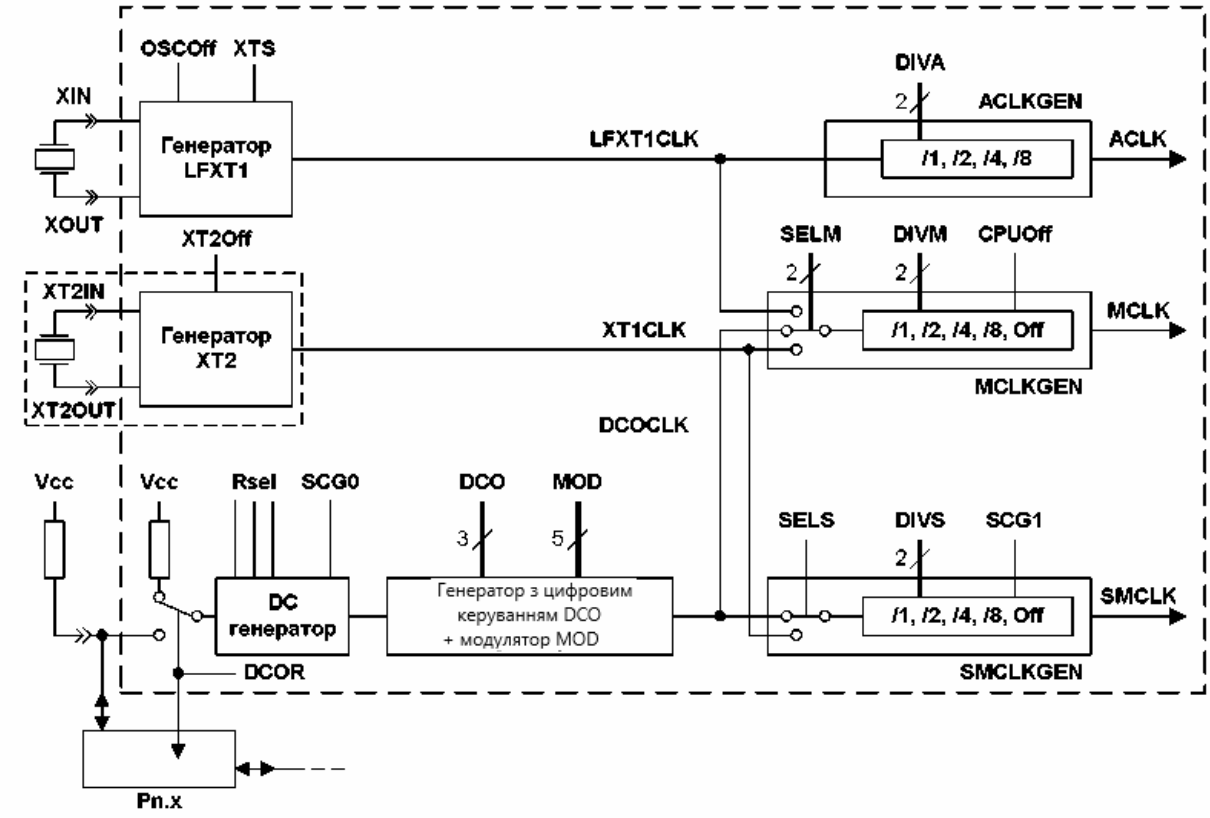


Рисунок 2.48 – Модуль системної синхронізації мікроконтролерів MSP430x11xx

У мікроконтролерах MSP430x4xx система синхронізації була вдосконалена введенням фазового автопідстроювання частоти (ФАПЧ) генератора DCO відносно LFXT1 або XT2, що істотно підвищує стабільність частоти DCO у системах із зовнішнім резонатором, але не позбавляє можливості працювати тільки від частотозадавального резистора (систему ФАПЧ можна програмно відключити). Модуль синхронізації з ФАПЧ одержав назву FLL+.

2.8.5. Енергозберігаючі режими

Крім активного режиму (АМ), коли всі внутрішні синхросигнали ак-тивні, підтримуються також п’ять режимів, що дозволяють тією або іншою мірою знизити споживану мікроконтролером потужність. Ці режими позначаються від LPM0 (Low Power Mode 0) до LPM4 і розрізняються комбінаціями включення й відключення синхросигналів MCLK, SMCLK, ACLK і генератора DCO. У всіх п’яти режимах центральний процесор вимкнений. Режим LPM4 (режим сну) є самим радикальним – припиняється будь-яка активність

МК, включаючи системний генератор, лише зберігається вміст оперативної пам’яті, регістрів і установлення портів введення-виведення. При цьому типова величина споживаного струму знижується до 0,1 мкА. Повернення до активного режиму після кожного із цих станів очікування може бути здійснене за перериванням від працюючої периферії або від RST/NMI. Вихід з режиму LPM4 можливий тільки за дозволеними зовнішніми перериваннями. На додаток до цих режимів, відключенням тих або інших периферійних пристроїв можна досягти ще більшого зниження струму споживання [36].

2.8.6. Цифрові входи/виходи МК

Порти P1–P2 є в пристроях MSP430x11xx. Порти P1–P3 реалізовані в пристроях MSP430x12xx. Порти P1–P6 реалізовані в пристроях MSP430x14, MSP430x15x і MSP430x16x [36].

Пристрої MSP430 мають до 6 портів цифрових входів/виходів від P1 до P6. Кожний порт має 8 виведень входу/виходу. Кожне виведення індивідуально конфігурується як вхід або вихід і кожна лінія введення-виведення може бути індивідуально полічена або записана.

Порти P1 і P2 мають можливість викликати переривання. Для кожної лінії введення/виведення портів P1 і P2 можна індивідуально дозволити переривання і сконфігурувати їх так, щоб переривання відбувалося по фронту або спаду вхідного сигналу. Всі лінії введення-виведення порту P1 є джерелом одного вектора переривання, а всі лінії введення-виведення порту P2 – джерело іншого вектора переривання.

Цифрові входи/виходи мають такі можливості:

– незалежні індивідуально програмувальні входи/виходи;

– будь-які комбінації входу або виходу;

– індивідуально конфігуровані переривання від P1 і P2;

– роздільні регістри даних для входів і виходів.

Цифрові входи/виходи конфігуруються програмним забезпеченням користувача. Невикористовувані ніжки введення-виведення повинні бути сконфігуровані на функцію введення-виведення, у напрямку виведення і залишатися непідключеними на друкованій платі для зменшення споживаної потужності. Значення біта PxOUT може бути будь-яким, оскільки ніжка не підключена.

2.8.7. Таймери

Таймер А – це 16-розрядний таймер/лічильник із трьома регістрами захоплення/порівняння. Він реалізований у всіх пристроях MSP430x1xx [36].

Таймер А може забезпечити безліч захоплень/порівнянь, виходів ШІМ і витримку тимчасових інтервалів. Таймер А також має великі можливості переривання. Переривання можуть бути згенеровані від лічильника при переповненні й від кожного з регістрів захоплення/порівняння.

Таймер А має такі можливості: асинхронний 16-розрядний таймер/лічильник із чотирма режимами роботи; обираний і конфігурований джерелом тактування; три конфігурованих регістри захоплення/порівняння; виходи, що конфігуруються з можливістю ШІМ; асинхронна фіксація (защіпка) входу і виходу; регістр вектора переривань для швидкого декодування всіх переривань таймера А.

Модуль таймера А конфігурується програмним забезпеченням користувача. Настроювання і робота таймера А розглядаються в нижченаведених розділах. Регістр 16-розрядного таймера/лічильника TAR інкрементується або декрементується (залежно від режиму роботи) з кожним наростаючим фронтом тактового сигналу. TAR може бути програмно прочитаний і записаний. Крім того, таймер може генерувати переривання при переповненні. TAR можна очистити установленням біта TACLR. Установлення TACLR також очищає дільник тактової частоти і напрямок рахунку для режиму вверх/вниз.

Таймер може бути запущений або перезагружений такими способами:

– таймер рахує, коли MCx > 0 і джерело тактових імпульсів активне;

– коли таймер у режимі «нагору» або «нагору/вниз», він може бути зупинений шляхом запису 0 в TACCR0. Потім таймер може бути перезагружений шляхом запису ненульового значення в TACCR0. У цьому випадку таймер починає інкрементуватися від нуля. Таймер має чотири режими роботи: «стоп», «нагору», «безперервний» і «нагору/вниз». Режими роботи вибираються за допомогою бітів MCx.

З 16-розрядним модулем таймера A пов'язані два вектори переривань:

1) вектор переривання TACCR0 для TACCR0 CCIFG;

2) вектор переривання TAIV для всіх інших прапорів CCIFG і TAIFG.

У режимі захоплення будь-який прапор CCIFG установлюється, коли значення таймера зафіксоване у відповідному регістрі TACCRx. У режимі порівняння встановлюється будь-який прапор CCIFG, якщо TAR дорахував до відповідного значення TACCRx. Програмне забезпечення може також установлювати або очищати будь-який прапор CCIFG. Усі прапори CCIFG запитують переривання, коли встановлені їхні відповідні біти CCIE і біт GIE.

Перелік регістрів таймера A наведений у табл. 2.24.

Таймер B – це 16-розрядний таймер/лічильник з декількома регістрами захоплення/порівняння. Таймер B3 (із трьома регістрами захоплення/порівняння) реалізований у пристроях MSP430x13x і MSP430x15x. Таймер B7 (із сімома регістрами захоплення/порівняння) реалізований у MSP430x14x і MSP430x16x [36].

Таймер B може підтримувати кілька режимів захоплення/порівняння, виведення ШІМ-сигналів і витримку тимчасових інтервалів. Таймер B також має розширені можливості переривань. Переривання можуть бути згенеровані при переповненні лічильника й від кожного з регістрів захоплення/порівняння.

Таймер B має такі можливості: асинхронний 16-розрядний таймер/лічильник із чотирма режимами роботи і чотирма, що настроюються тривалістю; обирані й конфігуровані джерела тактування; три або сім конфігурованих регістрів захоплення/порівняння; конфігуровані виходи з можливістю ШІМ; защіпки порівняння з подвійною буферизацією і що синхронізуються завантаженням.

Таблиця 2.24 – Регістри таймера А

Регістр	Коротке позначення	Тип регістра	Адреса	Вихідний стан
Керування таймером А	TACTL	Читання/запис	0160h	Скидання з POR
Лічильник таймера А	TAR	Читання/запис	0170h	Скидання з POR
Регістр 0 керування захопленням/порівнянням таймера А	TACCTL0	Читання/запис	0162h	Скидання з POR
Регістр 0 захоплення/ порівняння таймера А	TACCR0	Читання/запис	0172h	Скидання з POR
Регістр 1 керування захопленням/порівнянням таймера А	TACCTL1	Читання/запис	0164h	Скидання з POR
Регістр 1 захоплення/ порівняння таймера А	TACCR1	Читання/запис	0174h	Скидання з POR
Регістр 2 керування захопленням/порівнянням таймера А	TACCTL2	Читання/запис	0166h	Скидання з POR
Регістр 2 захоплення/ порівняння таймера А	TACCR2	Читаня/запис	0176h	Скидання з POR
Вектор переривання таймера А	TAIV	Тільки читання	012Eh	Скидання з POR

Регістр вектора переривання для швидкого декодування всіх переривань таймера В. Таймер В ідентичний таймеру А, але з такими виключеннями:

– довжина таймера В програмується й може становити 8, 10, 12 або 16 біт;

– регістри TBCCRx таймера В мають подвійну буферизацію і можуть групуватися;

– усі виходи таймера В можуть бути переведені в третій стан;

– функція біта SCCI не реалізована в таймері В.

Модуль таймера В конфігурується програмним забезпеченням користувача.

– 16-розрядний регістр таймера/лічильника TBR інкрементується і декрементується (залежно від режиму роботи) за кожним зростаючим фронтом тактового сигналу. Регістр TBR може програмно читатися і за-

писуватися. Крім цього, таймер може генерувати переривання при його переповненні.

Регістр TBR може бути очищений установленням біта TBCLR. Установка TBCLR також очищає дільник тактової частоти і напрямок рахунку для режиму «нагору/вниз».

Таймер може бути запущений або перезапущений такими способами:

– таймер визначає, коли MCx > 0 і активне джерело тактових сигналів;

– коли таймер перебуває в режимі рахунку «нагору» або «нагору/вниз», його можна зупинити завантаженням нуля в TBCL0. Таймер може бути тоді й перезапущений при завантаженні ненульового значення в TBCL0. У цьому випадку таймер починає інкрементування нагору від нуля.

Таймер має чотири режими роботи: «стоп», «нагору», «безперервний» і «нагору/вниз». Робочий режим вибирається за допомогою бітів MCx.

З 16-розрядним модулем таймера B зв’язані два вектори переривань:

1) вектор переривання TBCCR0 для TBCCR0 CCIFG;

2) вектор переривання TBIV для всіх інших прапорів CCIFG і TBIFG.

Перелік регістрів таймера B наведений у табл. 2.25.

Таблиця 2.25 – Регістри таймера B

Регістр	Коротке позначення	Тип регістра	Адреса	Вихідний стан
Керування таймером B	TBCTL	Читання/запис	0180h	Скидання з POR
Лічильник таймера B	TBR	Читання/запис	0190h	Скидання з POR
Регістр 0 керування захопленням/ порівнянням таймера B	TBCCTL0	Читання/запис	0182h	Скидання з POR
Регістр 0 захоплення/ порівняння таймера B	TBCCR0	Читання/запис	0192h	Скидання з POR
Регістр 1 керування захопленням/порівнянням таймера B	TBCCTL1	Читання/запис	0184h	Скидання з POR

Закінчення табл. 2.25

Регістр	Коротке позначення	Тип регістра	Адреса	Вихідний стан
Регістр 1 захоплення/ порівняння таймера B	TBCCR1	Читання/запис	0194h	Скидання з POR
Регістр 2 керування захопленням/порівнянням таймера B	TBCCTL2	Читання/запис	0186h	Скидання з POR
Регістр 2 захоплення/ порівняння таймера B	TBCCR2	Читання/запис	0196h	Скидання з POR
Регістр 3 керування захопленням/порівнянням таймера B	TBCCTL3	Читання/запис	0188h	Скидання з POR
Регістр 3 захоплення/ порівняння таймера B	TBCCR3	Читання/запис	0198h	Скидання з POR
Регістр 4 керування захопленням/порівнянням таймера B	TBCCTL4	Читання/запис	018Ah	Скидання з POR
Регістр 4 захоплення/ порівняння таймера B	TBCCR4	Читання/запис	019Ah	Скидання з POR
Регістр 5 керування захопленням/порівнянням таймера B	TBCCTL5	Читання/запис	018Ch	Скидання з POR
Регістр 5 захоплення/ порівняння таймера B	TBCCR5	Читання/запис	019Ch	Скидання з POR
Регістр 6 керування захопленням/порівнянням таймера B	TBCCTL6	Читання/запис	018Eh	Скидання з POR
Регістр 6 захоплення/ порівняння таймера B	TBCCR6	Читання/запис	019Eh	Скидання з POR
Вектор переривання таймера B	TBIV	Тільки читання	011Eh	Скидання з POR

У режимі захоплення будь-який прапор CCIFG установлюється, коли значення таймера зафіксоване у відповідному регістрі TBCCRx. У режимі порівняння встановлюється будь-який прапор CCIFG, якщо TBR дорахував до відповідного значення TBCLx. Програмне забезпечення може також установлювати або очищати будь-який прапор CCIFG. Усі прапори CCIFG запитують переривання, коли встановлені їхні відповідні біти CCIE і біт GIE.

2.8.8. Периферійний інтерфейс USART, режим UART

Універсальний синхронно/асинхронний прийомопередаючий (USART) периферійний інтерфейс підтримує два послідовних режими в одному апа-ратному модулі. USART0 реалізований у пристроях MSP430x12xx, MSP430x13xx і MSP430x15x. На додаток до USART0 у пристроях MSP430x14x і MSP430x16x реалізований другий ідентичний USART модуль – USART1.

В асинхронному режимі USART підключає MSP430 до зовнішньої системи через два зовнішніх виводи: URXD і UTXD.

Режим UART вибирається при очищенні біта SYNC.

Режим UART має такі можливості: 7- або 8-розрядні дані з перевіркою парності/непарності і без контролю парності; незалежні здвигові регістри передачі й прийому; роздільні буферні регістри передачі і прийому; передача і прийом починаються з молодшого біта даних; вбудовані комунікаційні протоколи вільної лінії й адресного біта для багатопроцесорних систем; визначення в приймачі стартового фронту сигналу для автоматичного пробудження з режимів LPMx; програмувальна швидкість передачі з модуляцією для підтримки дробових величин швидкостей; прапори статусу для виявлення помилок, блокування і визначення адреси; можливі незалежні переривання для прийому і передачі.

У режимі UART модуль USART передає й приймає символи на швидкості, асинхронній іншому пристрою. Синхронізація кожного символу заснована на обраній швидкості передачі USART. Для виконання функцій передачі й прийому використовується однакова швидкість у бодах.

USART має один вектор переривання для передачі і один вектор переривання для прийому.

Прапор переривання UTXIFGx установлюється передавачем для індикації готовності UxTXBUF до прийому іншого символу. Запит переривання генерується, якщо встановлені прапори UTXIEx і GIE. UTXIFGx автоматично скидається, якщо запит переривання обслужений або якщо символ записаний в UxTXBUF.

UTXIFGx установлюється після PUC або коли SWRST = 1. UTXIEx скидається після PUC або коли SWRST = 1.

Прапор переривання URXIFGx установлюється щоразу при прийманні символу і його завантаженні в UxRXBUF. Запит переривання гене-

рується, якщо також установлені прапори URXIEx і GIE. URXIFGx і URXIEx скидаються сигналом системного скидання PUC або коли SWRST = 1. URXIFGx скидається автоматично, якщо запит переривання оброблений (коли URXSE = 0) або коли прочитаний UxRXBUF.

URXEIE використовується для дозволу або заборони установлення URXIFGx від помилкових символів. У багатопроцесорному адресному режимі URXWIE використовується для автоматичного виявлення правильних символів адреси й відхилення небажаних символів даних.

Два типи символів не встановлюють URXIFGx:

- помилкові символи при URXEIE = 0;
- символи, що не є адресою при URXWIE = 1;
- коли URXEIE = 1, стан розриву встановить біт BRK і прапор URXIFGx.

Універсальний синхронно/асинхронний прийомопередаючий (USART) периферійний інтерфейс підтримує два послідовних режими в одному апаратному модулі інтерфейсу. USART0 реалізований у пристроях MSP430x12xx, MSP430x13xx і MSP430x15x. На додаток до USART0, у пристроях MSP430x14x і MSP430x16x реалізований другий ідентичний USART модуль-USART1. У синхронному режимі USART підключає MSP430 до зовнішньої системи через три або чотири виведення: SIMO, SOMI, UCLK і STE.

Режим SPI вибирається, коли біт SYNC установлений, а біт I2C очищений. Він має такі можливості: 7-ми або 8-розрядні дані; робота SPI з 3-ма або 4-ма виведеннями; режими ведучий або ведений; незалежні зсувні регістри передачі і прийому; роздільні буферні регістри передачі й прийому; обирана полярність UCLK і керування фазою; програмувальна частота UCLK у режимі ведучого; незалежна можливість переривання для прийому і передачі.

У синхронному режимі послідовні дані передаються і приймаються безліччю пристроїв з використанням загального тактування, забезпечуваного ведучим. Додаткове виведення STE, кероване ведучим, необхідне для дозволу прийому й передачі даних пристроєм.

USART скидається сигналом PUC або бітом у SWRST. Після PUC біт SWRST автоматично встановлюється, залишаючи USART у стані скидання. Коли він установлений, біт SWRST скидає біти URXIEx, UTXIEx, URXIFGx, OE, FE і встановлює прапор UTXIFGx. Біт USPIEx не змінюється бітом у SWRST. Для роботи USART необхідно очистити SWRST.

2.8.9. Компаратор А

Компаратор А – це аналоговий компаратор напруги. Компаратор А реалізований у пристроях MSP430x11x1, MSP430x12x, MSP430x13x, MSP430x14x, MSP430x15x і MSP430x16x.

Модуль компаратора А підтримує високоточні аналого-цифрові перетворення напруги, контроль напруги живлення й моніторинг зовнішніх аналогових сигналів. Блок-схема компаратора А показана на рис. 2.49.

Компаратор А має такі можливості: інвертує і не інвертує виводи вхідного мультиплексора; RC-фільтр на виході компаратора; підключення виходу до входу захоплення таймера А; програмне керування вхідним буфером порту; можливість переривання; генератор опорної напруги, що настроюється; можливість вимикання компаратора і опорного генератора.

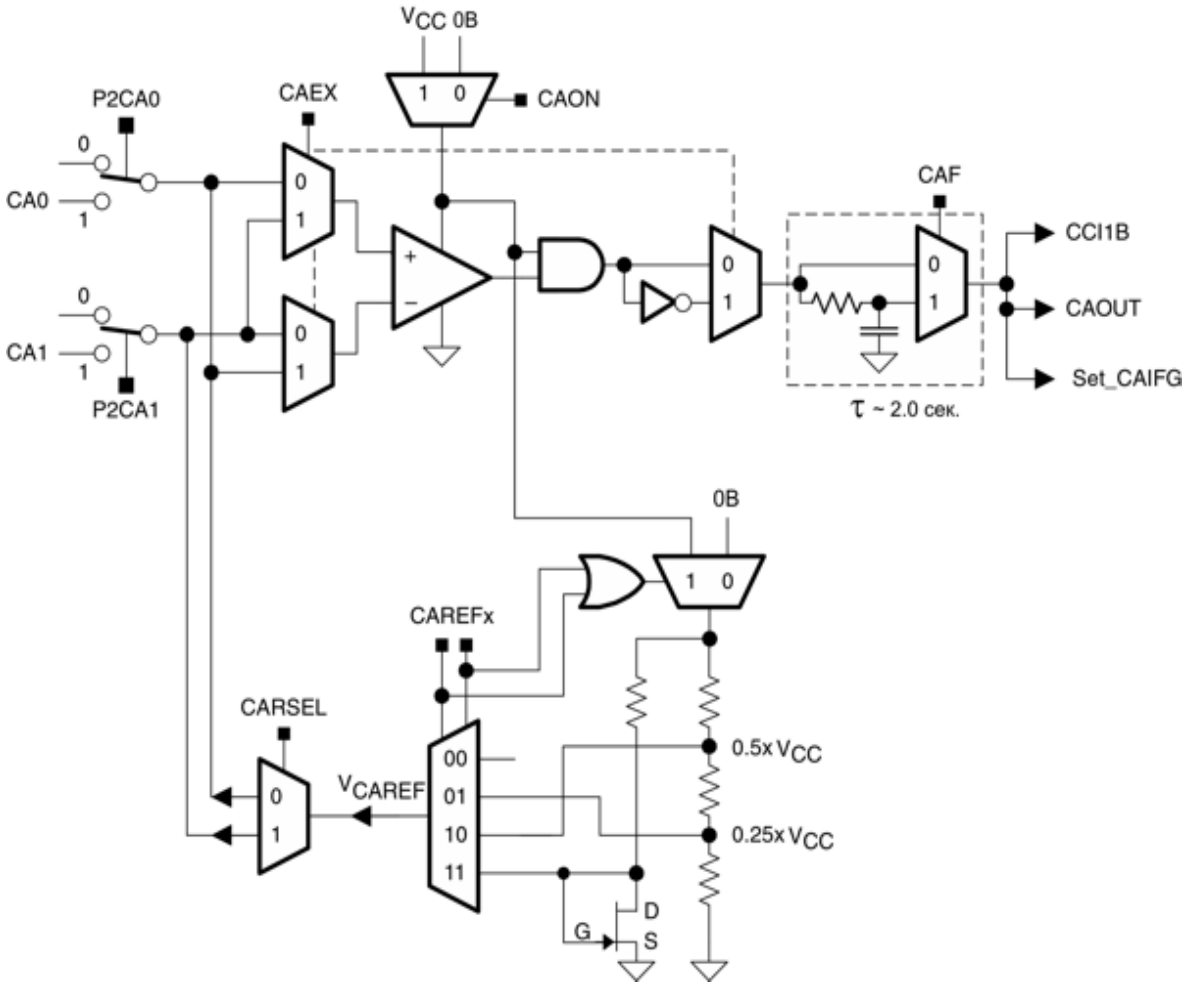


Рисунок 2.49 – Блок-схема компаратора А

Модуль компаратора А конфігурується програмним забезпеченням користувача. Компаратор зрівнює аналогові напруги на входах «+» і «–». Якщо вхід «+» більше позитивний, чим вхід «–», на виході компаратора CAOUT з'являється сигнал високого рівня. Компаратор може бути включений або виключений за допомогою керуючого біта CAON. Якщо компаратор не використовується, для зменшення споживаного струму, його необхідно виключати. Коли компаратор виключений, на виході CAOUT завжди сигнал низького рівня. Регістри компаратора А наведені в табл. 2.26.

Таблиця 2.26 – Регістри компаратора А

Регістр	Коротке позначення	Тип регістра	Адреса	Вихідний стан
Керуючий регістр 1 компаратора А	CACTL1	Читання/запис	059h	Скидання з POR
Керуючий регістр 2 компаратора А	CACTL2	Читання/запис	05Ah	Скидання з POR
Відключення порту компаратора А	CAPD	Читання/запис	05Bh	Скидання з POR

2.8.10. Модуль аналого-цифрового перетворювача АЦП12

Модуль АЦП12 являє собою високоефективний 12-розрядний аналого-цифровий перетворювач. АЦП12 реалізований у пристроях MSP430x13x, MSP430x14x, MSP430x15x і MSP430x16x [36].

Модуль АЦП12 забезпечує швидкі 12-розрядні аналого-цифрові перетворення. Модуль має 12-розрядне ядро SAR, схему вибірки, опорний генератор і буфер перетворення й керування обсягом 16 слів. Буфер перетворення й керування дозволяє одержувати й зберігати до 16 незалежних вибірок АЦП без втручання ЦПУ. АЦП12 має такі можливості: максимальна швидкість перетворення понад 200 ksps; монотонний 12-розрядний перетворювач без кодів помилок; вибірка і зберігання із програмувальними періодами вибірки, обумовленими програмним забезпеченням або таймерами; перетворення ініціюється програмним забезпеченням, таймером А або таймером В; програмно-обираний інтегрований генератор опорної напруги (1,5 В або 2,5 В); програмно-обиране внутрішнє або зовнішнє опорне джерело; вісім індивідуально

конфігурованих зовнішніх вхідних каналів; канали перетворення для внутрішнього температурного датчика, AVCC і зовнішніх опорних джерел; незалежні опорні джерела, що задаються шляхом вибору каналу, для обох позитивних і негативних опорних джерел; обиране джерело тактованих перетворень; одноканальний, повторний одноканальний, послідовний і повторно-послідовний режими перетворення; ядро АЦП і опорна напруга можуть вимикатися роздільно; регістр вектора переривань для швидкого декодування 18 переривань АЦП; 16 регістрів зберігання результату.

Блок-схема АЦП12 показана на рис. 2.50.

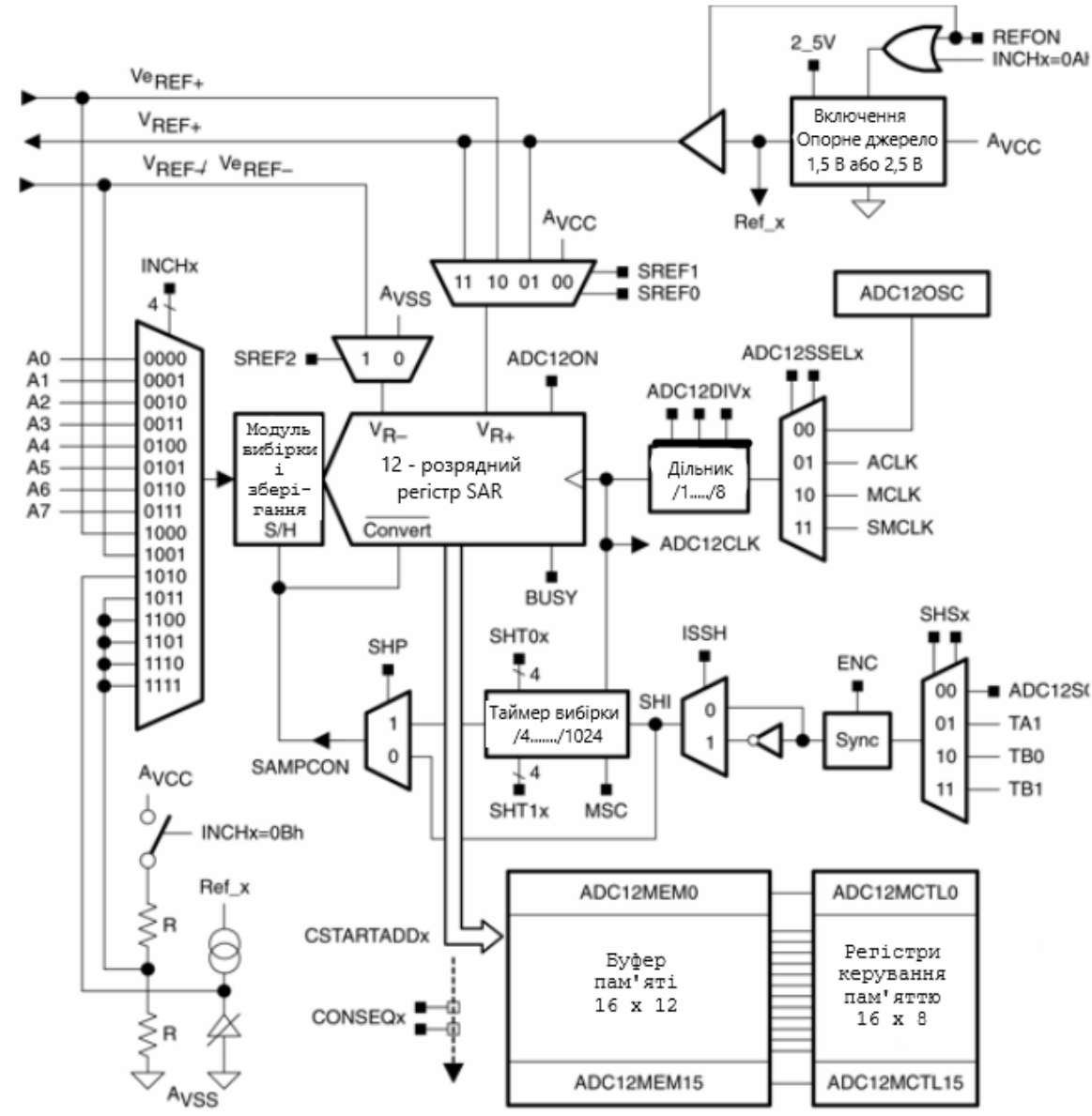


Рисунок 2.50 – Блок-схема АЦП12

Модуль АЦП12 конфігурується програмним забезпеченням користувача. Налагодження й робота АЦП12 розглядаються в наступних розділах.

Ядро АЦП перетворить аналоговий вхідний сигнал у 12-розрядне цифрове подання й зберігає результат у пам'яті перетворень. Ядро використовує два програмувальних/обираних рівні напруги (VR+ і VR–) для завдання верхньої й нижньої меж перетворення. На цифровому виході (NADC) представлена повна шкала (0FFFh), коли вхідний сигнал дорівнює або вище VR+, і нуль, коли вхідний сигнал дорівнює або нижче VR–. Вхідний канал і опорні рівні напруги (VR+ і VR–) задаються в пам'яті керування перетвореннями. Формула перетворення для результату АЦП NADC виглядає так:

$$N_{ADC} = 4095 \cdot \frac{Vin - V_{R-}}{V_{R+} - V_{R-}}.$$

Ядро АЦП12 конфігурується двома керуючими регістрами: ADC12CTL0 і ADC12CTL1. Ядро включається бітом у ADC12ON. Якщо ADC12 не використовується то для збереження енергії, воно може бути виключене. За деякими винятками біти керування АЦП12 можуть бути модифіковані, тільки коли ENC = 0. ENC повинен бути встановлений в 1 перед виконанням будь-якого перетворення.

Вісім зовнішніх і чотири внутрішніх аналогових сигнали вибираються як канал для перетворення аналоговим вхідним мультиплексором, який зображений на рис. 2.51. Вхідний мультиплексор має тип break-before-make (розрив перед включенням), що зменшує інжекцію шумів від каналу до каналу, що виникає при перемиканні каналів. Вхідний мультиплексор також є

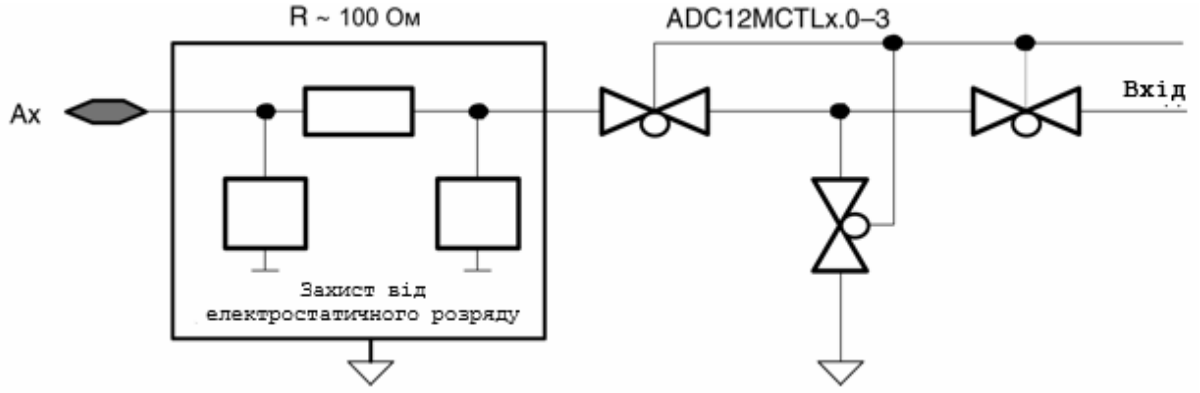


Рисунок 2.51 – Аналоговий мультиплексор

Т-перемикачем, що зменшує взаємозв'язок між каналами. Невибрані канали ізольовані від АЦП, а проміжний вузол підключений до аналогової землі (AVSS), тому паразитна ємність заземлюється, що допомагає усувати перехресні перешкоди.

АЦП12 використовує метод перерозподілу заряду. Коли входи внутрішньо перемикаються, перемикання може призвести до перехідних процесів на вхідному сигналі. Ці перехідні процеси загасають і встановлюються до появи помилкового перетворення.

Входи АЦП12 мультиплексовані з ніжками порту P6, що мають цифрові КМОП середовища. Коли аналогові сигнали подаються до цифрових КМОП-схем, то може текти паразитний струм від VCC до GND. Цей струм з'являється, якщо величина вхідної напруги знаходиться біля перехідного рівня середовища. Відключення буфера ніжки порту усуває протікання паразитного струму й внаслідок цього зменшує загальний споживаний струм. Біти P6SELx дають можливість відключати вхідні і вихідні буфери ніжки порту.

Модуль АЦП12 містить убудований генератор опорної напруги із двома обираними рівнями напруги: 1,5 В и 2,5 В. Кожна з цих опорних напруг може бути використана внутрішньо або зовні на виводі VREF+.

Установкою REFON = 1 включається внутрішнє опорне джерело. Коли REF2_5V = 1, внутрішня опорна напруга дорівнює 2,5 В, при REF2_5V = 0 опорна напруга дорівнює 1,5 В. Якщо генератор опорної напруги не використовується, він може бути вимкнений для зменшення споживання енергії.

Для правильної роботи внутрішнього генератора опорної напруги необхідно використати ємність тимчасового зберігання енергії, підключену між VREF+ і AVSS. Рекомендується як таку ємність використати комбінацію із ввімкнених паралельно конденсаторів на 10 мкФ і 0,1 мкФ. Після ввімкнення протягом максимум 17 мс необхідно дати можливість генератору опорної напруги зарядити конденсатори зберігання енергії. Якщо внутрішній опорний генератор не використовується при перетвореннях, конденсатори не потрібні

Зовнішні опорні джерела можуть бути задіяні для VR+ і VR– через виведення VeREF + і VRED– / VeREF– відповідно.

АЦП12 має чотири режими роботи, обирані бітами CONSEQx так, як описано в табл. 2.27.

Таблиця 2.27 – Зведений перелік режимів перетворення

CONSEQx	Режим	Операція
00	Одноканальний з одиночним перетворенням	Виконується одне перетворення в одному каналі
01	Послідовність каналів	Виконуються однократні перетворення послідовності каналів
10	Повторюваний одноканальний	Виконується повторюване перетворення в одному каналі
11	Повторювана послідовність каналів	Виконуються повторювані перетворення послідовності каналів

Припинення активності АЦП12 залежить від режиму роботи. Рекомендуються такі способи зупину активного перетворення або його послідовності:

- Скидання ENC в одноканальному режимі одиночного перетворення негайно зупиняє перетворення, при цьому результат виявляється непередбаченим. Для одержання правильного результату необхідно опитувати біт зайнятості до скидання перед очищенням ENC.
- Скидання ENC під час повторюваного одноканального перетворення зупиняє перетворювач наприкінці поточного перетворення. Скидання ENC під час послідовного або повторно-послідовного режимів зупиняє перетворювач наприкінці послідовності. Будь-який режим перетворення може бути негайно зупинений установленням CONSEQx = 0 і скиданням біта ENC. Дані перетворення будуть ненадійні.

АЦП12 має 18 джерел переривання: ADC12IFG0-ADC12IFG15; ADC12OV, переповнення AD12MEMx; ADC12TOV, переповнення часу перетворення АЦП12. Біти ADC12IFGx установлюються, коли в їхні відповідні регістри пам’яті ADC12MEMx завантажується результат перетворення. Якщо відповідний біт ADC12IEx і біт GIE установлені, генерується запит переривання. Стан ADC12OV з’являється, коли результат перетворення записується в будь-який регістр ADC12MEMx до прочитання попереднього результату. Стан ADC12TOV генерується, коли до завершення поточного перетворення викликане інше вибрання-перетворення.

Регістри АЦП12 наведені в табл. 2.28.

Таблиця 2.28 – Регістри АЦП12

Регістр	Коротке позначення	Тип регістра	Адреса	Вихідний стан
Керуючий регістр 0 АЦП12	ADC12CTL0	Читання/запис	01A0h	Скидання з POR
Керуючий регістр 1 АЦП12	ADC12CTL1	Читання/запис	01A2h	Скидання з POR
Регістр прапорів переривань АЦП12	ADC12IFG	Читання/запис	01A4h	Скидання з POR
Регістр дозволу переривань АЦП12	ADC12IE	Читання/запис	01A6h	Скидання з POR
Слово вектора переривань АЦП12	ADC12IV	Читання	01A8h	Скидання з POR
Регістр пам’яті 0 АЦП12	ADC12MEM0	Читання/запис	0140h	Не змінюється
Регістр пам’яті 1 АЦП12	ADC12MEM1	Читання/запис	0142h	Не змінюється
Регістр пам’яті 2 АЦП12	ADC12MEM2	Читання/запис	0144h	Не змінюється
Регістр пам’яті 3 АЦП12	ADC12MEM3	Читання/запис	0146h	Не змінюється
Регістр пам’яті 4 АЦП12	ADC12MEM4	Читання/запис	0148h	Не змінюється
Регістр пам’яті 5 АЦП12	ADC12MEM5	Читання/запис	014Ah	Не змінюється
Регістр пам’яті 6 АЦП12	ADC12MEM6	Читання/запис	014Ch	Не змінюється
Регістр пам’яті 7 АЦП12	ADC12MEM7	Читання/запис	014Eh	Не змінюється
Регістр пам’яті 8 АЦП12	ADC12MEM8	Читання/запис	0150h	Не змінюється
Регістр пам’яті 9 АЦП12	ADC12MEM9	Читання/запис	0152h	Не змінюється
Регістр пам’яті 10 АЦП12	ADC12MEM10	Читання/запис	0154h	Не змінюється
Регістр памя’яті 11 АЦП12	ADC12MEM11	Читання/запис	0156h	Не змінюється
Регістр пам’яті 12 АЦП12	ADC12MEM12	Читання/запис	0158h	Не змінюється
Регістр пам’яті 13 АЦП12	ADC12MEM13	Читання/запис	015Ah	Не змінюється

Закінчення табл. 2.28 – Регістри АЦП12

Регістр	Коротке позначення	Тип регістра	Адреса	Вихідний стан
Регістр пам’яті 14 АЦП12	ADC12MEM14	Читання/запис	015Ch	Не з мінюється
Регістр пам’яті 15 АЦП12	ADC12MEM15	Читання/запис	015Eh	Не змінюється
Керування регістром пам’яті 0 АЦП12	ADC12MCTL0	Читання/запис	080h	Скидання з POR
Керування регістром пам’яті 1 АЦП12	ADC12MCTL1	Читання/запис	081h	Скидання з POR
Керування регістром пам’яті 2 АЦП12	ADC12MCTL2	Читання/запис	082h	Скидання з POR
Керування регістром пам’яті 3 АЦП12	ADC12MCTL3	Читання/запис	083h	Скидання з POR
Керування регістром пам’яті 4 АЦП12	ADC12MCTL4	Читання/запис	084h	Скидання з POR
Керування регістром пам’яті 5 АЦП12	ADC12MCTL5	Читання/запис	085h	Скидання з POR
Керування регістром пам’яті 6 АЦП12	ADC12MCTL6	Читання/запис	086h	Скидання з POR
Керування регістром пам’яті 7 АЦП12	ADC12MCTL7	Читання/запис	087h	Скидання з POR
Керування регістром пам’яті 8 АЦП12	ADC12MCTL8	Читання/запис	088h	Скидання з POR
Керування регістром пам’яті 9 АЦП12	ADC12MCTL9	Читання/запис	089h	Скидання з POR
Керування регістром пам’яті 10 АЦП12	ADC12MCTL10	Читання/запис	08Ah	Скидання з POR
Керування регістром пам’яті 11 АЦП12	ADC12MCTL11	Читання/запис	08Bh	Скидання з POR
Керування регістром пам’яті 12 АЦП12	ADC12MCTL12	Читання/запис	08Ch	Скидання з POR
Керування регістром пам’яті 13 АЦП12	ADC12MCTL13	Читання/запис	08Dh	Скидання з POR
Керування регістром пам’яті 14 АЦП12	ADC12MCTL14	Читання/запис	08Eh	Скидання з POR
Керування регістром пам’яті 15 АЦП12	ADC12MCTL15	Читання/запис	08Fh	Скидання з POR

2.8.11. Модуль цифро-аналогового перетворювача ЦАП12

Модуль ЦАП12 являє собою 12-розрядний цифро-аналоговий перетворювач. У пристроях MSP430x15x і MSP430x16x реалізовано два модулі ЦАП12. Модуль АЦП12 являє собою 12-розрядний ЦАП на основі матриці резисторів R-2R. ЦАП12 може бути сконфігурований у 8-ми або 12-розрядному режимі й може використатися разом з контролером DMA. Коли в пристрої є кілька модулів ЦАП12, вони можуть бути згруповані разом для синхронного відновлення [36].

ЦАП12 має такі можливості: 12-розрядний монотонний вихід; 8-ми або 12-розрядний дозвіл вихідної напруги; програмувальний час установлення залежно від споживаної потужності; вибір внутрішнього або зовнішнього опорного джерела; натуральний двійковий формат даних або формат з доповненням до двох; опція самокалібрування для корегування зсуву; можливість синхронного відновлення при наявності декількох модулів ЦАП12; кілька модулів ЦАП12 можуть бути згруповані для одночасного відновлення.

Блок-схема двох модулів ЦАП12 у пристроях MSP430F15x/16x показана на рис. 2.52.

Модуль ЦАП12 конфігурується програмним забезпеченням користувача. ЦАП12 може бути сконфігурований на роботу у 8-ми або 12-розрядному режимі за допомогою біта DASC12RES. Крім того, повний діапазон виведення програмується через біт DAC12IR і може бути 1- або 3-кратний обраній опорній напрузі. Ця можливість дозволяє користувачеві керувати динамічним діапазоном ЦАП12. Коли використається внутрішнє опорне джерело, повний діапазон виведення завжди дорівнює опорній напрузі. Біт DAC12DF дозволяє користувачеві вибирати для ЦАП натуральні двійкові дані або дані з доповненням до двох. Коли використається натуральний двійковий формат даних, справедлива формула для вихідної напруги подана в табл. 2.29.

У 8-розрядному режимі максимальне використовуване значення для DAC12_xDAT дорівнює 0FFh, а в 12-розрядному режимі максимальне використовуване значення для DAC12_xDAT дорівнює 0FFFh. Значення, що перевищують зазначені величини, можуть бути записані в регістр, але всі перші біти будуть знехтувані.

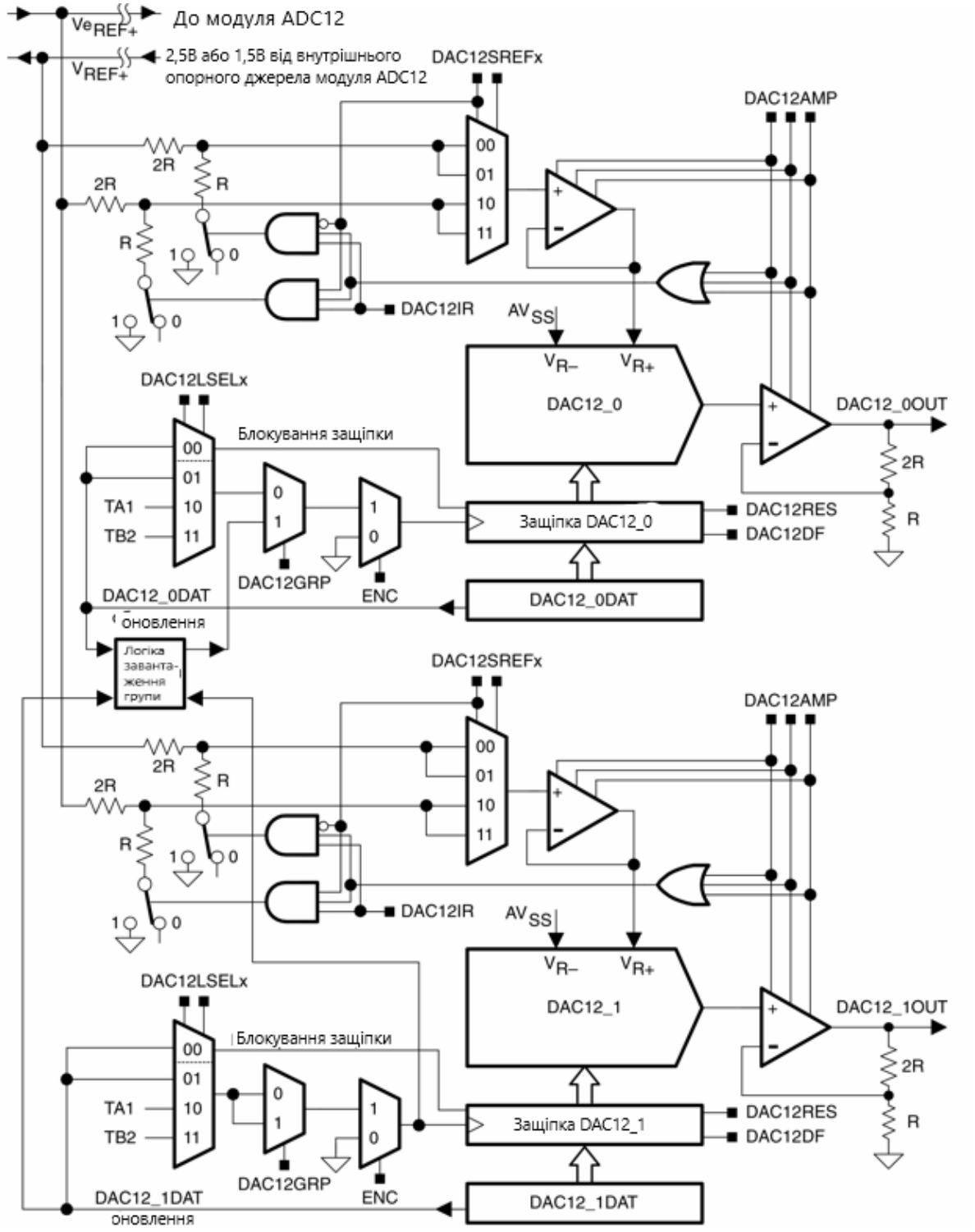


Рисунок 2.52 – Блок-схема ЦАП12

Таблиця 2.29 – Повний діапазон ЦАП12

Дозвіл	DAC12RES	DAC12IR	Формула вихідної напруги
12 біт	0	0	$V_{out} = V_{ref} \times 3 \times \frac{\text{DAC12} \times \text{DAT}}{4096}$
12 біт	0	1	$V_{out} = V_{ref} \times \frac{\text{DAC12} \times \text{DAT}}{4096}$
8 біт	1	0	$V_{out} = V_{ref} \times 3 \times \frac{\text{DAC12} \times \text{DAT}}{256}$
8 біт	1	1	$V_{out} = V_{ref} \times \frac{\text{DAC12} \times \text{DAT}}{256}$

Виходи ЦАП12 мультиплексовані з ніжками порту P6 і аналоговими входами АЦП12. Коли DAC12AMPx > 0, для ніжок автоматично вибирається функція ЦАП12, незалежно від стану пов'язаних з ними бітів P6SELx і P6DIRx. *Опорне джерело* для ЦАП12 конфігурується для використання або двох зовнішніх опорних напруг, або внутрішнього опорного джерела 1,5В/2,5В від модуля АЦП12 за допомогою бітів DAC12SREFx. Коли DAC12SREFx = {0,1}, як опорний використовується сигнал VREF+, а коли DAC12SREFx = {2,3}, як опорний використовується сигнал VeREF+. При використанні внутрішнього опорного джерела АЦП12 він повинен бути включений і сконфігурований через відповідні керуючі біти АЦП12 (див. п. 2.8.10). Як тільки опорне джерело АЦП12 сконфігуроване, опорна напруга подається на VREF+.

Буфери вхідного опорного сигналу й вихідної напруги ЦАП12 можуть бути сконфігуровані для оптимізації часу установлення 1 залежно від споживаної потужності. Вісім можливих комбінацій вибираються за допомогою бітів DAC12AMPx. При низькому настроюванні час установлення найбільший, а споживаний обома буферами струм найменший. Середні й високі настроювання дозволяють одержати менший час установлення, однак споживаний струм зросте (див. довідкове керівництво конкретного пристрою для з'ясування докладних параметрів).

Вектор переривань ЦАП12 є загальним з контролером DMA. Програмне забезпечення повинне перевіряти прапори DAC12IFG і DMAIFG для визначення джерела переривання. Біт DAC12IFG установлюється,

коли DAC12xLSELx > 0 і дані ЦАП12 увімкнуті від регістра DAC12_xDAT у защіпці даних. Коли DAC12xLSELx = 0, прапор DAC12IFG не встановлюється. Установлений біт DAC12IFG показує, що ЦАП12 готовий для прийому нових даних. Якщо встановлені обидва біти, DAC12IE і GIE, DAC12IFG, генерує запит переривання. Прапор DAC12IFG не скидається автоматично, а повинне скидатись програмним забезпеченням.

Регістри ЦАП12 наведені в табл. 2.30.

Таблиця 2.30 – Регістри ЦАП12

Регістр	Коротке позначення	Тип регістра	Адреса	Вихідний стан
Керування DAC12_0	DAC12_0CTL	Читання/запис	01C0h	Скидання з POR
Дані DAC12_0	DAC12_0DAT	Читання/запис	01C8h	Скидання з POR
Керування DAC12_1	DAC12_1CTL	Читання/запис	01C2h	Скидання з POR
Дані DAC12_1	DAC12_1DAT	Читання/запис	01CAh	Скидання з POR

Приклад програми керування МК MSP430 тепловим захистом АВ мовою Асемблер наведено нижче [44].

```
            CMP   #0x001a,R7
            JZ    Outnimb4
            CMP   #0x001c,R7
            JZ    Outnimb5
            ADD   #M0,R7
            mov   @R7,R15
ContOut:    INV   R15
          mov.B R15,&P5OUT
          CMP Ir85,Imaxper2
          JL  NoInd
          CMP Ir105,Imaxper2
          Jl  Ind
          BIT  #0x0100,R12
          JZ  NoInd
```

```
Ind:    BIC.B #BIT4,&P5OUT
        JMP TstProg
Outnimb1: Mov Ihold,R15
          SWPB  R15
          RRA   R15
          RRA   R15
          RRA   R15
          RRA   R15
          JMP  ContOut
Outnimb2: Mov Ihold,R15
          SWPB  R15
          JMP  ContOut
Outnimb3: Mov Ihold,R15
          RRA   R15
          RRA   R15
          RRA   R15
          RRA   R15
          JMP  ContOut
Outnimb4: Mov Ihold,R15
          JMP  ContOut
Outnimb5: Mov Shold,R15
          JMP  ContOut
NoInd:  BIS.B #BIT4,&P5OUT
//--перевірка режиму програмування параметра
    (натиснення 4 сек кнопки «Програмування»)
TstProg:  BIT.B #BIT0,&P2IN
     JZ    Tst4sec  //кнопка натиснута
          Mov   #0,Time4sec
          JMP   TeploZ
Tst4sec:  CMP   #4000,Time4sec
          JGE   IndWr
          INC   Time4sec
//--------робота теплового захисту 1 раз в 20 мсек--
TeploZ: CMP     Imax,Imaxpernew
        JGE     NoMore
        MOV     Imax,Imaxpernew
```

```
NoMore: CMP     #20,R13
        JLO     WaitStart
        CLR     R13
        MOV     Imaxpernew,Imaxper
        CLR     Imaxpernew
//*****************
        MOV     Imaxper,R15
        SUB     Imaxper2,R15
        RRA     R15
        RRA     R15
        RRA     R15
        RRA     R15
        ADD     R15,Imaxper2
//******************
//        JMP     WaitStart ////
TZ:     CMP     Irst,Imaxper
        JL      NoWorkR
//-------робота теплового захисту---
        BIT     #Rstart,Status
        JNZ     ContRstart
        BIS     #Rstart,Status
        BIC     #Norm,Status
        MOV     Trres,Tres
        MOV     I2TrLow,TimI2TrLow
        MOV     I2TrHigh,TimI2TrHigh
ContRstart: MOV Imaxper,R7
        RRA     R7
        RRA     R7
        RRA     R7
        RRA     R7
        dint
        MOV     R7,&MPYS
    MOV     R7,&OP2
        nop
        SUB     &RESLO,TimI2TrLow
        SUBC    &RESHI,TimI2TrHigh
        eint
        JGE     WorkR
```

```
SetRtrim: BIS   #Rtrim,Status
        BIC     #Norm+Rstart,Status
        BIC.B   #BIT0,&P4OUT  //Swith OFF
NoWorkR: BIT    #Rstart,Status
        JZ      WorkR
        DEC     Tsdres
        JGE     WorkR
        BIC     #Rstart,Status
        BIS     #Norm,Status
        DEC     Tres
        JGE     WorkR
        BIC     #Rstart,Status
        BIS     #Norm,Status
WorkR:  JMP     WaitStart
//************************************************
//запис нових значень уставок у флеш-пам'ять
// Частота флеш-генератора
   f(FCT)=MCLK/20=8MHz/20=400kHz
IndWr:   Mov.B &P1IN,R7
        INV   R7
        RRA   R7
        RRA   R7
        RRA   R7
        AND   #0x001e,R7
        ADD   #RM0,R7
        mov.B &P2IN,R15
        INV   R15
        RRA   R15
        RRA   R15
        RRA   R15
        RRA   R15
        AND   #0x000f,R15
        mov   R15,0(R7)
        MOV     #FWKEY+FSSEL_1+FN4+FN1+FN0,&FCTL2
//-Стиранння блока налагоджень і запис нових налаго-
   джень у флеш-пам'ять
                dint            //Заборона переривань
        clr.b   &IE1     //Заборона немаскованих
                                переривань
        Mov     #5A80h,&WDTCTL          //Зупинка WDT
        mov     #FWKEY,&FCTL3
```

```
Test_Busy2  BIT     #BUSY,&FCTL3
            JNZ     Test_Busy2
        mov     #FWKEY+erase,&FCTL1
        mov     #0000h,&1000h   //Стирання сегмента
                                  з налагодженнями
        mov     #FWKEY,&FCTL1
        mov     #FWKEY+lock,&FCTL3
Test_Busy1  BIT #BUSY,&FCTL3
            JNZ Test_Busy1
//---Запис нових налагоджень у флеш-пам'ять
            mov #RM0,R7     //Початкова адреса нових
                              налагоджень
WRT1        mov #FWKEY,&FCTL3
Test_Busy3  BIT #BUSY,&FCTL3
            JNZ Test_Busy3
            mov #FWKEY+wrt,&FCTL1
            mov @R7+,M0-RM0-2(R7)   //Запис слова у
                                      флеш-пам'ять
            mov #FWKEY,&FCTL1
            mov #FWKEY+lock,&FCTL3
Test_Busy4  BIT #BUSY,&FCTL3
            JNZ Test_Busy4
            CMP #RM15+2,R7   //Запис всіх налагоджень
                               закінчено?
            JNZ WRT1
        BR      RESET   //Після закінчення запису -
                          перевантаження процесора
//---------------------
#define  Const1   1481//1520//1900//2183// 2347 //1565
//номінальний струм - 400А
//#define    Const2   17*Const1/20 //85% номінального
струму
//#define    Const3   21*Const1/20 //105%
номінального струму
Inorm dw Const1
//Inorm85 dw Const2
//Inorm105 dw Const3
//---------------------
Kg1 DW  40  //
Kg2 DW  50  //
Kg3 DW  60  //
```

```
Kg4 DW  70  //
Kg5 DW  80  //
Kg6 DW  90  //
Kg7 DW  100  //
//---------------------
Tr1 DW  2
Tr2 DW  4
Tr3 DW  8
Tr4 DW  12
Tr5 DW  16
//---------------------
Ksd1 DW  3  //  /2
Ksd2 DW  4  //  /2
Ksd3 DW  6  //  /2
Ksd4 DW  8  //  /2
Ksd5 DW  10  //  /2
Ksd6 DW  12  //  /2
Ksd7 DW  16  //  /2
Ksd8 DW  20  //  /2
//---------------------
Tsd1 DW  0   // ms
Tsd2 DW  100  // ms
Tsd3 DW  200  // ms
Tsd4 DW  300  // ms
//---------------------
Ki1 DW  3  //  /2
Ki2 DW  4  //  /2
Ki3 DW  6  //  /2
Ki4 DW  8  //  /2
Ki5 DW  10  //  /2
Ki6 DW  12  //  /2
Ki7 DW  16  //  /2
Ki8 DW  20  //  /2
Ki9 DW  22  //  /2
//---------------------
Ti1 DW  0   // ms
Ti2 DW  10  // ms
Ti3 DW  20  // ms
#include "TimA0.CPP"
//#include "Div32_16.CPP"
//#include "BinDec.CPP"
#include "IntVect.C"
```

Контрольні запитання і завдання

1. Призначення, схема і принцип дії однокристальних МК.
2. Розкажіть про призначення і роботу основних елементів МК51.
3. Пристрій і режими роботи портів введення-виведення МК51.
4. Призначення, будова і режими роботи послідовного порту УАПП МК51.
5. Назвіть найбільш важливі особливості архітектури ОМК.
6. Де зберігаються програма і дані в МК51?
7. Викладіть основні функції, що виконуються АЛП в МК51.
8. Регістри спеціальних функцій МК51, їх структура і призначення.
9. Призначення, будова і режими роботи таймера/лічильника МК51 і порядок його програмування.
10. Особливості роботи МК51 в режимах скидання, покрокового виконання команд, холостого ходу і зниженого енергоспоживання.
11. Призначення і функціонування пристрою керування і синхронізації.
12. Викладіть загальну характеристику системи команд МК51.
13. Наведіть команди передачі даних і приклади їх застосування.
14. Команди арифметичних і логічних операцій та приклади їх застосування.
15. Назвіть команди операцій з бітами і наведіть приклади їх застосування.
16. Наведіть команди передачі керування і приклади їх застосування.
17. Наведіть приклади запису програм на мові Асемблер МК51.
18. Призначення і функціонування налагоджувача програм FD-51.
19. Призначення асемблюючої програми ASM-51.
20. Назвіть основні етапи і порядок налагодження програм.
21. Склад програмного забезпечення крос-засобів налагодження програм.
22. Особливості архітектури МК фірми Atmel серій AT89 і AT90.
23. Архітектура МК фірми Microchip сімейств PIC16/17 і PIC16X7XX.
23. Особливості архітектури і застосування МК фірми Motorola.
25. Структура МК сімейства HC05 і його основні характеристики.
26. Назвіть сфери застосування МК фірми Atmel.

27. Назвіть сфери застосування МК фірми Microchip.

28. Назвіть сфери застосування МК фірми Motorola.

29. Наведіть блок-схему МК сімейства HCO5.

30. Призначення і будова послідовного асинхронного інтерфейсу зв'язку SCI.

31. Особливості роботи і призначення багатофункціонального таймера MFT.

32. Призначення системи переривань реального часу RTI.

33. Наведіть блок-схему МК MC68HC705MC4 і назвіть сфери його застосування.

34. Наведіть приклади можливого застосування МК AT89 і AT90 в електроапаратобудуванні (ЕА) і електропобутовій техніці (ЕПТ).

35. Наведіть приклади застосування МК HC05 для захисту в ЕА і ЕПТ.

36. Назвіть сфери застосування 16-розрядних МК HC16 і 32-розрядних МК68300.

37. Наведіть приклади застосування МК PIC16 у ЕПТ і ЕА .

38. Призначення та будова сторожового таймера і температурного сенсора в МК.

39. Охарактеризуйте систему команд МК PIC16/17 і наведіть приклади застосування команд у керуванні ЕА і ЕПТ.

40. Викладіть особливості конвеєрної системи оброблення команд у мікроконтролерах PIC16/17.

41. Призначення і будова високошвидкісного синхронного послідовного інтерфейсу SPI.

42. Призначення і функціювання системи переривань реального часу RTI (Real Time Interrupt).

43. Призначення і функціювання підсистеми таймера вхідної фіксації IC і вихідного порівняння OC.

44. Призначення і характеристики широтно-імпульсного модулятора.

45. Назвіть серії мікроконтролерів сімейства HC05, призначених:

– для керування електродвигунами в ЕПТ (холодильниках, пральних машинах, мікрохвильових печах, кухонних комбайнах і т. ін.);

– у системах керування автомобіля (приладова панель, склопідйомники, підвіска і т. ін.).

– у різноманітних побутових пристроях, пультах керування.

– у відео- і телевізійній апаратурі для керування кольоровими екранними дисплеями, яскравістю зображення, гучністю звуку, вибором певного каналу і т. ін.

46. Назвіть сферу застосування та особливості архітектури МК ATmega16.

47. Наведіть особливості розробки та налагодження програм в інтегральному середовищі Bascom AVR.

48. Наведіть структурну схему, особливості архітектури і системи команд МК MSP430.

49. Центральний процесорний пристрій MSP430.

50. Призначення і будова периферійного інтерфейсу USART MSP430.

51. Призначення та будова АЦП, ЦАП та таймерів А і В MSP430.

52. Наведіть приклади можливого застосування МК MSP430 в електромеханіці, електричних апаратах і електропобутовій техніці (ЕПТ).

53. Призначення, будова та принципдії таймерів А і В МК MSP430.

54. Призначення, будова та принципдії сторжового таймера MSP430.

55. Призначення, будова та принцип дії температурного сенсора в МК.

56. Наведіть загальну характеристику системи команд МК MSP430.

57. Призначення, будова, принцип дії, режими роботи таймерів А і В МК MSP430 і їх програмування.

58. Призначення, будова, принцип дії компаратора А в МК MSP430.

59. Призначення та будова джерела опорної напруги МК MSP430.

60. Призначення стека і указівника стека (SP/R1) і їх принцип дії.

61. Призначення, будова та принцип дії аналогового мультиплексора.

62. Назвіть значимі фактори які впливають на вибір мікроконтролера.

3. ЛАБОРАТОРНИЙ ПРАКТИКУМ І СТЕНД МК 51

Останніми роками позначилася тенденція широкого впровадження у системи керування електромагнітними і електроенергетичними системами та їх елементами однокристальних мікроконтролерів серії МК51 як найбільш популярних і корисних у зв'язку з низькою їх вартістю і доступністю. Тому і в основу лабораторного стенда для вивчення однокристальних мікро-ЕОМ покладений МК такого ж типу – 1816ВЕ51 (i8051), що є «de facto» всесвітнім промисловим стандартом.

Особливостями сімейства МК51 є такі: високий рівень уніфікації елементів, можливість перепрограмування на реалізацію тих або інших функцій без заміни складу комплексу технічних засобів, низькі витрати на обслуговування, контроль і діагностування, програмна сумісність з високопродуктивними (25 MIPS) МК фірми Cygnal.

Розроблені на кафедрі «Електричні апарати» НТУ «ХПІ» стенди МК51 і лабораторний практикум (див. нижче) дозволяють глибше вивчити структуру і особливості роботи МК51 у різних режимах, набути практичних навичок у роботі з мікроконтролерами цього сімейства, у складанні і налагодженні програм за допомогою крос-засобів.

3.1. Лабораторна робота 1. Дослідження лабораторного стенда мікроконтролера МК51

Мета роботи – ознайомлення зі структурою стенда мікроконтролера серії МК51, командами, режимами роботи, складанням і виконанням програм.

Порядок підготовки до роботи

Вивчити:

- рекомендовану літературу [20, 21, 25, 26];
- опис лабораторної установки;
- методичні вказівки до виконання лабораторної роботи.

Опис лабораторного стенда. Лабораторний стенд включає:

- макет мікроконтролера МК51 (рис. 3.1), який за допомогою послідовного порту RS232C, що має розв'язку на оптронах, з'єднаний з ПЕОМ ;

• джерело живлення +5 В (15 В) виконано на окремій платі і підключається до електричної мережі напругою 220 В і частотою 50 Гц;

• двопроменевий осцилограф С1-55, який використовується для дослідження різних режимів роботи МК51 шляхом підключення його до тих або інших пристроїв за допомогою гнізд відповідно до функціональної схеми лабораторного стенда, наведеної на рис. 3.1.

Електрична схема і комутаційні гнізда розташовані на лицьовій панелі макета, яка виконана з органічного скла. Тут же знаходяться перемикачі і кнопки, які використовуються для подання на входи різних пристроїв логічної інформації, а також 24 світлодіоди, які призначені для індикації стану шини адреси і даних, порту Р1.

Основні відомості про лабораторний стенд МК51

Лабораторний стенд (ЛС), виконаний на базі однокристального мікроконтролера МК51, належить до мікропроцесорних пристроїв обчислювальної техніки і може використовуватись у системах керування технологічним устаткуванням у складі випробувального, науково-дослідного, контрольно-вимірювального та іншого устаткування. Призначення виводів мікросхеми, узагальнена і структурна схеми МК наведені на рис. 2.1, 2.2 і 3.8.

Лабораторний стенд дозволяє проводити такі дослідження:

• структурної схеми МК51, арифметико-логічного пристрою, резидентної пам’яті, пристрою керування і синхронізації;

• портів введення-виведення інформації і доступу до зовнішньої пам’яті;

• таймера-лічильника, послідовного інтерфейсу, системи переривань і режимів читання та запису даних;

• особливих режимів роботи, завантаження і верифікації прикладних програм, покрокового виконання команд, скидання, режимів холостого ходу і зниженого енергоспоживання;

• системи команд, введення, редагування, трансляції і налагодження прикладних програм;

• організації взаємодії мікроконтролера з об’єктом керування, вводу інформації з датчиків, виводу сигналів, що управляють, з МК51, масштабування, перетворення кодів;

- аналого-цифрових (АЦП) і цифро-аналогових (ЦАП) перетворень, введення інформації з клавіатури, виведення і відображення інформації;
- послідовного порту МК51 для зв'язку з інтерфейсом RS232C.

Структура функціональної схеми

Функціональна схема лабораторного стенда, призначеного для вивчення мікроконтролерів сімейства МК51, наведена на рис. 3.1.

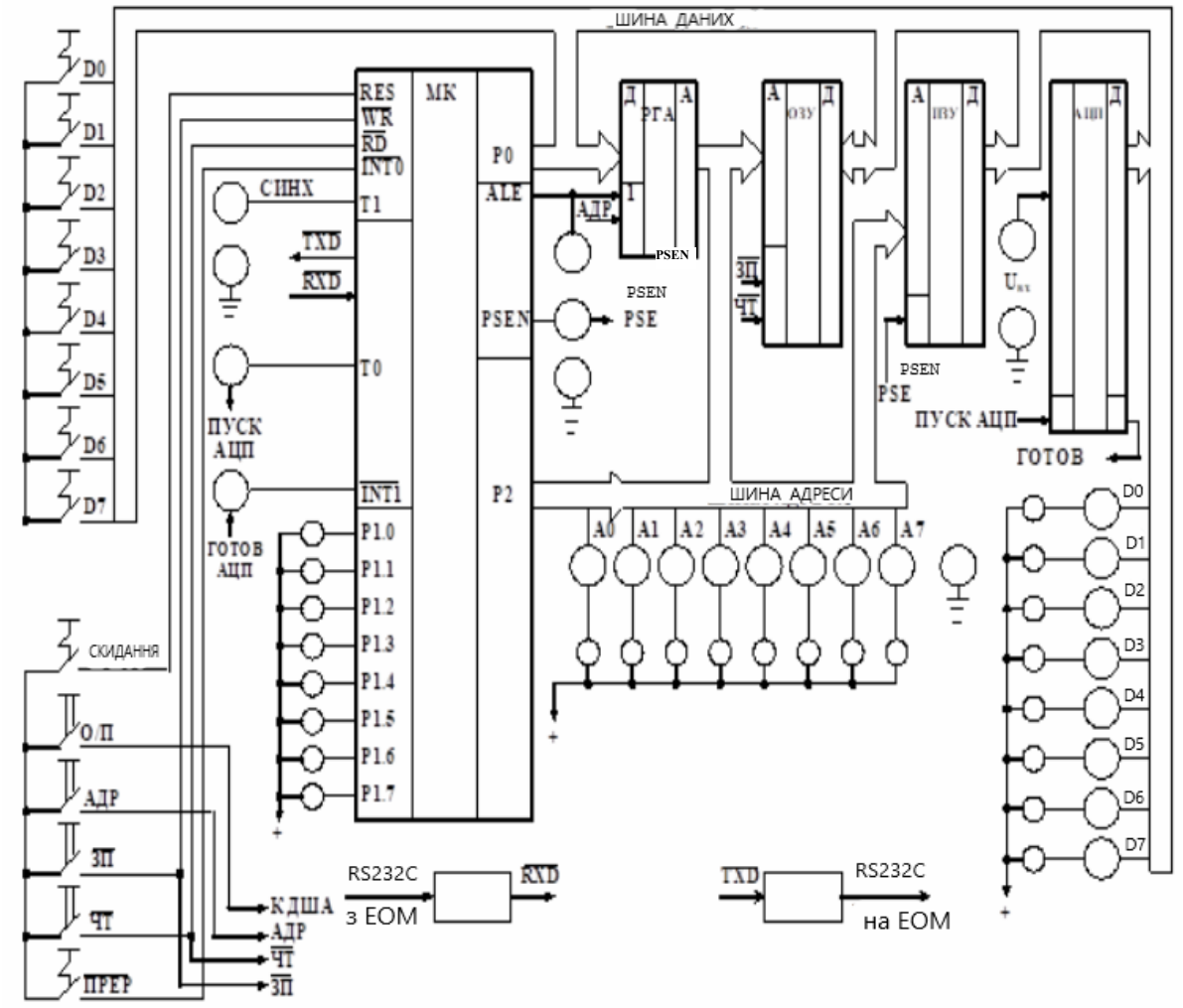


Рисунок 3.1 – Функціональна схема лабораторного стенда МК51

Схема включає: мікроконтролер типу КР1830ВЕ31, ВІС ОЗП – КР537РУ10, ВІС ППЗП – К537РФ2, АЦП – К1113ПВ1, мультиплексор на 8 каналів 590КН6, 8-розрядний регістр адреси, 8-розрядний регістр даних (індикація), 8-розрядний регістр даних, 8-розрядний регістр аналого-цифрового перетворювача (АЦП), 8-розрядний регістр введення, кла-

віші для введення інформації, логіку для керування доступом до ОЗП/ПЗП і регістрів, 8 кнопок Д0-Д7 з фіксацією для введення адреси і даних для програмування стенда, кнопку «СКИДАННЯ» з фіксацією, кнопку ПРЕР (переривання) з фіксацією, кнопки АДР (адреса), ЗП (запис), ЧТ (читання) – без фіксації, кнопку О/П (ОЗП/ПЗП, англ. Start ROM-RAM), 24 світлодіоди АЛ307Б для індикації стану шини даних, шини адреси і порту Р1, 30 контактних гнізд для підключення вимірювальних приладів та імітаційних сигналів, інтерфейсні мікросхеми для зв'язку стенда з ПЕОМ, роз'єм для підключення ПЕОМ через порт COM2, блок живлення.

Призначення елементів і функціонування схеми лабораторного стенда

Контактні гнізда «СИНХ», «ПУСК АЦП», «ГОТОВИЙ АЦП», «ALE», «PSEN», «Uвх» служать відповідно для виконання студентами необхідних підключень у схемі. Для підключення зовнішнього генератора подій служить контактне гніздо «СИНХ». До нього може бути підключений генератор тактової частоти і генератор зовнішніх одиночних тактових імпульсів, які підраховуватиме внутрішній лічильник мікро-ЕОМ відповідно до заданої програми. Гнізда «ПУСК АЦП» і «ГОТОВИЙ АЦП» служать для підключення керуючих сигналів аналого-цифрового перетворювача стенда. При цьому вихід Т0 мікро-ЕОМ виступає як джерело сигналів запуску на вимірювання, а вхід INT1 – для прийому сигналів готовності чергового вимірювання АЦП.

Контактні гнізда «ALE», «PSEN», служать для підключення осцилографа з метою спостереження часових діаграм роботи мікро-ЕОМ або для синхронізації роботи зовнішніх пристроїв, підключених до стенда.

Гніздо «Uвх» призначено для подання на вхід АЦП вимірюваної на-пруги.

Кнопки Д0–Д7 мають фіксацію і призначені для набору кодів даних і адреси, які необхідно занести в оперативну пам'ять контролера.

Світлодіодні індикатори Д0–Д7 забезпечують відображення кодів даних, які задаються відповідними кнопками.

Світлодіодні індикатори А0–А7 призначені для відображення стану шини адреси контролера.

Кнопка «СКИДАННЯ» (рос. «СБРОС») служить для початкового встановлення однокристальної мікро-ЕОМ. При її натисненні мікро-ЕОМ встановлює на шині даних адресу команди 0000Н.

Кнопка «АДР» служить для запису коду даних, набраного за допомогою кнопок Д0–Д7 в регістр адреси контролера РГА.

Кнопка «ЗП» призначена для запису коду даних, набраних за допомогою кнопок Д0–Д7, в заданий елемент оперативної пам'яті контролера.

Кнопка «ЧТ» служить для читання коду даних, який записаний в елементі пам'яті, адреса якої зберігається в регістрі РГА, і відображення його – на світлодіодних індикаторах Д0–Д7.

Кнопка «ПРЕР» служить для подачі на вхід переривання мікроЕОМ команди, яка забезпечує виконання її програми обслуговування переривання.

Контактне гніздо «Т0» призначено для подачі сигналу на виконання циклу вимірювань. Цей сигнал надходить на відповідний вхід мікро-ЕОМ, що забезпечує виконання підпрограми обслуговування АЦП.

Світлодіодні індикатори, підключені до ліній порту Р1, служать для виведення логічних сигналів, які повідомляють про виконання деяких програмних подій. Наприклад, цей порт може бути використаний для виведення 8-розрядних двійкових кодів результатів обчислень. Оптронні розв'язки ОР1 і ОР2 служать для електричного розділення ланцюга каналу зв'язку з ПЕОМ.

Лабораторний стенд побудований з функціонально закінчених модулів, які з'єднані між собою за магістральним принципом. Системна магістраль (канал) дозволяє адресувати 64К 8-розрядних слів пам'яті команд і пам'яті даних. Активним пристроєм у каналі є МК, крім випадку, коли МК знаходиться у режимі скидання.

У лабораторному стенді передбачені два основні режими роботи: «ПДП» і «РОБОТА». У режимі «ПДП» введення в ОЗП з клавіатури програми або даних виконується при натиснутій кнопці «СКИДАННЯ» (рос. «СБРОС»).

У режимі «РОБОТА» кнопка «СКИДАННЯ» віджата, мікроконтролер виконує програму, яка знаходиться або у ПЗП, або раніше введена в ОЗП.

При включенні стенда мікроконтролер починає виконувати програму за нульовою адресою. Для роботи разом з ЕОМ IBM PC повинна

бути запущена програма, яка розташована у ПЗП, а для автономної роботи – програма, яка вводиться користувачем в ОЗП. Для вибору джерела початку виконання програми (ОЗП/ПЗП) служить кнопка S10 – О/П. Коли кнопка віджата, за адресою 0–32768 знаходиться ПЗП, а ОЗП знаходиться у кінці адресного простору (32768–40960). При натиснутій кнопці за адресою 0–8192 знаходиться ОЗП, а за адресою 8193-32768 – ПЗП. Головним елементом електричної принципової схеми мікроконтролерного стенда є мікроконтролер КР1816ВЕ51, частота роботи якого задається зовнішнім кварцем. Виводи порту Р0 утворюють двонаправлену 8-бітову шину даних, яка з'єднує контролер з ОЗП, ПЗП, портами введення-виведення, зібраними на регістрах.

Система ПЗП (ДД8) включає програми режимів початкового пуску, резистентного перевіряючого тесту, збирання і обробки інформації з АЦП і портами введення-виведення, програми для зв'язку з комп'ютером. Молодша частина шини адреси А0–А7 фіксується у регістрі ДД7 за спадом сигналу, який надходить на вхід С. Залежно від режимів роботи стенда на вхід С ре-гістра ДД7 можуть поступати або сигнал ALE з мікроконтролера, або сигнал ADR з кнопки «АДР». Функцію перемикача залежно від режиму роботи виконує мультиплексор, зібраний на мікросхемах ДД 4.4 і ДД 5.4. Керування мультиплексором проходить за сигналом RES. Старша частина адреси А8–А15 виходить з порту Р2 МК. У сумі це складає 16 адресних ліній, які забезпечують доступ до 65535 байт зовнішньої пам'яті.

Схема дозволу для доступу до ОЗП / ПЗП побудована на елементах ДД 12.4 / ДД 3.3. Якщо МК встановлює на А15 логічну одиницю, тобто адресу 32768, то включається доступ до ОЗП.

При натиснутій кнопці О/П доступ до ПЗП включається, якщо А13 або А14 = 1, тобто при адресі 8192-32768. В інших діапазонах пам'яті працює ОЗП.

Сигнал RD RAM (читання пам'яті) формується на мікросхемах ДД 2.1, ДД 2.4. На вхід елемента «І» подається RD, PSEN. RDC1. Таким чином, вибірка даних з ОЗП відбуватиметься при читанні МК з пам'яті, запуску програми з ОЗП і читанні з клавіатури у режимі «ПДП».

Сигнал WR RAM формується також на елементі ДД 2.2. Запис в ОЗП можливий як у режимі «РОБОТА», так і з клавіатури у режимі «ПДП».

На елементі ДД 2.3 зібрана схема усунення дзенькіту контактів кнопок «ПЕРЕР» і «СКИДАННЯ». Елементи «І» сполучені так, що утворюють нетактований RS-тригер. Схема керування регістром ДД1 (кнопка даних) виконана так, що вона пропускає дані на вихід тільки у режимі «ПДП», якщо натиснута кнопка «АДР» або «ЗП».

Аналого-цифрова частина зібрана на мультиплексорі аналогових каналів КР560КН6 (ДА1), 10-бітовому АЦП К1113ПВ1 (ДА2) і карті читання даних з АЦП, зібраній на регістрі КР1533ИР33 (ДД).

АЦП може працювати у двох режимах з різною точністю – 8 і 10 біт. При розподільній здатності 8 біт, якщо ведеться передача перетвореної інформації через послідовний порт, швидкість буде трохи вища.

Процес перетворення аналогових сигналів у цифрову форму відбувається таким чином. Мікроконтролер записує у порт Р1 номер каналу для перетворення (0–8), потім необхідно провести запуск АЦП послідовного наближення К1113 ПВ1 (ДА2). Це виконується шляхом установки біта 5 порту Р3 у 1. Після закінчення перетворення на виходах АЦП встановлюється перетворений цифровий код сигналу, який подається на вхід вибраного каналу мультиплексора. Одночасно з видачею даних АЦП тестує переривання мікроконтролера, після чого викликається відповідна процедура оброблення, яка через регістр ДД16 прочитує цей код. Для перетворення всіх восьми каналів необхідно у тому ж порядку повторити послідовність операцій, які описані вище. Таким чином, швидкість перетворення аналогової інформації знаходиться у прямій залежності від кількості каналів введення, які обробляються. Для одного каналу введення час буде мінімальним і дорівнювати 30 мкс, для N каналів він складатиме 30 мкс×N.

Інтерфейс зв’язку лабораторного стенда з комп’ютером типу IBM зібраний відповідно до вимог, встановлених стандартом RS232C. Крім того, на оптронах, зібраних у Д1Г8 корпус (АОТ101), зроблена гальванічна розв’язка від решти частини схеми.

Схемою передавача є перетворювач рівнів TTL у напругу –5…+5 В.

Приймач проводить зворотне перетворення. За логічну одиницю вважається рівень напруги –3...–5 В, а за логічний нуль +3...+5 В.

Порядок виконання роботи

Для підготовки лабораторного стенда необхідно:

- підключити стенд до мережі змінного струму 220 В;
- вимикач живлення на задній панелі стенда перевести у верхнє положення («Включити»); при цьому на індикаторах стенду з'явиться якась випадкова інформація;
- перевести мікроконтролер у режим ручного керування (без ПЗП); для цього клавіша «О/П» повинна бути натиснута;
- натиснути клавішу «СКИДАННЯ» (фіксується при натисненні).

Завдання 1. Виконати запис даних у пам'ять згідно з варіантами (табл. 3.1). Для виконання цієї операції необхідно спочатку встановити адресу заданої комірки пам'яті у двійковому коді, а після цього записати у комірку необхідні дані (також у двійковому коді).

Встановлення необхідної адреси виконується таким чином:

1. На клавішах з фіксацією D0–D7 набирають адресу в двійковому коді. При цьому треба звернути увагу на те, що D_0 – це молодший розряд, а D_7 – старший.

Наприклад, щоб встановити адресу комірки пам'яті з номером 5, потрібно натиснути клавіші D2 і D0, що відповідає двійковому коду 00000101 (інші клавіші повинні бути не натиснуті).

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
□	□	□	□	□	■	□	■

2. Для запису адреси, встановленої на клавішах, у регістр адреси потрібно натиснути клавішу «АДР» (без фіксації). Після цього на індикаторах адреси A_0–A_7 з'явиться встановлена адреса у двійковому коді.

У нашому прикладі для адреси 5 індикатори матимуть такий вигляд:

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
○	○	○	○	○	●	○	●

3. Для запису даних у комірку пам'яті за встановленою адресою необхідно на клавішах D0–D7 набрати відповідне число у двійковому коді, після чого натиснути клавішу «ЗП». При цьому на індикаторах D0–D7 повинно з'явитися набране число у двійковому коді. Таким чином, дані будуть записані за встановленою адресою (адреса відображається на індикаторах A_0–A_7). Наприклад, щоб записати число 7, потрібно натиснути клавіші D2, D1, D0, що відповідає двійковому коду 00000111:

D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
□ □ □ □ □ ■ ■ ■

Після цього потрібно натиснути клавішу «ЗП». При цьому на індикаторах D0–D7 з’явиться записане за встановленою адресою у комірку пам’яті число 7 у двійковому коді:

D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
○ ○ ○ ○ ○ ● ● ●

Примітка. У разі запису даних у декілька комірок пам’яті адресу кожної з них перед записом треба встановлювати окремо.

Варіанти даних для виконання завдання наведені у табл. 3.1.

Таблиця 3.1 – Варіанти даних

Номер варі-анта	Початкова адреса N	Дані для введення у комірки, починаючи з адреси N			Номер варіанта	Початкова адреса N	Дані для введення у комірки, починаючи з адреси N		
		N	N + 1	N + 2			N	N + 1	N + 2
1	00	37	54	34	14	39	58	23	11
2	03	69	44	46	15	42	54	123	03
3	06	33	50	22	16	45	12	64	10
4	09	07	00	255	17	48	15	54	45
5	12	04	214	23	18	51	08	68	25
6	15	23	16	56	19	54	09	82	124
7	18	69	61	04	20	57	35	28	02
8	21	38	69	36	21	60	01	78	255
9	24	57	68	12	22	63	23	54	12
10	27	68	46	69	23	66	58	12	02
11	30	89	56	98	24	69	54	84	200
12	33	125	91	245	25	72	98	58	145
13	36	45	27	56	26	75	65	05	204

Завдання 2. Дослідити зміст пам’яті.

1. Аналогічно п. 1 попереднього завдання встановлюємо клавішами D0–D7 потрібну адресу комірки пам’яті згідно з варіантом.

2. Натискаємо клавішу «ЧТ». Після цього на індикаторах повинен з’явитися у двійковому коді вміст комірки пам’яті за встановленою адресою.

3. Одержані результати звести у табл. 3.2.

Таблиця 3.2 – Одержані результати

АДРЕСА	Початкові дані	Прочитані дані
N	Двійковий код, встановлений у комірці N	Двійковий код, прочитаний з комірки N
N + 1	Двійковий код, встановлений у комірці N + 1	Двійковий код, прочитаний з комірки N + 1
N + 2	Двійковий код, встановлений у комірці N + 2	Двійковий код, прочитаний з комірки N + 2

Зміст звіту

- мета роботи, структурна схема стенда;
- одержані результати (табл. 3.2).

Контрольні запитання

1. Структура МК51і призначення виводів його мікросхеми.
2. Структура лабораторного стенда і система команд МК51.
3. Основні функції і правила користування стендом.
4. Режими роботи лабораторного стенда.

3.2. Лабораторна робота 2. Програмування і введення програм з клавіатури стенда МК51

Мета роботи – вивчення команд і особливостей організації програм на мові Асемблер МК51, їх введення з клавіатури стенда та виконання.

Порядок підготовки до роботи

- вивчити рекомендовану літературу [20, 21, 25, 26];
- ознайомитися з системою команд мікроконтролера і основними їх функціями (див. табл. 2.7); звернути увагу на особливості складання і виконання програм на мові Асемблер МК51.

Використовуване устаткування – лабораторний стенд МК51.

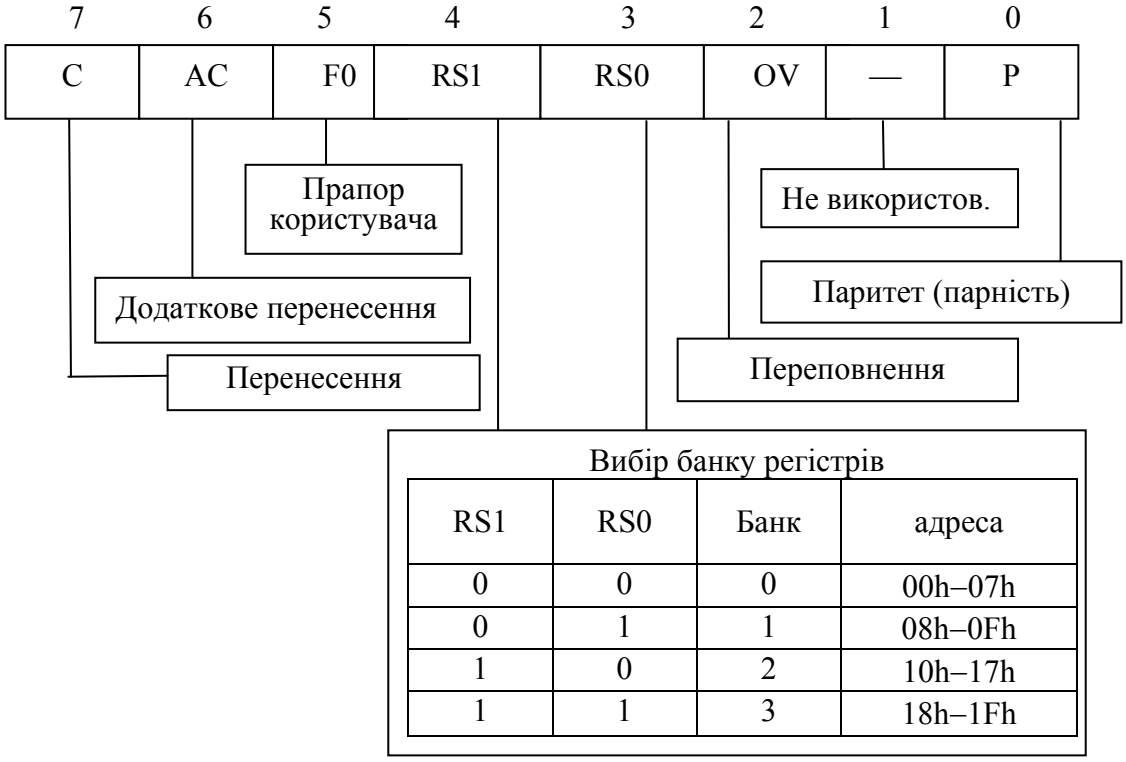
Основні відомості

Аналогічно з регістром ознак процесора серії КР580 у мікроконтролері МК51 є регістр, який містить в собі інформацію про результати виконання обчислень: ознаки перенесення (C), додаткового перенесення (AC), перепов-нення (OV), паритету (P) і прапор (біт) користувача (F0). Але,

на відміну від регістру ознак, цей регістр ще містить у собі інформацію про те, який з чотирьох можливих банків регістрів використовується програмою. Цей регістр називається регістром «слова стану» (PSW – англ.).

Для встановлення відповідного банку регістрів, який використовуватиметься програмою, необхідно на початку програми встановити відповідні значення бітів RS0 і RS1 у регістрі «слова стану» (PSW).

Формат регістру «слова стану» програми ССП (PSW) наведений нижче на рис. 3.2.



RS1	RS0	Банк	адреса
0	0	0	00h–07h
0	1	1	08h–0Fh
1	0	2	10h–17h
1	1	3	18h–1Fh

Рисунок 3.2 – Формат регістру «слова стану» (PSW)

Для цього використовують команди встановлення і скидання бітів відповідно SETB bit і CLR bit. Адреса регістру «слова стану» – 0Dh. Формат цих двобайтових команд такий: SETB xxxx xxxx, або CLR xxxx xxxx.

Перший байт зліва – двійковий код команди SETB (11010010) або CLR (11000010). Старші чотири біти у другому байті команд – це адреси слова стану PSW (1101 у двійковому коді), а молодші чотири біти – номер біта у регістрі PSW, який встановлюється або скидається (RS0 (0011), RS1 (0100)).

Наприклад, щоб встановити нульовий банк регістрів, треба виконати дві команди скидання бітів RS0 і RS1 у регістрі слова стану (PSW):

CLR PSW. 3 11000010 1101 0011
CLR PSW. 4 11000010 1101 0100

Для того щоб встановити перший банк регістрів, треба виконати одну команду скидання і одну встановлення бітів:

SETB PSW. 3 11010010 11010011
CLR PSW. 4 11000010 11010100

Коди операцій SETB і CLR: SETB – 11010010 CLR 11000010.

Для того щоб результат розрахунків програми можна було перевірити візуально, у кінці програми розрахунків треба поставити команду виведення вмісту акумулятора у порт P1, до якого підключені світлодіоди. Таким чином, судячи зі стану світлодіодів, можна прочитувати вміст акумулятора. При цьому треба врахувати одну особливість – дані у порт виводяться в інвертованому вигляді, тобто перш ніж записувати акумулятор у порт, його треба проінвертувати. Команда інвертування вмісту акумулятора – CPL A 11110100. Команда запису в порт вмісту акумулятора така:

MOV P1, A 1111010 10010000.

У кінці програми треба поставити оператор відносного переходу, щоб програма зупинилася на циклі на останньому операторі.

Це пояснюється тим, що у системі команд МК51 у зв'язку зі специфікою роботи не передбачена команда зупинки програми користувача, а тому у кінці програми замість оператора зупинки використовується оператор переходу на відповідну кількість кроків назад. У мнемонічному коді це повинно виглядати так: Мітка1: NOP – команда «нема операції»;

SJMP Мітка1: команда відносного переходу

У двійковому коді для нашого випадку це виглядає таким чином:

NOP 00000000 SJMP rel 10000000 11111101

Порядок виконання роботи

Завдання 1. Скласти алгоритм і програму арифметичного складання (або віднімання) двох чисел – N1 і N2 і логічної операції з числом N3 згідно з відповідними виразами (1–4). До складу програми повинні увійти команди встановлення активного банку регістрів (у регістрі PSW), команди занесення даних у регістри Rn і акумулятор, команди арифметич-

них і логічних операцій, інвертування вмісту акумулятора, запису вмісту акумулятора у порт, а також команда відносного переходу в кінці програми. Варіанти завдання наводяться нижче у табл. 3.3.

$(N1 + N2) \vee N3$ (1) $\quad (N1 - N2) \wedge N3$ (2)

$(N1 + N2) \veebar N3$ (3) $\quad (N1 - N2) \veebar N3$ (4)

Таблиця 3.3 – Варіанти даних у десятковому коді

Номер за списком	Номер виразу	N1	N2	N3	Банк робочих регістрів	Номери регістрів, Rn
1	1	3	9	6	0	R1, R2
2	2	6	4	10	1	R3, R4
3	3	10	2	56	2	R5, R6
4	4	64	48	12	3	R1, R2
5	1	4	12	16	2	R3, R4
6	2	18	2	11	1	R5, R6
7	3	40	8	1	0	R1, R2
8	4	56	6	25	1	R3, R4
9	1	10	38	6	2	R5, R6
10	2	128	64	2	3	R1, R2
11	3	2	2	2	2	R3, R4
12	4	4	2	2	1	R5, R6
13	1	1	3	4	0	R1, R2
14	2	26	2	12	1	R3, R4
15	3	10	10	1	2	R5, R6
16	4	100	96	2	3	R1, R2
17	1	4	12	6	2	R3, R4
18	2	7	5	10	1	R5, R6
19	3	14	2	56	0	R1, R2
20	4	64	48	18	1	R3, R4
21	1	4	12	16	2	R5, R6
22	2	18	2	10	3	R1, R2
23	3	40	8	2	2	R3, R4
24	4	56	6	28	1	R5, R6
25	1	1	5	3	0	R1, R2

Регістри для виконання програми використовуються згідно з варіантом завдання. При виконанні завдання доцільно користуватися командами, які наведені у табл. 3.4.

Таблиця 3.4 – Команди Асемблера МК51

Мнемокод команди	Двійковий код команди	Операція. Назва команди
SJMP rel	10000000 + rel	(PC) ← (PC) + 2 + rel. Короткий відносний перехід всередині сторінки у 256 байт, де rel – 8-бітна відносна адреса передачі керування
MOV Rn, *#d*	01111rrr # d	(Rn) ← *# d*. Завантаження у регістр безпосереднього операнда константи *# d*
MOV A, Rn	11101rrr	(A) ← (Rn). Пересилання з регістра в акумулятор; rrr – трибітове поле, яке визначає номер регістра (Rn) в двійковому коді (n = 0÷7) (000–R0, 001–R1 ... 111–R7)
MOV A, *#d*	01110100 *#d*	(A) ← *# d*. Завантаження в акумулятор константи
MOV P1, A	11110101 10010000	(P1) ← (A). Записати у порт P1 вміст акумулятора
ADD A, Rn	00101rrr	(A)←(A) + (Rn). Додавання акумул. з регістром (n = 0÷7)
SUBB A, Rn	10011rrr	(A)← (A) – (C) – (Rn). Віднімання з акумулятора регістра і займу
RL A	00100011	Циклічний зсув акумулятора вліво
RR A	00000011	Циклічний зсув акумулятора вправо
ANL A,Rn	01011rrr	(A) ← (A) ∧ (Rn). Логічне «І» акумулятора і регістра
ORL A, Rn	01001rrr	(A) ← (A) ∨ (Rn). Логічне «АБО» акумулятора і регістра
XRL A,Rn	01101rrr	(A) ← (A) ⊻ (Rn). Логічне «Виключаюче АБО» акумулятора і регістра
MOVX @DPTR, A	11110000	((DPTR)) ← (A). Пересилання байта даних з акумулятора в комірки розширеної зовнішньої пам’яті даних
CPL A	11110100	(A) ← ($\overline{A}$). Інвертування вмісту акумулятора
NOP	00000000	Нема операції

Завдання 2. Виконати розрахунки відповідно до розробленої програми на лабораторному стенді і вручну згідно з варіантом табл. 3.3. Одержані результати порівняти і зробити висновки.

Приклад програми для варіанта 16 наводиться нижче.

Мітка	Команда	1-й байт	2-й байт
	SETB PSW.3	11010010	11010011
	SETB PSW.4	11010010	11010100
	MOV A, #100	01110100	01100100
	MOV R1, #96	01111001	01100000
	SUBB A, R1	10011001	

MOV R2, #2	01111010	00000010
XRL A, R2	01101010	
CPL A	11110100	
MOV P1, A	11110101	10010000
LOOP: NOP	00000000	
SJMP LOOP	10000000	(адр. LOOP:)

Примітка. Введення програми або даних з клавіатури ЛС виконується при натиснутій кнопці «СКИД», а «О/П» віджатій. Для виконання програми необхідно встановити початкову адресу програми і віджати кнопку «СКИД».

Зміст звіту

- текст програми в мнемонічному і двійковому коді;
- результати розрахунків за програмою і вручну.

Контрольні запитання

1. Поясніть програму і команди, які використовуються у ній.
2. Як здійснюється вибір активного банку регістрів і виконання роботи у цілому?
3. Наведіть приклади команд роботи з бітами та їх формати.
4. Поясніть команди арифметичних і логічних операцій.
5. Викладіть команди переходів і виклику підпрограм.
6. Наведіть команди запису в порт та інвертування вмісту акумулятора і їх двійкові коди.
7. Наведіть команди пересилання і циклічного зсуву акумулятора.
8. Призначення і формат регістру «слова стану» (PSW).
9. Особливості команд відносного, довгого, абсолютного і непрямого переходів.
10. Поясніть команди SETB PSW.3 і CLR PSW.3 і запишіть їх двійковий код та наведіть приклади їх застосування.

3.3. Лабораторна робота 3. Складання і налагоджування програм

Мета роботи – навчитися складати, компілювати і налагоджувати програми на мові Асемблер МК51 крос-засобами, використовуючи персональний комп’ютер.

Порядок підготовки до роботи

- вивчити рекомендовану літературу [20, 21, 25, 26];
- ознайомитися з системою команд МК51 і правилами роботи з програмним забезпеченням крос-засобів та налагоджувачем FD-51;
- ознайомитися із структурою карти резидентної (внутрішньої) пам'яті даних МК51.

Опис устаткування

- персональний комп'ютер сумісний з IBM;
- програмне забезпечення крос-засобів;
- лабораторний стенд МК51.

Основні відомості. До мінімального складу програмного забезпечення крос-засобів належать:

- системна програма для введення початкового тексту програми, його редагування і запису на зовнішній носій (так званий редактор тексту або символьний редактор);
- програма-транслятор, яка перетворює початковий текст програми на об'єктний модуль ASM-51, з редактором зовнішніх зв'язків, що дозволяє включити у програму модулі, розроблені незалежно один від одного;
- програма-налагоджувач – емулятор, що забезпечує перекладання преміщуваних програмних модулів на абсолютні адреси.

До засобів розробки і налагоджування належать програми-Асемблера, компонування і програми-налагоджувачі – емулятори.

Програма-Асемблер – це програма, яка транслює початкову програму в об'єктні коди. Вона виконує привласнення дійсних адрес, перетворення чисел, привласнення дійсних значень символьним змінним.

Програма на Асемблері складається з *команд, директив і керуючих параметрів.*

Команди Асемблера – мнемонічне кодування дій, які повинен виконати процесор.

Директиви – псевдокоманди, які при трансляції не перетворюються на машинні коди команд і використовуються для визначення змінних, структури програми, задання констант і т. ін.

Керуючі параметри вказують транслятору на виконання певних дій.

Програма Асемблера ASM-51 дозволяє використовувати модульний принцип, тобто обробляти окремі частини складної програми – модулі.

Після налагоджування всіх модулів окремо вони можуть бути зібрані в один файл спеціальною програмою компонування RL51.

Початкова програма (файл name.a51) створюється за допомогою редактора текстових файлів і містить команди на мові Асемблер, керуючі директиви параметри асемблера і коментарі. Після створення початкової програми вона обробляється викликаною програмою асемблера ASM51.EXE (транслятором) за допомогою команди ASM51 name.a51.

Результатом роботи програми буде:

- *об'єктний файл,* поданий у машинних кодах (форма початкових даних асемблера, які виконуються, що є абсолютним форматом шістнадцяткового коду, і файл, який може бути запрограмований на МК51 і включає інформацію, необхідну для складання і налагоджування);
- *лістинговий файл* (запис початкової програми і об'єктного коду, в який Асемблер вносить повідомлення про всі помилки кодування і службову інформацію).

Асемблер має декілька директив, які дозволяють користувачу встановити символьні імена (директива EQU), зарезервувати та ініціалізувати місце у пам'яті, керувати розміщенням програми.

Директиви керування: ORG – керування лічильником адреси; (задає асемблеру адресу елемента пам'яті, в якій повинна розміщуватися наступна за нею команда прикладної програми); END – кінець програми (вказує на закінчення трансляції).

Налагоджувач асемблерних програм дозволяє:

- завантажити для налагодження HEX-файл, що здійснюється трансляторами з мови Асемблера (крос-засоби), а також файли чистого двійкового коду, зчитані з ПЗП;
- переглянути на екрані дизасембльований текст завантаженої програми, адреси і коди команд, область імітованого ОЗП даних, зовнішньої пам'яті, пам'яті програм, вміст усіх регістрів;
- виконати завантажену програму покроково з прогляданням результатів після кожного кроку, внести зміни у програму;
- внести зміни у вміст регістрів, прапорів і пам'яті та ін.

Функціональні клавіші, які використовуються у цій роботі:

– F1 – виконати поточну інструкцію завантаженої програми (поточною є підкреслена інструкція у вікні тексту програми (рис. 3.3));

– F3 – дозволяє надавати числову інформацію у десятковому, двійковому, шістнадцятковому кодах для регістрів;

– F4 – перемикає вікно пам’яті даних з внутрішньої пам’яті на зовнішню;

– F6 – перемикає форму подання пам’яті у вікні в двійкову і шістнадцяткову системи.

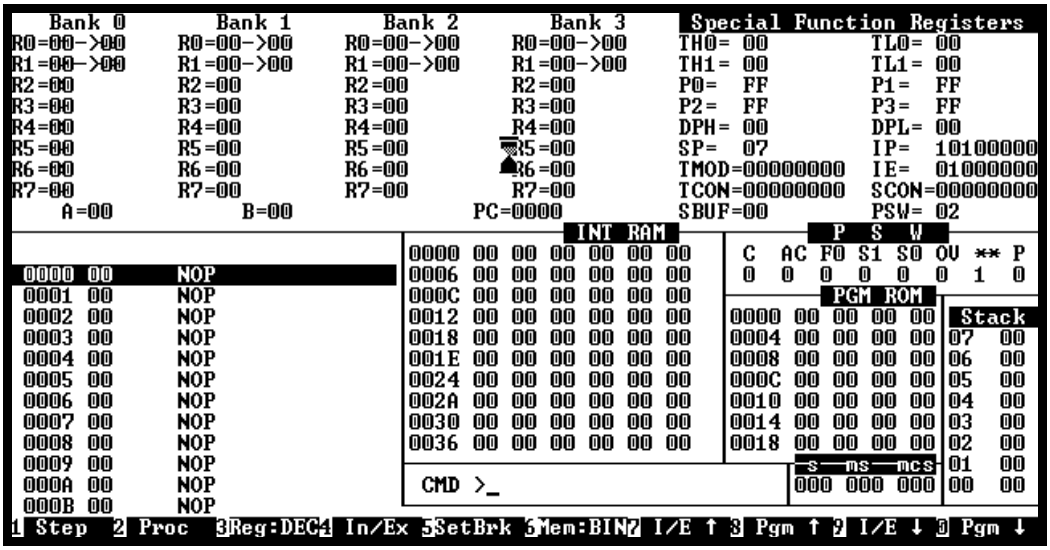


Рисунок 3.3 – Вигляд вікна налагоджувача емулятора FD51

Команди налагоджувача:

H – для отримання довідки;

L [«тип пам’яті» «початкова адреса»] «файл. HEX»[/A] – завантажити файл у пам’ять. Тип пам’яті: I (внутрішня), E (зовнішня) і P (програмна). Початкова адреса і тип пам’яті вказується тільки при завантаженні двійкового файлу;

/A – ключ тільки при завантаженні програми ISIS – II Macro Assembler;

R «номер регістра» = «число» – занести число у регістр поточного банку. Наприклад, R4 = FF;

«ім’я регістра» = «число» – занести число у регістр спеціального призначення: A, B TH0, TH1, TL0, TL1, DPL, DPH, DPTR, SP, IP, IE, TMOD, TCON, SCON, SBUF, PC. Наприклад, TH0 = FF;

«ім’я прапора» = «число» – встановити або зняти прапор (біт) у PSW. Наприклад, S1 = 0;

RST – імітується скидання процесора;
QUIT – вихід у DOS;
N – очищення вмісту налагоджувача.

Правила запису програм на мові Асемблер для МК51

Початковий текст програми на мові Асемблер має певний формат. Кожна команда (і псевдокоманда) це рядок чотириланкової конструкції, яка наводиться нижче:

Мітка	Операція	Операнд	Коментар

Ланки відділяються одна від одної довільною кількістю пропусків.

Мітка. У полі мітки розміщується символічне ім'я елемента пам'яті, в якому зберігається відмічена команда або операнд. Мітка літерно-цифровою комбінацією, що починається з літери. Використовуються тільки літери латинського алфавіту. Асемблер МК51 допускає використання у мітках символу підкреслення (_). Довжина мітки не повинна перевищувати 31 символ для МК51. Мітка завжди завершується двокрапкою.

Псевдокоманди Асемблера не перетворюються на двійкові коди, а тому не можуть мати міток. Винятки становлять псевдокоманди резервування пам'яті і визначення даних (DS, DB, DW). У псевдокоманд, які здійснюють визначення символічних імен, у полі мітки записується обумовлене символічне ім'я, після якого двокрапка не ставиться.

Як символічні імена і мітки не можуть бути використані мнемокоди команд, псевдокоманд і операторів Асемблера, а також мнемонічні позначення регістрів та інших внутрішніх блоків МК.

Операція. У полі операції записується мнемонічне позначення команди МК або псевдокоманди Асемблера, що є скороченням (абревіатурою) повного англійського найменування виконуваної дії. Наприклад: MOV– move – перемістити JMP – jump – перейти, DB – define byte – визначити байт.

Для МК51 використовується строго визначений і обмежений набір мнемонічних кодів. Будь-який інший набір символів, розміщений у полі операції, сприймається Асемблером як помилковий.

Операнди. У цьому полі визначаються операнди (або операнд), які беруть участь в операції. Команди Асемблера можуть бути без-, одно-,

або двооперандними. Операнди розділяються комою (,). Операнд може бути заданий безпосередньо або у вигляді його адреси (прямої або непрямої). Безпосередній операнд подається числом (MOV A, #18) символічним або ім'ям (ADDC A, # OPER 1) з обов'язковою вказівкою префікса безпосереднього операнда (#). Пряма адреса операнда може бути задана мнемонічним позначенням (IN A, P1), числом (INC 50), символічним ім'ям (MOV A, MEMORY). Непряма адресація вказується префіксом @. У командах передачі керування операндом може бути число (LCALL 0135H), мітка (JMP LABEL), непряма адреса (JMP @ A) або вираз (JMP ¤ – 2, де ¤ – поточний вміст лічильника команд). Символічні імена і мітки, що використовуються як команди повинні бути визначені, а числа подані з указівкою системи числення, для чого використовується суфікс (літера, яка стоїть після числа): B – для двійкової, Q – для вісімкової, D – для десяткової і H – для шістнадцяткової. Число без суфікса за умовчанням вважається десятковим.

Обробка виразів у процесі трансляції

Асемблер МК51 допускає використання виразів у полі операндів, значення яких обчислюються у процесі трансляції. Вираз є сукупністю симво-лічних імен і чисел, зв'язаних оператором Асемблера. Оператори Асемблера забезпечують виконання арифметичних («+» – складання, «–» – віднімання, « * » – множення, « / » – цілочислове ділення, MOD – ділення за модулем) і логічних (OR – АБО, AND – I, XOR, – виключаюче АБО, NOT – HI) операцій у форматі двобайтових слів. Наприклад, запис ADD A, # (NOT 13+1) еквівалентна запису ADD A, # OF3H і забезпечує складання вмісту акумулятора з числом 13, поданим у додатковому коді. Використовуються також оператори LOW і HIGH, які дозволяють виділити молодший і старший байти двобайтового операнда.

Коментар. Поле для коментаря може бути використано програмістом для текстового або символьного пояснення логічної організації прикладної програми. Поле для коментаря не компілюється Асемблером, а тому в ньому допустимо використання будь-яких символів. За правилами мови Асемблера поле для коментаря починається після крапки з комою.

Псевдокоманди Асемблера. Асемблююча програма транслює початкову програму в об'єктні коди. Хоча програма бере на себе багато чого з рутинних завдань програміста, таких як привласнення дійсних адрес, перетворення чисел, привласнення дійсних значень символьним змінним

і т. д., але програміст усе-таки повинен указати їй деякі параметри, наприклад, початкову адресу прикладної програми, кінець асембльованої програми, формати даних та ін. Усю цю інформацію програміст вставляє у початковий текст своєї прикладної програми у вигляді псевдокоманд (директив) Асемблера, які тільки керують процесом трансляції і не перетворюються на коди об'єктної програми.

Наприклад, псевдокоманда ORG 10H задає Асемблеру адресу елемента пам'яті (10H), в якій повинна бути розташована наступна за нею команда прикладної програми.

Псевдокомандою EQU можна будь-якому символічному імені, яке використовується у програмі, поставити у відповідність певний операнд. Наприклад, запис SEM EQU 15 приводить до того, що у процесі компіляції усюди, де зустрічається символічне ім'я SEM, його буде замінено числом 15.

Символічні імена операндів, переобумовлених у процесі виконання програми, визначаються псевдокомандою SET:

```
ALFA   SET  3
...
ALFA   SET  ALFA +1
```

Асемблер МК51 дозволяє визначити символічне ім'я як адреси внутрішніх (псевдокоманда DATA), зовнішніх (XDATA) даних або адресу біта (псевдокоманда BIT). Наприклад, директива ERROR...FLAG BIT 25H3 визначає символічне ім'я ERROR_FLAG як третьої комірки середовища ОЗП з адресою 25H.

Псевдокоманда DB забезпечує занесення у ПП константи, яка є байтом.

Псевдокомандою END програміст дає Асемблеру вказівку про закінчення трансляції.

У результаті трансляції повинна бути одержана карта пам'яті програм, де кожному елементу пам'яті буде поставлений у відповідність код, який зберігається у ній. Відповідно формату команд, для подання їх об'єктних кодів виділяється один, два або три елементи пам'яті програм. У першій комірці завжди розташовується код операції, у другій (а для МК51 і у третій) – безпосередній операнд, адреса операнда, який прямо адресується, або зсув (для команд передачі керування МК51). Для команд LCALL і LJMP у другому і третьому байтах об'єктного коду вказується

адреса передачі керування (у другому – старша частина, у третьому – молодша).

Послідовний інтерфейс

При використанні МК необхідно здійснювати передачу і прийом інформації, подану послідовним кодом (молодшими бітами вперед) у повному дуплексному режимі обміну. Для цього використовується універсальний асинхронний приймач-передавач (УАПП). До його складу входить спеціальний буфер-регістр (SBUF). Запис байта у буфер приводить до автоматичного перепису байта у зсуваючий регістр передавача та ініціює початок передачі байта. Наявність буфера-регістра приймача дозволяє суміщати операцію читання раніше прийнятого байта з прийомом чергового байта.

Послідовний порт МК51 може працювати у чотирьох режимах.

Режим 0. У цьому режимі інформація приймається і передається через зовнішнє виведення входу приймача (RXD). Приймаються або передаються 8 біт даних. Через зовнішні вихідні виводи передавача (TXD) видаються імпульси зсуву, які супроводжують кожен біт. Частота передачі біта інформації дорівнює 1/12 частоти резонатора.

Режим 1. У цьому режимі передаються через TXD або приймаються з RXD 10 біт інформації: старт-біт (0), 8 біт даних і стоп-біт (1). Швидкість прийому / передачі – величина змінна і задається таймером.

Режим 2. Дані передаються через TXD або приймаються з RXD по 11 біт: старт-біт, 8 біт даних і стоп-біт. При передачі даних дев’ятий біт може приймати значення «0» або «1». Наприклад, для підвищення надійності передачі шляхом контролю по парності у ньому може бути поміщено значення ознаки паритету і слова стану програми (PSW.0). Частота прийомів/передач вибирається програмно і може дорівнювати 1/32 або 1/64 частоти резонатора залежно від керуючого біта SMOD.

Режим 3. Збігається з режимом 2 в усіх деталях за винятком частоти прийому передачі, що є величиною змінною і задається таймером.

Керування режимом роботи УАПП здійснюється через спеціальний регістр (SCON). Цей регістр містить не тільки керуючі біти, які визначають режим роботи послідовного порту, але і дев’ятий біт прийнятих або переданих даних (RB8 і TB8), а також біти переривання приймачів-передавачів RI, TI.

Швидкість прийому передачі у режимі «0» залежить тільки від резонансної частоти кварцового резонатора: $f_0 = f_p / 12$. Призначення бітів регістрів керування відображені у табл. 3.5.

За один машинний цикл послідовний порт передає один біт інформації. У режимах 1, 2, 3 швидкість прийому передачі залежить від значення керуючого біта SMOD у регістрі керування потужністю PCON. 7

Програма «Монітор» для ПЕОМ надає користувачу для забезпечення зручного зв'язку з лабораторним стендом МК51 ряд можливостей, які наводяться нижче:

- прийом (передача) даних із (в) внутрішнього ОЗП контролера;
- прийом (передача) даних із (в) ПЕОМ;
- запуск і налагодження програми користувача.

Наявність зрозумілого інтерфейсу і набір директив керування дозволяють проводити налагоджування програми користувача і контроль результатів роботи безпосередньо на стенді. Програма «Монітор» написана на мові Паскаль, і для забезпечення її роботи потрібна ПЕОМ, сумісна з IBM, ОЗП не менше 2 Мбайт, і наявність не менше ніж 20 Кбайт на диску. Зв'язок зі стендом здійснюється через послідовний порт COM2. «Монітор» подається у вигляді одного виконуваного файлу monitor.exe і не потребує додаткових налагоджень. Запуск програми здійснюється з командного рядка DOS командою > monitor.exe (або будь-яким іншим чином).

Таблиця 3.5 – Призначення бітів регістра керування УАПП (SCON)

Символ	Позиція	Ім'я і призначення
SM0 SM1	SCON. 7 SCON. 6	Біти керування режимом УАПП. Встановлюються і скидаються програмно (див. примітку)
SM2	SCON. 5	Біт керування режимом УАПП. Встановлюється програмно для заборони прийому повідомлень, у яких дев'ятий біт дорівнює «0»
REN	SCON. 4	Біт дозволу прийому. Встановлюється і скидається програмно для дозволу / заборони прийому послідовних даних
TB8	SCON. 3	Передача біта 8. Встановлюється / скидається програмно. Для задання дев'ятого переданого біта у режимі УАПП – 9 біт
RB8	SCON. 2	Прийом біта 8. Встановлюється / скидається апаратно. Для фіксації дев'ятого прийнятого біта у режимі УАПП – 9 біт
TI	SCON. 1	Прапор переривання передавача встановлюється апаратно при закінченні передачі байта. Скидається програмно після обслуговування переривання
RI	SCON. 0	Прапор переривання приймача встановлюється апаратно при прийомі байта. Скидається програмно після обслуговування переривання

Примітка. Коди режимів роботи УАПП

SM0	SM1	Режим роботи УАПП
0	0	Зсуваючий регістр розширення введення-виведення
0	1	УАПП 8 біт – Змінна швидкість передачі
1	0	УАПП 9 біт – Фіксована швидкість передачі
1	1	УАПП 9 біт – Змінна швидкість передачі

Примітка 1. Перед запуском програми переконайтеся, що стенд увімкнений і підключений до ПЕОМ.

Після запуску програми «Монітор» на екрані буде виведене вікно програми, яка складається з 7 вікон (рис. 3.4).

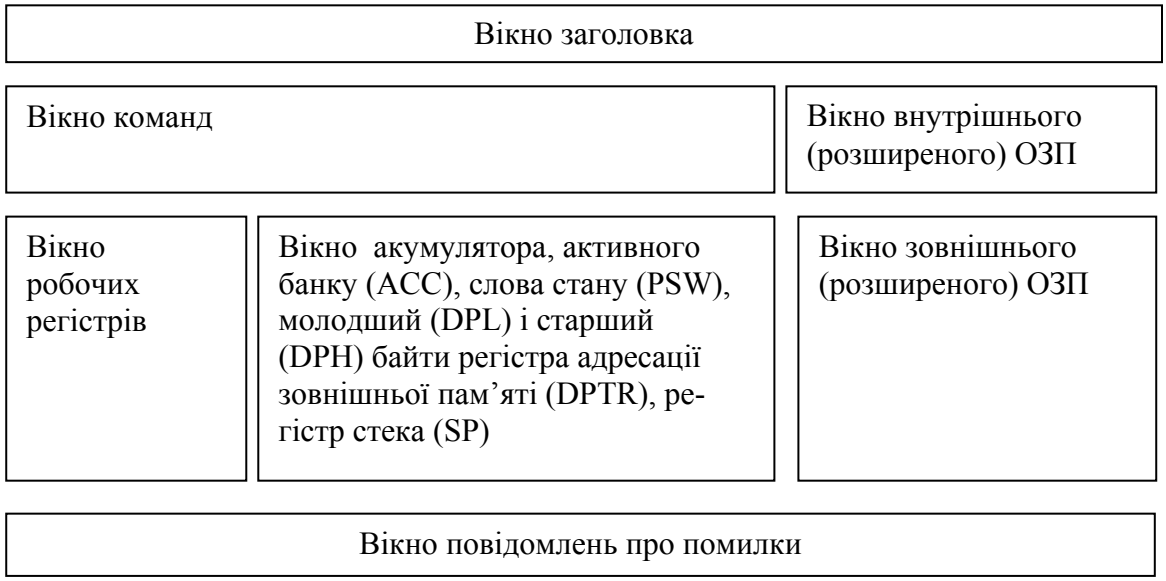


Рисунок 3.4 – Вікно програми «Монітор»

Введення директив здійснюється у вікні команд (місце для нової директиви позначається запитом DOS). Результати роботи директиви будуть відображені у відповідному вікні. У разі виникнення помилки повідомлення про неї буде виведено у вікні «Повідомлення про помилки».

Примітка 2. Якщо директива містить цифрові параметри (адреса або дані), то їх треба вводити у 16-тковому коді.

Примітка 3. Якщо після запуску програми «Монітор» виникає непоправна помилка, подібна до «Відсутній зв'язок з контролером», то вихід з програми здійснюється клавішею «Esc». Під час роботи вихід з програми здійснюється шляхом введення директиви Q.

Директиви програми «Монітор» наведені у табл. 3.6.

Таблиця 3.6 – Директиви програми «Монітор»

Код	Формат	Призначення	Примітки
L	LFName.hex	Ввод масиву даних з файлу	FName.hex – ім'я 16-кового файлу
G	GAAAA BBBB	Перехід до програми користувача із зупинкою	AAAA – стартова адреса BBBB – адреса зупинки
J	JAAAA	Запуск програми користувача без зупинки	AAAAA – стартова адреса
P	PAAAA	Продовження виконання програми від точки зупинки	AAAAA – адреса повторного запуску програми користувача
D	DAAAA BBBB	Читання масиву даних з ОЗП у ПЕОМ	AAAA – початкова адреса BBBB – кінцева адреса
K	KAAAA BBBB	Знаходження контрольної суми масиву даних	AAAA – початкова адреса BBBB – кінцева адреса
F	FAAAA BBBB CC	Заповнення масиву константою	AAAA – початкова адреса BBBB – кінцева адреса CC – константа
T	TAAAA BBBB CCCC	Копіювання масиву даних в ОЗП	AAAA – початкова адреса BBBB – кінцева адреса CCCC – початкова адреса копії масиву
M	MAAAA	Читання байта даних із зовнішнього ОЗП	AAAA – адреса байта даних
R	RAA	Читання байта даних із внутрішнього ОЗП	AA – адреса байта даних
Y	YDD	Запис байта даних за поточною адресою	DD – байт даних R0 – адреса призначення
A	A <масив>	Запис 32 байт у внутрішнє ОЗП	60h – початкова адреса масиву
B	B	Читання 32 байт з внутрішнього ОЗП	60h – початкова адреса масиву – джерела
Q	Q	Вихід з програми	

Програма «Монітор» для стенда МК51. Монітор є ядром програмного забезпечення стенда. В основу інтерфейсу обміну мікропроцесорного контролера (МК) стенда і ПЕОМ покладений протокол, який передбачає передачу спеціальної серії інформаційних повідомлень, необхідних для успішного функціонування робочої програми користувача. Алгоритм складається з таких кроків.

Крок 1. Виконується ініціалізація контролера стенда, яка складається з таких операцій:

- формування заборони всіх переривань;
- призначення стека монітора;
- заповнення ОЗП константою FF (очищення пам’яті);
- установлення робочого банку даних (банк 3-18H…1FH);
- очищення ОЗП стану програми користувача (запис значення 0 у комірки ACCBuf, PSWBuf, DPTR);
- призначення стека користувача (DPTR: = # Stek1);
- установлення адреси процедури збереження стану програми користувача (DPTR: = # Return);
- початкове установлення коду в точці зупинки ([Kod]:= # 02 код команди LJMP) і адреси повернення ([Kod + 1]: = Return);
- ініціалізація порту послідовного обміну.

Алгоритм функціонування монітора наведено на рис. 3.5.

Крок 2. Виконується запит обміну з ПЕОМ шляхом посилання в канал обміну коду запиту 00H і введення коду директиви монітора;

Крок 3. Виконується аналіз коду директиви і відповідно до табл. 3.6 одна з директив монітора. Якщо одержано неправильний код директиви, виконується перехід до кроку 2.

Виконання роботи.

Частина 1. Налагоджування програми крос-засобами

Завдання. Записати у мнемонічному коді скореговану програму в роботі № 2 у файл name.a51. Зробити трансляцію файлу в об’єктний і 16-тковий коди. Перевірити роботу програми у середовищі програми-налагоджувача FD51. Занести у звіт вміст акумулятора.

Підготовка до роботи

1. Включити стенд. Усі клавіші стенда перевести у відтиснене положення, а потім включити комп’ютер.

2. Натиснути клавішу «Esc» (виконується завантаження комп’ютера Norton Commander (NC)). Працюємо у лівій панелі NC. Для цього, утримуючи клавішу «ALT», натискаємо клавішу F1.

У вікні дисків вибираємо клавішами керування курсором («←» вліво, «→» вправо) диск D (жовтий колір на D), а на деяких комп’ютерах диски C або E.

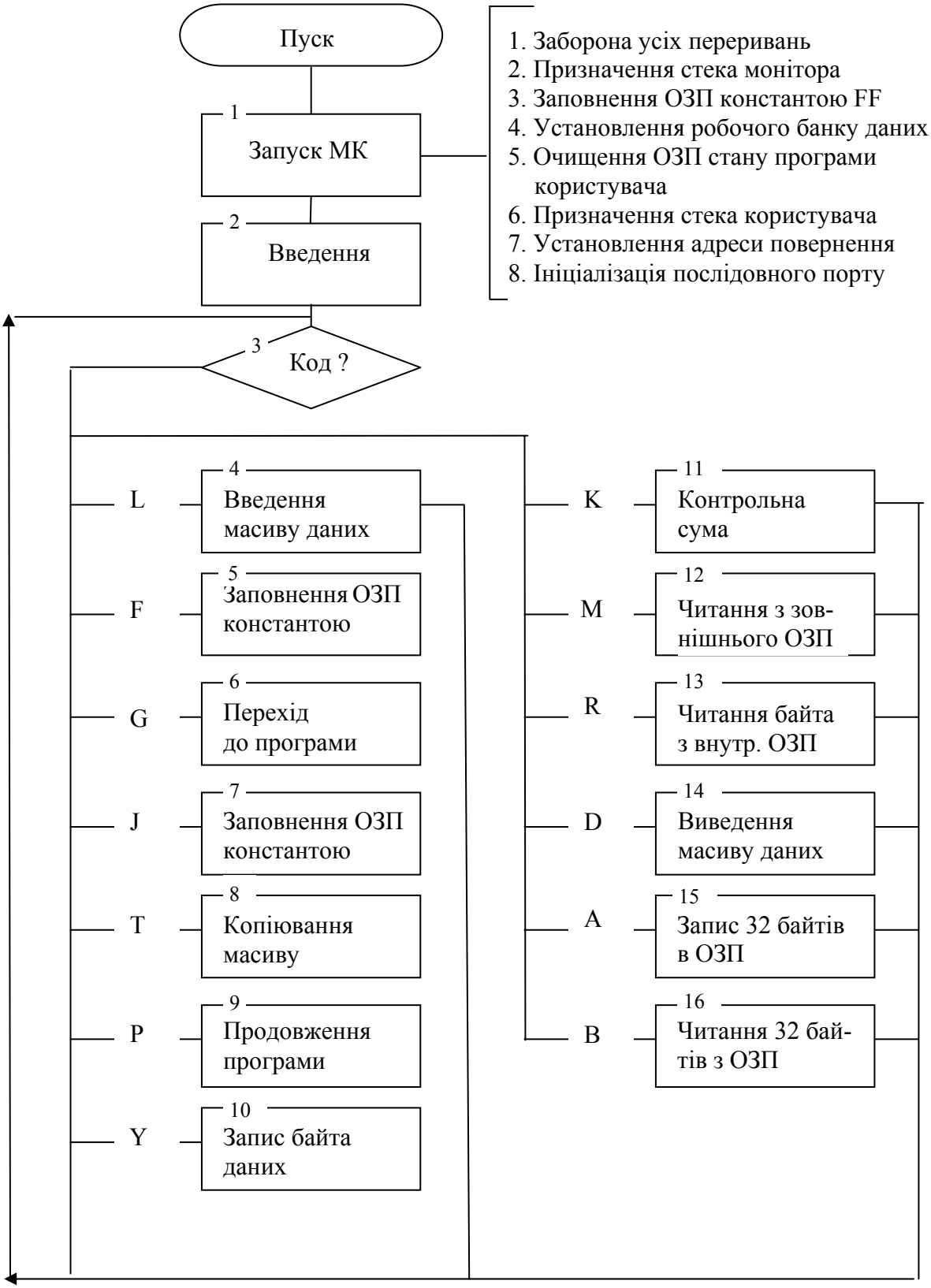


Рисунок 3.5 – Алгоритм функціонування монітора лабораторного стенда

3. На вибраному диску, наприклад D, знаходимо директорію Lab_3 (D:\) (клавішами «↑», «↓» наводимо курсор на директорію Lab_3 і, натиснувши клавішу «Enter», потрапляємо у директорію D:\Lab_3.

4. Знаходячись у директорії D:\Lab_3, створимо новий файл. Для цього натискаємо комбінацію клавіш «Shift»+«F4» і у вікні редактора, яке з'явилося на екрані, введемо ім'я нового файлу name.a51 (де name, – прізвище студента до 7 символів); натискаємо «Enter» і у відкритому вікні нового файлу набираємо програму в мнемонічному коді, побудовану в лабораторній роботі 2, скореговану таким чином.

На початку програми записуємо директиву ORG 00H, потім команду SJMP 30H, за нею директиву ORG 30H, далі програму, яка використовувалася у лабораторній роботі 2. Блок команд в кінці програми, який раніше використовувався для зациклювання (NOP-SJMP), замінюється одним оператором повернення з підпрограми – RET. Це пояснюється тим, що будь-яка програма, завантажена з ПЕОМ за директивою «L», сприймається «монітором» стенда як підпрограма. Як відомо, повернення з підпрограми виконується за командою RET.

Створений таким чином новий файл name.a51 (програма) буде мати такий вигляд:

```
ORG 00H
SJMP 30H
ORG 30H
SETB PSW.3
SETB PSW.4
MOV A, #100
MOV R1, #96
SUBB A, R1
MOV R2, #2
XRL A, R2
CPL A
MOV P1, A
RET
```

Для збереження файлу і змін у ньому використовується клавіша «F2».

Виконання програми

1. Створену нову програму (файл name.a51) обробляємо програмою Асемблер (файл ASM51.EXE). Для цього введемо у командному рядку CMD> команду ASM51 name.a51. Результатом виконання програми буде об'єктний файл (name.obj), поданий у машинних кодах, а також лістинговий файл name.lst, який містить текст початкової програми, її шістнадцятковий код, відомості про помилки (якщо вони є) і службову інформацію.

2. Одержаний файл name.obj необхідно перекодувати в шіснадцятковий файл, для чого, використовуючи команду OH.EXE, введемо у командний рядок OH name.obj і натиснемо «ENTER». Результатом виконання цієї команди буде файл у шістнадцятковому коді name.hex. Надалі виконання роботи ведеться саме з цим файлом (name.hex).

3. У директорії D:\LAB_3 знаходимо файл налагоджувача програм FD51.EXE і, натиснувши клавішу «ENTER», запускаємо його.

4. У відкритому вікні налагоджувача у вікні команд CMD> вводимо команду L name.hex і натискаємо «ENTER». У вікні програми буде виведено дизасембльовану програму.

5. Натискаючи клавішу «F1», крок за кроком виконаємо програму. Дійшовши до команди RET, зафіксуємо у звіті вміст акумулятора.

6. Вихід з програми налагоджувача виконується командою QUIT, яка вводиться у вікні команд CMD>.

7. Результати роботи (файли з назвою name і з розширеннями a51, hex, lst, obj) зберегти у директорії D:\LAB_3.

Частина 2. Виконання програми і перевірка результатів

Підготовка до роботи

Перед виконанням частини 2 необхідно скорегувати програму, складену в частині 1. Для цього досить замінити на початку програми у директивах ORG адреси з 00H на 8000H, а 30H на 8030H і у команді SJMP з 30H на 8030H. Подальше виконання роботи проводиться згідно з п. 1–6 ч. 2.

Порядок виконання роботи

1. Користуючись програмою «Монітор» ПЕОМ, за допомогою директиви «F» заповнити адресний простір з 8000 h до 8050 h константою FF.

2. За допомогою директиви «L» завантажити hex-файл програми в ОЗП стенда за адресою 8000h.

3. За допомогою директиви «D» з урахуванням початкової і кінцевої адрес вивести на екран текст завантаженої програми.

4. Користуючись директивою «J» з початковою адресою програми як параметром, запустити програму на виконання.

5. Перевірити індикацію порту P1 лабораторного стенда МК51.

6. Перевірити двійковий код вмісту порту P1, що висвітився на індикаторах (P0–P7). Він повинен відповідати інвертованому вмісту акумулятора, який був зафіксований у звіті у першій частині роботи.

Зміст регістру A (відображається у 16-ковому коді) і вміст порту P1 (відображається у двійковому коді) повинні бути інверсними.

Зміст звіту. Повний текст скорегованої програми з директивами і коментарями та результати розрахунку.

Контрольні запитання

1. Структура програми на Асемблері.
2. Функції налагоджувача.
3. Структура вікна і команди налагоджувача FD51.
4. Функції і структура вікна програми «Монітор».
5. Директиви програми «Монітор».
6. Алгоритм функціонування монітора лабораторного стенда.
7. Призначення програми ASM-51.EXE і результати її виконання.
8. Будова, принцип дії і режими роботи УАПП.
9. Зміст файлів name.obj і name.lst.
10. Призначення команди OH name.obj і результати її виконання.
11. Який файл використовується при роботі з налагоджувачем програм?
12. Як запустити файл налагоджувача програм?

3.4. Лабораторна робота 4. Дослідження аналого-цифрового перетворювача

Мета роботи – ознайомитися з принципом дії аналого-цифрового перетворювача (АЦП) і навчитися використовувати його можливості при складанні керуючих програм на асемблері МК51.

Порядок підготовки до роботи

– вивчити рекомендовану літературу [20, 21, 26];

– ознайомитися з принципом дії і правилами роботи з АЦП.

Використовуване устаткування: ПЕОМ, лабораторний стенд МК51, джерело живлення напругою 5В, вольтметр, потенціометр.

Основні відомості про аналого-цифровий перетворювач (АЦП) і цифро-аналоговий перетворювач (ЦАП)

При використанні мікроконтролера для вирішення деяких завдань керування виникає необхідність проведення перетворення аналогових сигналів у цифрові і навпаки – цифрового коду в аналоговий сигнал. Для цього використовують АЦП і ЦАП. Одним із найпоширеніших є метод розрядного зрівноважування, названий також методом послідовного наближення. В АЦП, побудованому цим методом, код у регістрі результату змінюється так, щоб забезпечити по можливості швидке зрівноважування вхідної напруги або струму напругою або струмом, одержуваним з виходу ЦАП, приєднаного до згаданого регістру. Зрівноважування починається зі старшого розряду. У цьому розряді спочатку встановлюється одиниця й оцінюється знак різниці перетвореного сигналу й сигналу, що врівноважує, формованого у ЦАП. Якщо з'ясовується, що сигнал, що врівноважує, менше перетвореного, то встановлена у старшому розряді одиниця надалі зберігається, а якщо більше – то одиниця скидається, тобто надалі у цьому розряді буде зберігатися нуль. Далі у такий же спосіб перевіряється, чи потрібна одиниця у сусідньому молодшому розряді регістру. І так зрівноважування триває доти, поки не будуть опитані всі розряди регістра, включаючи самий молодший.

При виборі цих пристроїв враховують розрядність, швидкодію і вартість. Перетворення аналогового сигналу в цифровий код, який може бути оброблений МК, можна здійснити декількома способами, що наводяться нижче.

1. Апаратний. Реалізований на основі ВІС АЦП, що підключається до порту МК. У цьому випадку МК тільки ініціює АЦП і через задані періоди дискретизації прочитує з нього цифровий код. Цей спосіб характеризується найвищою швидкодією, але вимагає використання ВІС АЦП, що не завжди виправдано економічно.

2. Апаратно-програмний. Реалізується на основі ВІС ЦАП і програми зважування біт (послідовних наближень побітового врівноваження). Цей спосіб характеризується високою швидкодією і можливістю використання простих і дешевих схем ЦАП і операційного підсилювача.

3. Програмно-апаратний – побудований на основі методу подвійної інтеграції. Найбільш дешевий і повільний. Може забезпечити високу точність перетворення. З додаткового устаткування вимагає два операційних підсилювача і аналоговий мультиплексор на два входи.

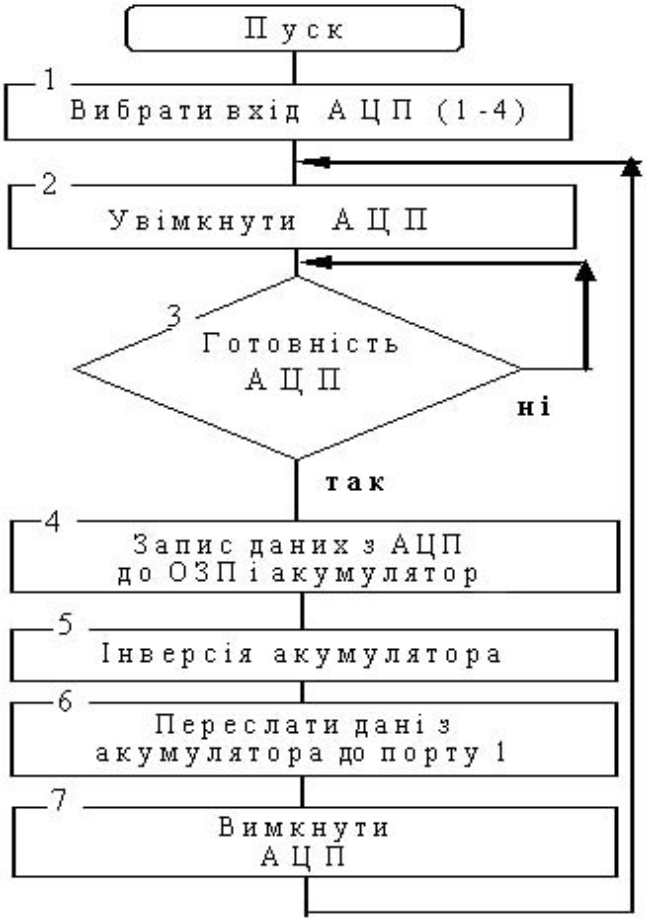
4. Апаратно-програмний. Реалізується на основі перетворювача напруга – частота і програми вимірювання періодів сигналу.

Після оброблення одержаного сигналу МК видає сигнали керування у двійковому коді, які необхідно перед подачею на пристрій керування об'єктом перетворити на аналоговий сигнал. Для цієї мети використовують ЦАП. ВІС ЦАП підключають до одного з портів МК. Видача аналогового керуючого сигналу здійснюється за командою (OUTL P1, A). При цьому на виході ЦАП з'явиться напруга (струм), пропорційна двійковому коду, завантаженому в порт 1.

Ряд об'єктів керування може потребувати безперервного керуючого сигналу в складній формі. Для реалізації такого сигналу в МК використовуються цифрові методи інтеграції. На кожному інтервалі часу функція, яка інтегрується, замінюється її дискретним значенням, а програма реалізується з використанням процедур видачі коду і часової затримки заданої тривалості.

У цій роботі використовується апаратний спосіб перетворення аналогового сигналу в двійковий код. Аналого-цифрова частина зібрана на мультиплексорі аналогових каналів КР560КН6 (ДА1), десятибітового АЦП К1113ПВ1 (ДА2), і карти читання даних з АЦП, зібраної на регістрі КР1533ИР33 (ДД). АЦП може працювати у двох режимах з різною точністю – 8 і 10 біт. При роздільній здатності 8 біт, якщо ведеться передача перетвореної інформації через послідовний порт, швидкість буде трохи вища. Перетворені у двійковий код дані поступають у порт Р1 мікроконтролера.

Алгоритм прийому даних мікроконтролером від АЦП наведений на рис. 3.6, а текст цієї програми поданий нижче.



Програма прийому даних МК від АЦП

```
AJMP AZP
ORG 030H
AZP:
LOOP:
CLR P3.4
WAIT:
JB P3.3,WAIT
MOV R0, # C000H
MOVX A, @R0
MOV P1, A
SETB P3.4
SJMP LOOP
END
```

Рисунок 3.6 – Алгоритм прийому даних мікроконтролером від АЦП

Для вибору номера входу треба занести код входу в регістр, який керує номером входу АЦП. Адреса регістра – A000h.

Коди входів АЦП:

Вхід 1 (0)	00B	00H	Вхід 3 (2)	10B	02H
Вхід 2 (1)	01B	01H	Вхід 4 (3)	11B	03H

Порядок виконання роботи. Для виконання завдання необхідно зібрати схему, наведену на рис. 3.7, у якій подаватиметься сигнал (постійна напруга від 0 до 5 В) на вхід АЦП.

Схема складається з джерела живлення напругою 5 В, потенціометра і вольтметра, підключеного паралельно до входу АЦП. Вольтметр використовується для еталонного контролю напруги сигналу.

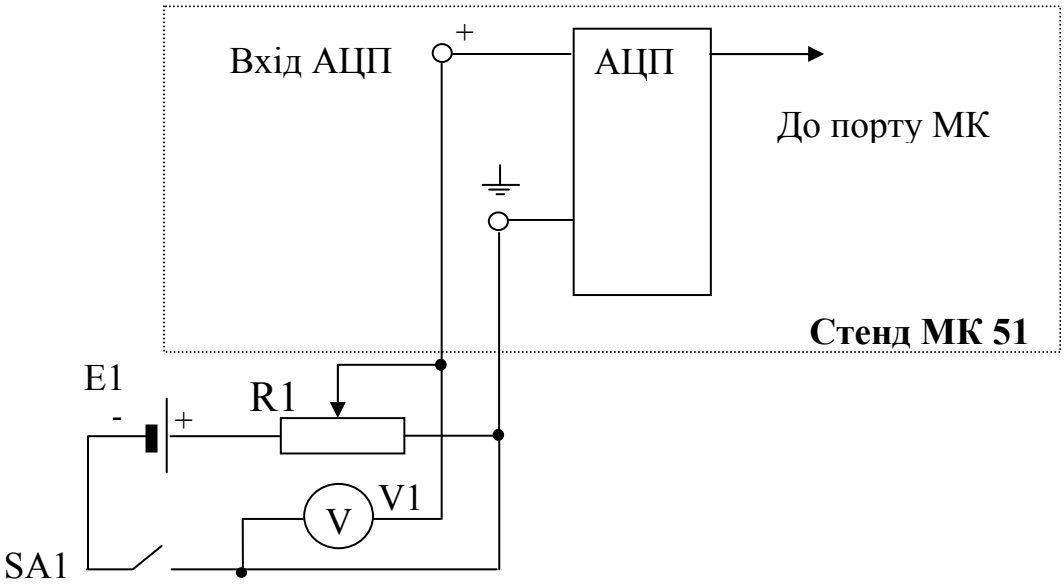


Рисунок 3.7 – Схема емулятора вхідного сигналу для АЦП

Після того як схема підключена, в ОЗП стенда з персонального комп'ютера необхідно завантажити програму для роботи з АЦП. Програма у 16-ковому коді знаходиться у файлі azp.hex, і після завантаження в ОЗП повинна читати сигнал із входу АЦП і передавати його у порт P1. При цьому відповідно до зміни напруги на вході АЦП (змінюється потенціометром) змінюватиметься і зміст порту P1, що відображатиметься на світлодіодах

P0–P7. Для завантаження програми треба перевести стенд у режим роботи з ПЗП, а далі діяти за схемою лабораторної роботи № 3 (ч. 2): запустити програму «monitor» і за допомогою директиви «L» завантажити файл azp.hex. Після того як програма завантажена в ОЗП стенда, його треба перевести у режим роботи без ПЗП і вимкнути режим скидання.

За допомогою директиви J8030 програма запускається на виконання, після чого на індикаторах порту P0–P7 у режимі реального часу починає відображатися стан входу АЦП.

Завдання 1. Розрахувати розрядність цифрової індикації порту P1.

Розв'язання. Для вирішення цього завдання необхідно 5–10 вибраних значень напруги записати у таблицю стану світлодіодів (вміст порту P1) і розрахувати вагомість розряду порту P1 за формулою

$$C = U/P1,$$

де C – відношення, яке характеризує величину зміни напруги на вході АЦП, що викликає зміну сигналу на виході (у порту Р1) на одну одиницю; U – напруга на вході АЦП, показана вольтметром V1; $P1$ – зміст порту Р1 у десятковому коді.

Зміст звіту:

- схема підключення джерела сигналу до входу АЦП (рис. 3.7);
- таблиця залежності сигналів у порту Р1 від сигналів на АЦП;
- розрахунок вартості розряду індикації порту.

Контрольні запитання

1. Викладіть принцип дії схеми (рис. 3.7) і порядок виконання лабораторної роботи.
2. На чому засновані принципи дії АЦП і ЦАП ?
3. Поясніть алгоритм програми прийому даних мікроконтролером з АЦП.
4. Складіть програму прийому даних з АЦП.
5. Укажіть способи перетворення аналогового сигналу в цифровий код і назад.
6. Поясніть структурну схему МК51, що наводиться нижче на ис. 3.8 і її роботу при збиранні інформації з аналогових датчиків.
7. Назвіть призначення виводів мікросхеми МК51, поданої на рис. 2.1, і поясніть схему підключення АЦП до МК51.
8. Поясніть структуру карти адресації бітів у резидентній пам’яті даних, подану на рис. 3.9.
9. Поясніть структуру карти адресації бітів у блоці регістрів спеціальних функцій, подану на рис. 3.10 і призначення регістрів спеціальних функцій.
10. Як проводиться компіляція програми і які повідомлення при цьому виводяться на екран?
11. Що таке емулятор?
12. Як можна активізувати емулятор та що він забезпечує?

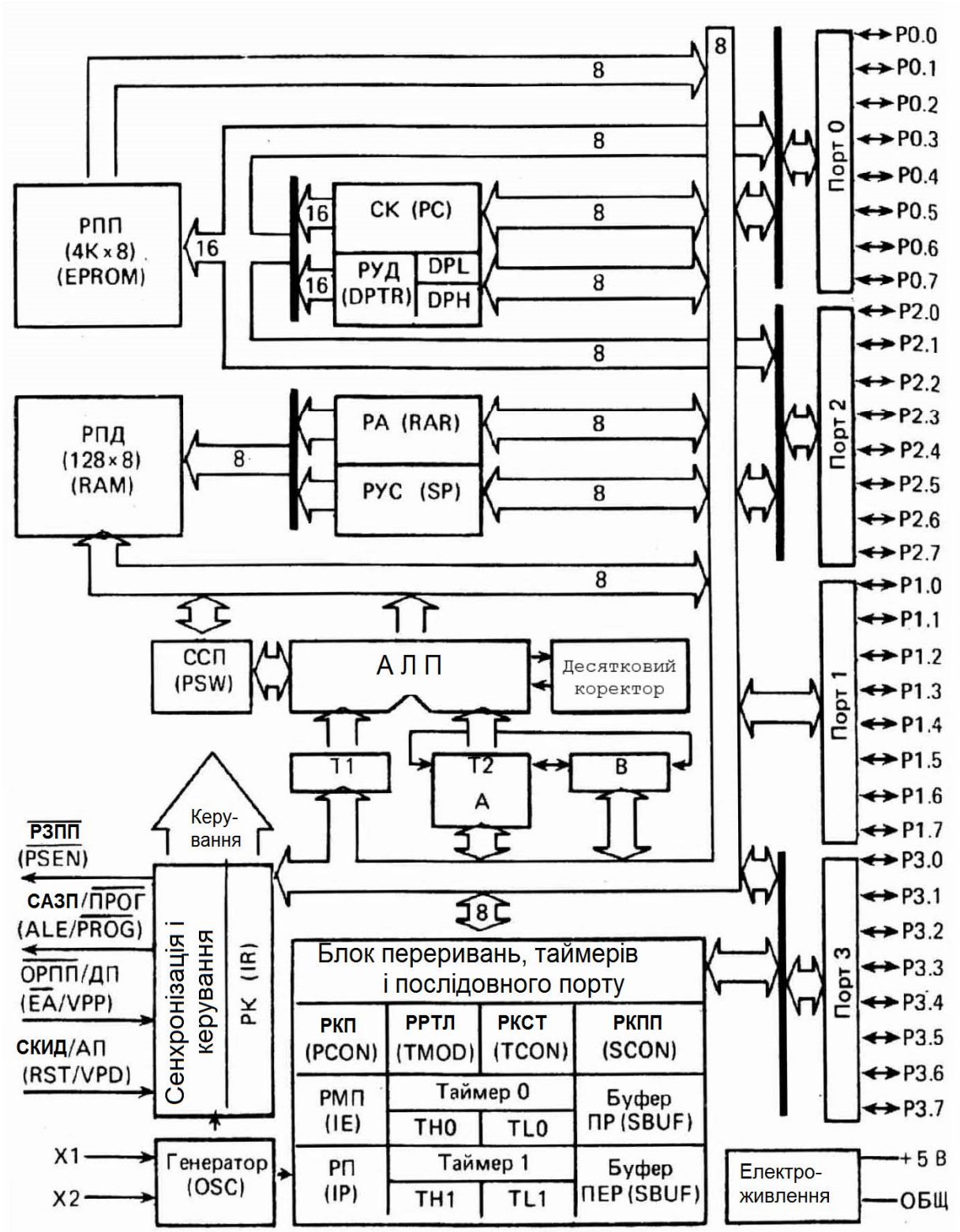


Рисунок 3.8 – Структурна схема МК51

Карти адресації бітів

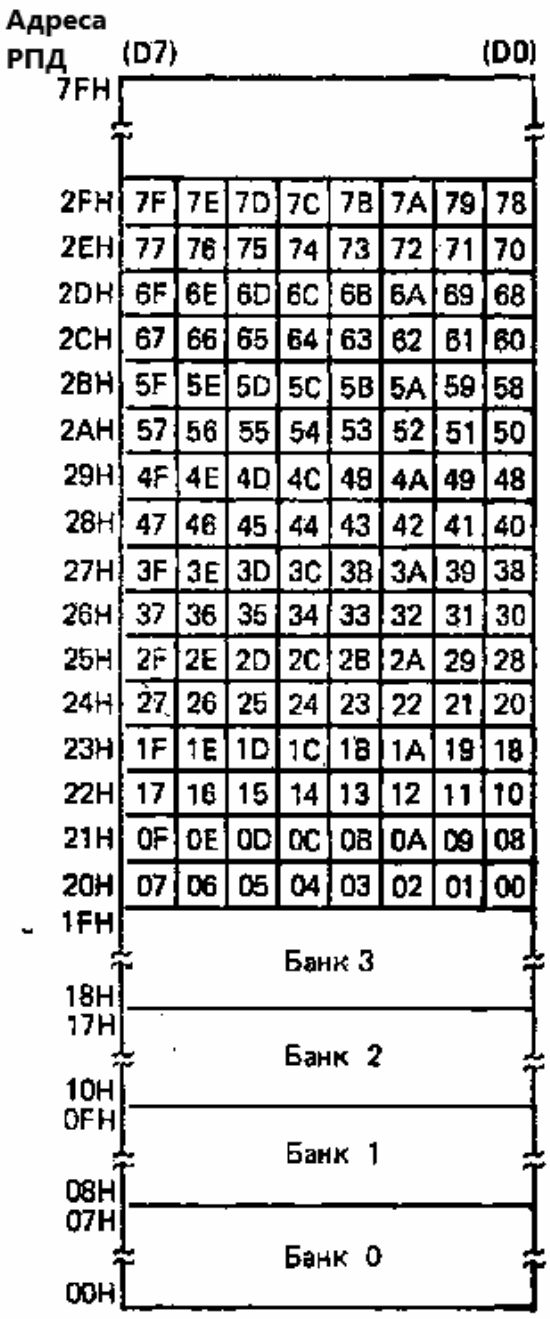


Рисунок 3.9 – Карта адресації бітів у резидентній пам’яті даних (РПД)

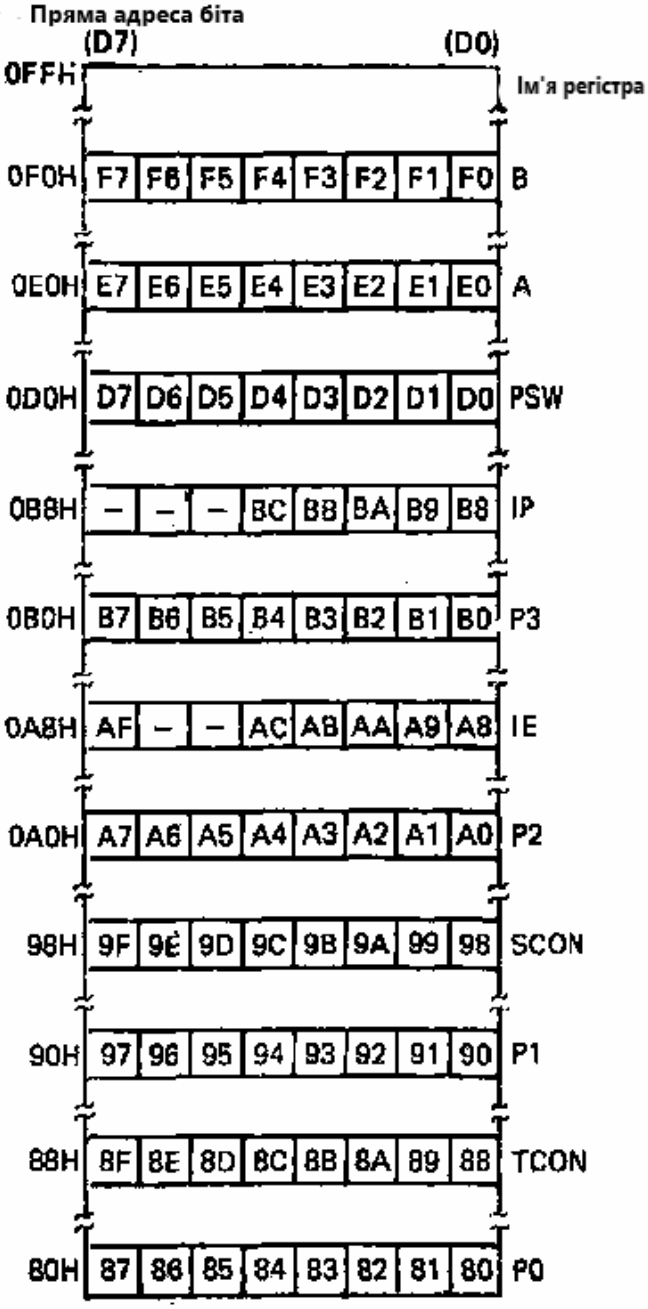


Рисунок 3.10 – Карта адресації бітів у блоці регістрів спеціальних функцій

3.5. Лабораторна робота 5.
Дослідження емулятора мікроконтролера МК 51

Мета роботи – ознайомлення зі структурою емулятора мікроконтролера МК 51, командами роботи з емулятором, режимами роботи емулятора, складанням, налагодженням і виконанням програм.

Порядок підготовки до роботи

- вивчити рекомендовану літературу [20, 25, 26] і методичні вказівки;
- ознайомитись з командами емулятора і програмним забезпеченням крос-засобів налагоджувача FD51, наведеними у лабораторній роботі № 3.

Опис лабораторної установки:

- ПЕОМ;
- програмне забезпечення емулятора МК 51.

Основні відомості

Програмне забезпечення емулятора мікроконтролера МК 51 дозволяє імітувати функціонування МК 51 за допомогою ПЕОМ, компіляцію програми, складеної мовою Асемблер, в об'єктний код і її налагодження.

У структуру програмного забезпечення входить редактор мови Асемблер (текстовий файл), компілятор програми в об'єктний код (файл ASM51.exe), перекодувальник в шістнадцятковий код (файл OH.exe), внутрішньосхемний емулятор МК51. Для запису пакета програм на ПЕОМ необхідна інсталяційна дискета, за допомогою якої програма інсталюється на комп'ютер.

Редактор мови Асемблер входить в оболонку NORTON COMMANDER (NC), що дозволяє у зручній формі записати програму мовою Асемблер. Для цього потрібно завантажити (запустити) NC. Після чого у лівій колонці можна проводити запис програми і її редагування.

У редакторі використовують такі клавіші і їхні комбінації:

– Home – у початок рядка; – End – у кінець рядка;
– PgUp – перегортати нагору; – PgDn – перегортати вниз;
– Ctrl-Home – у початок файлу; – Ctrl-End – у кінець файлу;
– Ctrl-PgUp – у початок екрана; – Ctrl-PgDn – у кінець екрана;
– Ctrl-Left – ролінг ; – Ctrl-Right – ролінг униз.

Для редагування введеного тексту використовуються клавіші:

- Del – видалення символу під курсором;
- BackSpace – видалення символу ліворуч від курсора. Якщо курсор перебуває у першій позиції рядка, то поточний рядок буде з'єднуватися з попереднім, але тільки у тому випадку, якщо довжина результуючого рядка не перевершує 128 знаків;
- Ins – перемикання режиму «вставка / заміщення». У цьому режимі при наборі символу в середині рядка символи, які стоять праворуч від курсору, зміщуються на одну позицію праворуч. У режимі заміщення символ, що стоїть над курсором, заміняється на введений.

Програма-Асемблер – це програма, що транслює вихідну програму в об'єктні коди. Вона виконує присвоєння дійсних адрес, перетворення чисел, присвоєння дійсних значень символьним змінним.

Програма на Асемблері складається:

- з команд;
- директив;
- керуючих параметрів.

Команди Асемблера – мнемонічне кодування дій, які повинен виконати процесор.

Директиви – псевдокоманди, які при трансляції не перетворюються у машинні коди команд і використовуються для визначення змінних, структури програми, завдання констант і т. д.

Керуючі параметри вказують транслятору на виконання певних дій.

Програма Асемблера ASM 51 дозволяє використати модульний принцип, тобто обробляти окремі частини складної програми – модулі.

Вихідна програма (файл name.a51) створюється за допомогою редактора текстових файлів і містить команди мовою Асемблера, управляючі директиви, параметри Асемблера й коментарі.

Після створення вихідної програми вона обробляється викликаною програмою Асемблера ASM 51.EXE (транслятором) за допомогою команди ASM 51 name.a51.

У результаті проробленої роботи буде створений:

- об'єктний файл, поданий у машинних кодах (форма вихідних даних асемблера виконується і являє собою абсолютний формат шістнад-

цяткового коду, файл якого може бути запрограмований на МК51 і включати необхідну інформацію для складання й налагодження);

- лістинговий файл (запис вихідної програми й об'єктного коду, в який Асемблер вносить повідомлення про всі помилки кодування й службову інформацію).

Асемблер має кілька директив, які дозволяють користувачеві встановити символьні імена (директива EQU), зарезервувати й ініціалізувати місце у пам'яті, керувати розміщенням програми.

Директиви керування: ORG – керування лічильником адреси (задає Асемблеру адресу комірки пам'яті, в якій повинна розміщатися наступна за нею команда прикладної програми); END – кінець програми (вказує на закінчення трансляції).

Налагоджувач асемблерних програм дозволяє:

- завантажити для налагодження HEX файл, вироблюваний трансляторами з мови Асемблера (крос-засобу), а також файли чистого двійкового коду, зчитані з ПЗП;
- переглянути на екрані у вікні налагоджувача-емулятора FD51 (рис 3.3) дизасембльований текст завантаженої програми, адреси й коди команд, область імітованого ОЗП даних, зовнішньої пам'яті, пам'яті програм, вміст усіх регістрів;
- виконати завантажену програму покроково, з переглядом результатів після кожного кроку, внести зміни у програму;
- внести зміни у вміст регістрів, прапорів, пам'яті і т. д.

Функціональні клавіші, які використовуються у цій роботі:

F1 – виконати поточну інструкцію завантаженої програми (поточною є підкреслена інструкція у вікні тексту програми);

F3 – дозволяє надавати числову інформацію у десяткових, двійкових, шістнадцяткових кодах для регістрів;

F4 – перемикає вікно пам'яті даних із внутрішньої на зовнішню;

F6 – перемикає форму подання пам'яті у вікні у двійкову й шістнадцяткову.

Команди налагоджувача:

- H – для одержання довідки;
- L [«тип пам'яті» «початкова адреса»] «файл.HEX» [/A] – завантажити файл у пам'ять.

Тип пам’яті: I (внутр.), E (зовнішня) і P (програмна).

Початкова адреса й тип пам’яті вказується тільки при завантаженні двійкового файлу.

/A – ключ тільки при завантаженні програми ISIS – II Macro Assembler.

R «номер регістра» = «число» – занести число у регістр поточного банку. Наприклад, R4 = FF.

«Ім’я регістра» = «число» – занести число у регістр спеціального призначення: A, B, TH0, TH1, TL0, TL1, DPL, DPH, DPTR, SP, IP, IE, TMOD, TCON, SCON, SBUF, PC. Наприклад, TH0 = FF.

«Ім’я прапора» = «число» – установити або зняти прапор (біт) у PSW.

Наприклад, S1 = 0.

RST – імітується скидання процесора;

QUIT – вихід в DOS;

N – очищення вмісту налагоджувача.

Порядок виконання роботи

1. Включити комп’ютер.

2. Виконати завантаження комп’ютера й NC і перейти у ліву панель NC.

Для цього утримуючи клавішу «ALT», натискаємо клавішу F1. У вікні дисків вибираємо клавішами «(» (уліво), «(» (вправо) диск C (жовтий колір на C).

3. На диску C вибираємо директорію Lab (C:\) (клавішами <(>,<(> приводимо курсор на директорію Lab і, нажавши клавішу «Enter», потрапимо у директорію C:\Lab.

4. Перебуваючи у директорії C:\Lab, створимо новий файл. Для цього натискаємо комбінацію клавіш «Shift» + «F4» і у вікні редактора, що з’явилося на екрані, вводимо ім’я нового файлу name.a51 (де name – прізвище студента до 7 символів), натискаємо «Enter» і у відкритому вікні нового файлу набираємо програму в мнемонічному коді, побудовану в лабораторній роботі. Для збереження змін у файлі використайте клавішу «F2».

5. Створену нову програму (файл name.a51) обробляємо програмою асемблер (файл ASM51.EXE). Для цього введемо у командному рядку

CMD> команду ASM51 name.a51. Результатом виконання програми буде об'єктний файл (name.obj), поданий у машинних кодах, а також лістинговий файл name.lst, що містить текст вихідної програми, її шістнадцятковий код, відомості про помилки (якщо вони є) і службову інформацію.

6. Отриманий файл name.obj необхідно перекодувати у шістнадцятковий файл, для чого використовуючи команду OH.EXE введемо у командний рядок OH name.obj і натиснемо «ENTER». Результатом виконання цієї команди буде файл у шістнадцятковому коді name.hex. Надалі виконання роботи ведеться саме із цим файлом (name.hex).

7. У директорії C:\LAB знаходимо файл налагоджувальника програм FD51.EXE і, нажавши клавішу «ENTER», запускаємо його.

8. У відкритому вікні налагоджувальника у вікні команд CMD> уводимо команду L name.hex і натискаємо «ENTER». У вікні програми буде виведено дизасембльовану програму.

9. Натискаючи клавішу «F1», крок за кроком виконаємо програму. Дійшовши до команди RET, зафіксуємо у звіті вміст акумулятора.

10. Вихід із програми налагоджувальника проводиться командою QUIT, що вводиться у вікні команд CMD>.

11. Результати роботи (файли з назвою name й з розширеннями a51, hex, lst, obj) по черзі записати у директорію GR № A або GR № B, використовуючи клавішу «F6».

Приклад виконання

З а в д а н н я. Скласти програму для розрахування значення функції $15x + 10$ у інтервалі від 5 до 20 з кроком 1. Результат розмістити у РПД із адреси 20H (у масив послідовно занести спочатку молодший, а потім старший байт результату).

```
        MOV R0,#20H;    (R0)←20H    Початкова адреса у РПД результату
        MOV R1 ,#05H;   (R1)←05H    Початкове значення x у R1
        MOV R2,#0FH;    (R2)←0FH    Множник 15
        MOV R3,#10H;    (R3)←10H    Кількість x від 5 до 20 із кроком 1
Ml:     MOV A,R1;       (A)←(R1)    Поточне значення x в акумулятор
        MOV B,R2;       (B)←R2      Множник 15 із R2 заноситься
                                    у розширник акумулятора B
```

MUL AB;	(AB)←(B)·(A)	множення А на В
ADD A,#0AH;	(A)←(A) + 0AH	додавання 10 до вмісту акумул.
MOV @R0,A;	((R0))←(A)	У РПД заноситься молодший байт результату
INC R0;	(R0) ← (R0) + 1	інкремент R0
MOV A,B;	(A) ← (B)	Старший байт в акумулятор
ADDC A,#00;	(A) ← (A) + 00	Облік переносу з молодшого байта
MOV @R0,A;	((R0)) ← (A)	У РПД заноситься старший байт результату
INC R0;	(R0) ← (R0)	+1 інкремент R0
INC R1;	(R1) ← (R1)	+1 інкремент R1
DJNZ R3, M1	(PC)←(PC) + 2, (R3) ← (R3) -1	Декремент регістра й перехід на M1, якщо не 0
END		Завершення програми

Модернізована програма

```
     MOV R0,#20H
Ml:  MOVA,R0
     SUBB A,#1BH
     MOV B,#0F
     RRC A
     MUL AB
     ADD A,#0AH
     MOV @R0,A
     INC R0
     MOV A,B
     ADDC A,#00
     MOV @R0,A
     INC R0
     CJNE R0,#40H,Ml
     END
```

Висновок: у процесі вдосконалення первісного варіанта програми були досягнуті такі результати: 1) обсяг займаної пам’яті зменшився на 2 байти; 2) кількість використовуваних у програмі регістрів зменшилася з 4 до 1.

Недоліком удосконалення є те, що час виконання програми збільшився на 33 мкс.

Зміст звіту:

- текст програми у мнемонічному й двійковому кодах;
- результат роботи програми;
- висновки.

Контрольні запитання

1. Призначення емулятора МК 51 і його структура.
2. Викласти порядок виконання лабораторної роботи.
3. Назвати команди роботи з емулятором.
4. Охарактеризувати режими роботи емулятора.
5. Складання програм і перевірка їхньої працездатності за допомогою емулятора.
6. Призначення налагоджувальника програм.
7. Призначення програми Асемблер.
8. Які клавіші використовуються у редакторі Асемблера?
9. Які функції виконує програма Асемблера ASM 51?
10. Що є результатом роботи програми Асемблера ASM 51?
11. Які функції можуть бути виконані за допомогою налагоджувальника програми ASM 51?
12. Назвати клавіші, що використовуються при роботі з цією лабораторною роботою.
13. Назвати команди налагоджувальника та їхню структуру:

- одержання довідки, завантаження на пам’ять, занесення числа у регістр поточного банку;
- занесення числа у регістр спеціального призначення: TLO, TL1, DPL, DPH, DPTR, SP, IP, IE, TMOD, TCON, SCON, SBUF, PC;
- установлення або скидання прапора (біт) у регістрі PSW;
- скидання процесора RSC;
- команда виходу в DOS–QUIT;
- очистка вмісту налагоджувальника.

14. Призначення директиви ORG.
15. Наведіть приклад запису програми мовою Асемблер.

3.6. Лабораторна робота 6.
Дослідження обчислень арифметичних виразів у мікроконтролері МК 51

Мета роботи – дослідити обчислення арифметичних виразів у мікроконтролері К1816ВЕ51 (МК51).

Порядок підготовки до роботи

– вивчити рекомендовану літературу, [20, 25, 26] і методичні вказівки;

– ознайомитися із системою команд мікроконтролера й основних їхніх функцій, звернути увагу на особливості складання й виконання програм мовою асемблер.

Опис лабораторної установки:

- ПЕОМ;
- програма емулятора мікроконтролера МК 51.

Основні відомості

При виконанні цієї роботи треба використовувати групу команд арифметичних операцій (табл. 2.7). Цю групу складають 24 команди що виконують операції додавання, десяткової корекції, інкремента / декремента байтів, віднімання, множення й ділення байтів.

Команди ADD і ADDC допускають додавання акумулятора з більшою кількістю операндів. Аналогічно командам ADDC існують чотири команди SUBB, що дозволяє більш просто, ніж у МК 48, робити віднімання байтів і багатобайтних двійкових чисел. У МК 51 реалізується розширений (у порівнянні із МК 48) список команд інкремента / декремента байтів, введена команда інкремента 16-бітного регістра-указівника даних (РУД).

Порядок виконання роботи

1. Включити комп’ютер і завантажити програму емулятора МК 51.

2. Відповідно до варіанта завдання скласти програму мовою асемблер.

3. Використовуючи програму емулятора МК 51, зробити налагодження та виконання програми й перевірити результати її виконання.

Приклад виконання роботи

З а в д а н н я. Розрахувати значення функції 3*x* + 15 в інтервалі від 5 до 20 із кроком 1. Результат розмістити у РПД із адреси 20Н (у масив послідовно занести спочатку молодший, а потім старший байт результату).

MOV R0,#20H;	(R0)←20H	Початкова адреса у РПД результату
MOV R1 ,#05H;	(R1)←05H	Початкове значення x у R1
MOV R2,#03H;	(R2)←0FH	Множник 3 заноситься у R2
MOV R3,#0FH;	(R3)←10H	Кількість *x* від 5 до 20 з кроком 1
Ml: MOV A,R1;	(A)←(R1)	Поточне значення *x*
MOV B,R2;	(B)←R2	Множник 3 заноситься у розширник акумулятора B
MUL AB;	(AB)←(B)·(A)	множення вмісту A на вміст B
ADD A,#0AH;	(A) ← (A) + 0AH	Додовання 0AH до вмісту акумул.
MOV @R0,A;	((R0)) ← (A)	У РПД заноситься молодший байт результату
INC R0;	(R0) ← (R0) + 1	Інкремент R0
MOV A,B;	(A) ← (B)	Старший байт
ADDC A,#00;	(A) ← (A) + 00	Облік перенесення з молодшого байта
MOV @R0,A;	((R0)) ← (A)	У РПД заноситься старший байт результату
INC R0;	(R0) ← (R0) +1	інкремент R0
INC R1;	(R1) ← (R1) +1	інкремент R1
DJNZ R3,M1	(PC) ← (PC) + 2, (Rn)←(Rn)-1	Декремент регістра й перехід на, M1 якщо не 0
END		Завершення програми

Модернізована програма

```
      MOV R0,#20H
Ml:   MOVA,R0
      SUBB A,#1BH
      MOV B,#0F
      RRC A
      MUL AB
      ADD A,#0AH
      MOV @R0,A
      INC R0
      MOV A,B
      ADDC A,#00
      MOV @R0,A
      INC R0
      CJNE R0,#40H,Ml
      END
```

Висновок: досліджено застосування команди CJNE у мікроконтролері К1816ВЕ51.

Час роботи вдосконаленої програми склав 264 мкс, що на 4 мкс швидше, ніж час роботи початкової програми. Обсяг зайнятої пам'яті також зменшився.

Варіанти завдань

Номер варіанта вибирається відповідно до номера залікової книжки.

Варіант 1. Розрахувати значення функції $15x + 10$ у інтервалі від 5 до 20 із кроком 1. Результат розмістити у РПД із адреси 20Н (у масив послідовно занести спочатку молодший, а потім старший байт результату).

Варіант 2. Розрахувати значення функції $3x + 15$ у інтервалі від 10 до 100 із кроком 10. Результат розмістити у РПД із адреси 20Н (у масив послідовно занести спочатку молодший, а потім старший байт результату).

Варіант 3. Розрахувати значення функції $5x + 50$ у інтервалі від 0 до 20 із кроком 2. Результат розмістити у РПД із адреси 30Н.

Варіант 4. Скласти програму віднімання чотирибайтових беззнакових чисел. Перше число перебуває у РПД за адресою 20–23Н, друге – за адресою 28–31Н. Результат помістити на місце першого операнда.

Варіант 5. Масив чисел був архівований і поміщений у новий масив, у якому попередній елемент вказує число, а наступна кількість повторень цього числа у вихідному масиві. У результуючому масиві описано 8 чисел. Знайти суму членів вихідного масиву. Результат розмістити у регістрах R3, R4, R5.

Варіант 6. Розрахувати 16 значень функції $Y = 250 / x$ для x, що починається з 10 із кроком 8. Результати округлити до цілого значення й розмістити в РПД із адреси 20H.

Варіант 7. Перевести однобайтовий шістнадцятковий операнд у двійково-десятковий упакований формат. Вихідний операнд перебуває у регістрі R5. Результат розмістити в регістрах R4 (число сотень) і R5 (десятки, одиниці).

Варіант 8. У РПД, починаючи з адреси 20H, перебуває масив з 16 елементів. Підрахувати кількість елементів масиву, що потрапили в інтервал від 50 до 100. Результат запам'ятати у регістрі R2.

Варіант 9. У РПД, починаючи з адреси 20H, перебуває масив з 16 чисел. Знайти максимальний елемент масиву й помістити у регістр R2 його значення, а у регістр R3 його адресу.

Варіант 10. У регістрі R5 перебуває двійково-десятковий операнд. Перевести операнд у шістнадцяткове значення й помістити у R5.

Зміст звіту

- текст програми та її алгоритм;
- результат роботи програми й висновки.

Контрольні запитання

1. Загальна характеристика групи команд арифметичних операцій.
2. Назвати операції додавання, навести приклади, вказати алгоритм і коментарі.
3. Охарактеризувати команди ADDC і CDA.
4. Навести команди віднімання.
5. Пояснити команди інкремента: акумулятора, регістра, прямої адресації байта, біта у РПД, указівника даних.
6. Навести команди декремента з алгоритмом і коментарем: акумулятора, регістра, прямої адресації байта у РПД.

3.7. Лабораторна робота 7. Дослідження виконання логічних функцій, що оперують із бітами даних

Мета роботи – дослідити виконання логічних функцій, що оперують із бітовими даними, у мікроконтролері К1816ВЕ51.

Порядок підготовки до роботи

- вивчити рекомендовану літературу [20, 25, 26];
- ознайомитись із системою команд виконання логічних операцій (табл. 3.4) і основними їхніми функціями, звернути увагу на особливості складання й виконання програм мовою Асемблер.

Опис лабораторної установки:

- ПЕОМ;
- програма емулятора мікроконтролера МК51.

Основні відомості

Емулятор МК51 дозволяє проводити дослідження виконання логічних функцій, що використовують бітові дані. З цією метою у системі команд МК51 передбачені команди для роботи з бітами.

Відмінною рисою цієї групи команд є те, що вони оперують із однобітними операндами. Як такі операнди можуть виступати окремі біти деяких регістрів спеціальних функцій і портів, акумулятора й розширника, а також 128 програмних прапорів користувача. Докладний опис спеціальних регістрів наведено у [20].

Існують команди скидання (CLR), установки (SETB) та інверсії (CPL) біта, а також кон'юнкції й диз'юнкції біта й прапора перенесення. Команди операцій з бітами наведені у табл. 3.7. Для адресації біта використовується пряма 8-бітна адреса (bit). Непряма адресація біта неможлива.

Відзначені регістри (табл. 3.8) допускають адресацію окремих бітів.

Група команд логічних операцій

Дану групу утворюють 25 команд (табл. 2.7), що реалізують ті ж логічні операції над байтами, що й у МК48. Однак у МК51 значно розширена кількість типів операндів, що беруть участь в операціях.

На відміну від МК48 є можливість робити операцію «виключаюче АБО» із вмістом портів. Команда XRL може бути ефективно використана для інверсії окремих бітів-портів.

Таблиця 3.7 – Команди операцій з бітами

Мнемокод	КІП	ТК	Б	МЦ	Коментар
CLR C	11000011	1	1	1	C ← 0; скидання біта C
SETB C	11010011	1	1	1	C ← 1; установка біта C
CPL C	10110011	1	1	1	інверсія біта C у біт C
CLR bit	11000010	4	2	1	(bit) ← 0; скидання прямоадресованого біта
SETB bit	11010010	4	2	1	(bit) ← 1; установка прямоадресованого біта
CPL bit	10110010	4	2	1	Інверсія біта
ANL C, bit	10000010	4	2	2	Логічне «І» біта та біта C у C
ANL C, /bit	10110000	4	2	2	Логічне «І» інверсії біта та біта C у C
ORL C, bit	01110010	4	2	2	Логічне «АБО» біта та біта C у C
ORL C, /bit	10100000	4	2	2	Логічне «АБО» інверсії біта та біта C у C
MOV C, bit	10100010	4	2	1	Пересилання із прямоадресованого біта у біт C
MOV bit, C	10010010	4	2	2	Пересилання із біта C у прямоадресований біт

Таблиця 3.8 – Блок регістрів спеціальних функцій (SFR)

Адреса	Символ	Найменування
Dir		
0E0H	* ACC	Акумулятор
0F0H	* B	Регістр-розширник акумулятора
0D0H	* PSW	Слово стану процесора
0B0H	* P3	Порт 3
0A0H	* P2	Порт 2
090H	* P1	Порт 1
080H	* P0	Порт 0
0B8H	* IP	Регістр пріоритетів переривань
0A8H	* IE	Регістр маски переривань
99H	SBUF	Буфер послідовного прийомопередатчика
98H	* SCON	Регістр керування / статусу послідовного порту
89H	TMOD	Регістр режимів таймерів / лічильників
88H	* TCON	Регістр керування / статусу таймерів
8DH	TH1	Таймер 1 (старший байт)
8BH	TL1	Таймер 1 (молодший байт)
8CH	TH0	Таймер 0 (старший байт)
8AH	TL0	Таймер 0 (молодший байт)
83H	DPTR	Регістр-указівник даних (DPH) (старший байт)
82H	-"-	Регістр-указівник даних (DPL) (молодший байт)
81H	SP	Регістр-указівник стека
87H	PCON	Регістр керування потужністю споживання

Порядок виконання роботи

Дослідити виконання логічних функцій, що оперують із бітовими даними.

Скласти відповідно до варіанта завдання програму, що реалізує Булеву функцію чотирьох змінних. Вихідними значеннями є:

- A – 3 біт акумулятора;
- B – 5 біт середовища РПД за адресою 30H;
- C – 7 біт порту P0;
- D – прапор перенесення.

Приклад виконання програми

З а в д а н н я . Скласти програму, що реалізує Булеву функцію чотирьох змінних. Вихідними значеннями є: A – 3 біт акумулятора; B – 5 біт комірки РПД за адресою 30H; C – 7 біт порту P0; D – прапор перенесення.

Обчислити значення логічної функції (AVB)&(CVD).

```
RL A;          3 біт акумулятора переходить у 4
RL A;          4 біт акумулятора переходить у 5
ORL A, 30H;    A V B
ORL C, P0.7;   C V D
ANL C, ACC.5;  C & ACC.5
END
```

Висновок. Проведено дослідження виконання логічних функцій, що оперують із бітовими даними, у мікроконтролері К1816ВЕ51.

Спроба вдосконалення програми приводила до гірших показників часу роботи й займаної пам’яті. Час роботи програми склав 12 мкс.

Варіанти завдань

Варіант 1. Обчислити значення логічної функції (A \/ B) & (C \/ D).
Варіант 2. Обчислити значення логічної функції (A & C) \/ (B & D).
Варіант 3. Обчислити значення логічної функції A & (B \/C) &D.
Варіант 4. Обчислити значення логічної функції (A \/ C \/ D) & B.
Варіант 5. Обчислити значення логічної функції ((A &D) \/ C) & B.
Варіант 6. Обчислити значення логічної функції (A \/ B) \/ (C \/ D).
Варіант 7. Обчислити значення логічної функції A \/ B \/ C \/ D.

Варіант 8. Обчислити значення логічної функції A & B \/ C \/ D.
Варіант 9. Обчислити значення логічної функції A\/ B & C \/ D.
Варіант 10. Обчислити значення логічної функції A & B & C & D.

Зміст звіту:

- текст програми та її алгоритм;
- результат роботи програми й висновки.

Контрольні запитання

1. Загальна характеристика команд логічних операцій за їхньої особливості при роботі з різними операндами.
2. Які можуть бути операнди у командах логічних операцій?
3. Назвіть команди, що оперують із бітами та можливі види адресації біт.
4. Наведіть формат, алгоритм і коментарі до команд логічних операцій:

- логічне І акумулятора й регістра, акумулятора й прямої адресації бай-та, акумулятора й байта з RPD, акумулятора й константи, прямої адресації байта й акумулятора, прямої адресації байта й константи;
- логічне АБО акумулятора й регістра, акумулятора й прямої адресації байта, акумулятора й байта з RPD, акумулятора й константи, прямої адресації байта й акумулятора, прямої адресації байта й константи;
- логічне виключаюче АБО акумулятора й регістра, акумулятора й прямої адресації байта, акумулятора й байта з RPD, акумулятора й константи, прямої адресації байта й акумулятора, прямої адресації байта й константи.

5. Наведіть формат, алгоритм і коментарі до команд: скидання акумулятора, інверсії акумулятора, циклічний зсув акумулятора вліво й вправо, зсув акумулятора вліво й вправо через перенесення і команди обміну місцями тетрад в акумуляторі.
6. Умовний і безумовний перехід команд передачі керування.
7. Наведіть формат, алгоритм, і коментарі до команд: довгого переходу й довгого виклику підпрограми, абсолютного переходу в межах одного рядка пам’яті, а також непрямого переходу, відносного переходу.

3.8. Лабораторна робота 8.
Дослідження застосування команд умовних переходів

Мета роботи – дослідити застосування команд умовних переходів у мікроконтролері К1816ВЕ51 (МК 51).

Порядок підготовки до роботи

Вивчити рекомендовану літературу [20, 25, 26] методичні вказівки.

Ознайомитися із системою команд МК 51 і основними їхніми функціями, звернути увагу на особливості складання й виконання програм мовою Асемблер із застосуванням команд умовних переходів.

Опис використовуваного устаткування

- ПЕОМ;
- програма емулятор мікроконтролера МК 51.

Основні відомості

Група команд передачі керування

До цієї групи команд (табл. 3.9) належать команди, що забезпечують умовне й безумовне розгалуження, виклик підпрограм і повернення з них, а також команда порожньої операції NOP. У більшості команд використовується пряма адресація, тобто адреса переходу цілком (або його частина) утримується у самій команді передачі керування. Можна виділити три різновиди команд розгалуження за розрядністю адреси переходу, що вказується.

Довгий перехід. Перехід по всьому адресному просторі ПП. У команді утримується повна 16-бітна адреса переходу (ad 16). Трибайтні команди довгого переходу містять у мнемокоді букву L (Long). Усього існує дві такі команди: LJMP – довгий перехід і LCALL – довгий виклик підпрограми. На практиці рідко виникає необхідність переходу в межах усього адресного простору й частіше використовуються вкорочені команди переходу, що займають менше місця у пам’яті.

Абсолютний перехід. Перехід у межах однієї сторінки пам’яті програм розміром 2048 байт. Такі команди містять тільки 11 молодших бітів адреси переходу (ad 11). Команди абсолютного переходу мають формат 2 байти. Початкова буква мнемокода – A (Absolute). При виконанні команди в обчисленій адресі наступної за чергою команди ((PC)= (PC) + 2) 11 молодших біт заміняються на ad11 з тіла команди абсолютного переходу.

Таблиця 3.9 – Група команд передачі керування

Назва команди	Мнемокод	Т	Б	Ц	Операція
Довгий перехід у повному обсязі пам'яті програм	LJMP ad 16	12	3	2	(PC) ← ad 16
Абсолютний перехід усередині сторінки в 2 Кбайта	AJMP ad 11	6	2	2	(PC) ← (PC) + 2, ($PC_{0\text{-}10}$) ← ad 11
Короткий відносний перехід усередині сторінки в 256 байт	SJMP rel	5	2	2	(PC) ← (PC) + 2, (PC) ← (PC) + rel
Непрямий відносний перехід	JMP @A+ DPTR	1	1	2	(PC) ← (A) + (DPTR)
Перехід, якщо акумулятор дорівнює нулю	JZ rel	5	2	2	(PC) ← (PC) + 2, якщо (A) = 0, то (PC) ← (PC) + rel
Перехід, якщо акумулятор не дорівнює нулю	JNZ rel	5	2	2	(PC) ← (PC) + 2, якщо (A) ≠ 0, то (PC) ← (PC) + rel
Перехід, якщо перенесення дорівнює одиниці	JC rel	5	2	2	(PC) ← (PC) + 2, якщо (C) = 1, то (PC) ← (PC) + rel
Перехід, якщо перенесення дорівнює нулю	JNC rel	5	2	2	(PC) ← (PC) + 2, якщо (C) ≠ 0, то (PC) ← (PC) + rel
Перехід, якщо біт дорівнює одиниці	JB bit, rel	11	3	2	(PC) ← (PC) + 3, якщо (b) = 1, то (PC) ← (PC) + rel
Перехід, якщо біт дорівнює нулю	JNB bit, rel	11	3	2	(PC) ← (PC) + 3, якщо (b) ≠ 0, то (PC) ← (PC) + rel
Перехід, якщо біт установлений, з наступним скиданням біта	JBC bit, rel	11	3	2	(PC) ← (PC) +3, якщо (b) = 1, то (b) ← 0 і (PC) ← (PC) + rel
Декремент регістра й перехід, якщо не нуль	DJNZ Rn, rel	5	2	2	(PC) ← (PC) +2, (Rn) ← (Rn) – 1, якщо (Rn) ≠ 0, то (PC) ← (PC) + rel
Декремент прямоадресованого байта й перехід, якщо не нуль	DJNZ ad, rel	8	3	2	(PC) ← (PC) + 2, (ad) ← (ad) – 1, якщо (ad) ≠ 0, то (PC) ← (PC) + rel
Порівняння акумулятора із прямоадресованим байтом і перехід, якщо не дорівнює	CJNE A, ad, rel	8	3	2	(PC) ← (PC) + 3, якщо (A) ≠ # (ad), то (PC) ← (PC) + rel, якщо (A) < (ad), то (C) ← 1, інакше (C) ← 0

Закінчення табл. 3.9

Назва команди	Мнемокод	Т	Б	Ц	Операція
Порівняння акумулятора з константою й перехід, якщо не дорівнює	CJNE A, # d, rel	10	3	2	(PC) ← (PC) + 3, якщо (A) ≠# d, то (PC) ← (PC) + rel, якщо (A) < # d, то (C) ←1, інакше (C) ← 0
Порівняння регістра з константою й перехід, якщо не дорівнює	CJNE Rn, # d, rel	10	3	2	(PC) ← (PC) +3, якщо (Rn) ≠# d, то (PC) ← (PC) + rel, якщо (Rn) < # d, то (C) ←1, інакше (C) ← 0
Порівняння байта в РПД із константою й перехід, якщо не дорівнює	CJNE @Ri, #d,rel	10	3	2	(PC) ← (PC) + 3, якщо ((Ri)) # d, то (PC) ← (PC) + rel, якщо ((Ri)) < # d, то (C) ← 1, інакше (C) ← 0
Довгий виклик підпрограми	LCALL ad 16	12	3	2	(PC) ← (PC) + 3, (SP) ← (SP) + 1, ((SP)) ← (PC_{0-7}), (SP) ← (SP) + 1, ((SP)) ← (PC_{8-15}), (PC) ← ad 16
Абсолютний виклик підпрограми у межах сторінки в 2 Кбайта	ACALL ad 11	6	2	2	(PC) ← (PC) + 2, (SP) ← (SP) + 1, ((SP)) ← (PC_{0-7}), (SP) ← (SP) + 1, ((SP)) ← (PC_{8-15}), (PC_{0-10}) ←ad ll
Повернення з підпрограми	RET	1	1	2	(PC_{8-15}) ← ((SP)), (SP) ← (SP) – 1, (PC_{0-7}) ← ((SP)), (SP) ← (SP) – 1
Повернення з підпрограми оброблення переривання	RETI	1	1	2	(PC_{8-15}) ← ((SP)), (SP) ← (SP) – 1, (PC_{0-7}) ← ((SP)), (SP) ← (SP) – 1
Нема операції	NOP	1	1	1	(PC) ← (PC) + 1

Відносний перехід. Короткий відносний перехід дозволяє передати керування у межах −128…+127 байт щодо адреси наступної команди (команди, що випливають по черзі за командою відносного переходу). Існує одна команда безумовного короткого переходу SJMP (Short). Усі команди умовного переходу використовують цей метод адресації. Відносна адреса переходу (rel) утримується у другому байті команди.

Непрямий перехід. Команда JMP @A+DPTR дозволяє передавати керування за непрямою адресою. Ця команда зручна тим, що надає можливість організації переходу за адресою, яка обчислюється самою програмою й невідома при написанні вихідного тексту програми.

Умовні переходи. Розвинена система умовних переходів надає можливість здійснювати розгалуження за такими умовами: акумулятор містить нуль (JZ); вміст акумулятора не дорівнює нулю (JNZ); перенесення дорівнює одиниці (JC); перенесення дорівнює нулю (JNC); адресований біт дорівнює одиниці (JB); адресований біт дорівнює нулю (JNB).

Для організації програмних циклів зручно користуватися командою DJNZ, що працює аналогічно відповідній команді МК 48. Однак як лічильник циклів у МК 51 може використовуватися не тільки регістр, але й прямоадресований байт (наприклад, комірка РПД). Команда CJNE ефективно використовується у процедурах очікування якої-небудь події.

Усі команди цієї групи, за винятком CJNE і JBC, не впливають на прапори. Команда CJNE установлює прапор C, якщо перший операнд виявляється менше другого. Команда JBC скидає прапор C у випадку переходу.

Примітка: Асемблер допускає використання узагальненого імені команд JMP і CALL, які у процесі трансляції заміняються оптимальними за форматом командами переходу (AJMP, SJMP, LJMP) або виклику (ACALL, LCALL).

Порядок виконання роботи

Скласти програму відповідно до наведених варіантів і провести її налагодження.

Приклад написання і виконання програми

Завдання. Створити в РПД, починаючи з адреси 40H, масив з 10 чисел, елементами якого є квадрати чисел (від 0 до 15), прочитаних з порту P1. Таблиця квадратів чисел розташована у ПП, починаючи з адреси 100H.

	MOV R0, #40H;	(R0) ← 40H	Початкова адреса у РПД результату
	MOV R1, #0AH;	(R1) ← 0AH	Загрузка константи у R1
	MOV DPTR, #0100H;	(DPTR) ← 0100H	Початкова адреса у ПП таблиці квадратів чисел
Ml:	MOV A, P1;	(A) ← (P1)	Поточне прочитане з порту число в акум.
	MOVC A, @A + DPTR;	(A) ← ((A) + (DPTR))	Пересилання в акум. байта з пам'яті програм
	MOV @R0, A;	((R0)) ← (A)	Пересилання в акум. байта у РПД
	INC R0;	(R0) ← (R0) + 1	Інкремент R0
	DJNZ R1, M1;	(PC) ← (PC) + 2, (R1)←(R1)-1, якщо(R1) ≠ 0, то PC) ← (PC) + M1	Декремент R1 і перехід, якщо не нуль, на мітку M1
	END		Кінець програми

Висновки. Проведено дослідження роботи команд умовних переходів з використанням табличних даних при програмуванні контролера К1816ВЕ51.

Варіанти завдань

Варіант 1. У РПД, починаючи з адреси 20Н, перебуває масив з 16 елементів. Підрахувати й зберегти у регістрах: R2 – кількість елементів масиву, менші значення 128; R3 – кількість елементів масиву, рівних 128; R4 – кількість елементів масиву, більших 128.

Варіант 2. У РПД із адреси 20Н перебуває масив, що складається з 16 елементів. Підсумувати елементи масиву доти, поки значення суми не перевищить 512. Видати у R3 номер елемента, на якому відбулося переповнення. Якщо сума елементів не досягла значення 512, то видати у регістрі R3 значення 0.

Варіант 3. Для функції $Y = 20x + 45$ видати у R2 перше значення аргументу, при якому значення функції перевищить 1024. Початкове значення аргументу $x = 10$.

Варіант 4. У РПД із адреси 20H перебуває масив з 16 чисел. Елементами масиву є числа 32, 64, 96 і 128. Підрахувати й зберегти у регістрах R4–R7 кількість повторень кожного елемента.

Варіант 5. У РПД за адресами 20H-2FH перебуває масив. З адреси 30H створити масив, у який входять адреси елементів першого масиву, рівних 128. У регістрі R2 зберегти число елементів, що дорівнює 128. Перервати виконання програми, якщо буде знайдено 5 елементів зі значенням 128.

Варіант 6. У РПД із адрес 20H і 30H перебувають 2 масиви, що складаються з 16 елементів кожен. Підрахувати кількість елементів першого масиву, які дорівнюють значенням елементів в 2 масиві. Результат занести у регістр R2.

Варіант 7. Для функції $Y = 40x + 10$ одержати перше значення, що перевищує 512, починаючи з $x = 1$. Значення аргументу записати у R4, функції – у R5, R6.

Варіант 8. У ЗПД, починаючи з адреси 100H, перебуває масив з 10 елементів. Одержати у регістрі R3 кількість елементів, що дорівнює 55H. Рахунок перервати, якщо кількість елементів перевищить на 3.

Варіант 9. Для функції $15x + 85$ знайти перше значення аргументу, при якому молодший байт функції дорівнює 155.

Варіант 10. У ВПД із адреси 300H перебуває масив з 15 чисел. Елементами масиву є числа 10, 20, 30 і 180. Підрахувати й зберегти у регістрах R4–R7 кількість повторень кожного елемента.

Зміст звіту:

- текст програми у мнемокоді й результати її роботи;
- алгоритм програми;
- висновки.

Контрольні запитання

1. За допомогою якої команди можна встановити початкову адресу в пам’яті програм?
2. Поясніть алгоритми команди DJNZ R1, M1, CJNE R0, M2.

3. Як установити початкову адресу в резидентній пам’яті даних?

4. Як здійснити інкремент вмісту регістра?

5. Складіть і поясніть алгоритм відповідно до заданого варіанта.

6. Складіть програму створення у РПД масиву з 5-ти чисел (від 3 до 7).

3.9. Лабораторна робота 9. Дослідження роботи портів введення-виведення при асинхронному прийомі/передачі інформації

Мета роботи – дослідити роботу портів введення-виведення при асинхронному прийомі-передачі інформації у мікроконтролері К1816ВЕ51.

Порядок підготовки до роботи

- вивчити рекомендовану літературу [20, 25, 26] і методичні вказівки;
- ознайомитися з роботою портів введення-виведення, основними їхніми

функціями й усвідомити особливості складання й виконання програм на мові Асемблер.

Опис використовуваного устаткування

- ПЕОМ;
- програма емулятора мікроконтролера МК51.

Основні відомості

Усі чотири порти МК51 призначені для введення або виведення ін-формації побайтно. Кожен порт містить керований регістр-защіпку, вхідний буфер і вихідний драйвер. Вихідні драйвери портів 0 і 2, а також вхідний буфер порту 0 використовується при звертанні до зовнішньої пам’яті (ЗП). При цьому через порт 0 у режимі тимчасового мультиплексування спочатку виводиться молодший байт адреси ЗП, а потім видається або приймається байт даних. Через порт 2 виводиться старший байт адреси у тих випадках, коли розрядність адреси дорівнює 16 біт.

Порт 0 є двонаправленим, а порти 1, 2 і 3 – квазі-двонаправленими. Кожна лінія портів може бути використана незалежно для введення або виведення інформації. Для того щоб деяка лінія порту використовувалася

для введення, у D-тригер регістра-защіпки порту повинна бути записана 1, що закриває МОП – транзистор вихідного ланцюга.

За сигналом СБР у регістри-защіпки всіх портів автоматично записуються одиниці, що настроюють їх тим самим на режим введення.

Усі порти можуть бути використані для організації введення-виведення інформації по двонаправленим лініям передачі. Однак порти 0 і 2 не можуть бути використані для цієї мети у випадку, якщо МК система має зовнішню пам'ять (ЗП), зв'язок з якою організується через загальну поділювану шину адрес даних, що працює у режимі тимчасового мультиплексування.

Запис у порт. При виконанні команди, що змінює вміст регістра-защіпки порту, нове значення фіксується у регістрі у момент S6P2 останнього циклу команди. Однак опитування вмісту регістра-защіпки вихідною схемою здійснюється під час фази P1 і, отже, новий вміст регістра-защіпки з'являється на вихідних контактах порту тільки у момент S1P1 наступного машинного циклу.

Навантажувальна здатність портів. Вихідні лінії портів 1, 2 і 3 можуть працювати на одну ТЛЛ-схему. Лінії порту 0 можуть бути навантажені на два входи ТЛЛ-схем кожна. Лінії порту 0 можуть працювати і на n-МОП-схеми, однак при цьому їх необхідно підключати на джерело електроживлення через зовнішні навантажувальні резистори, за винятком случаю, коли шина порту 0 використовується як шина адрес-даних зовнішньої пам'яті.

Вхідні сигнали для МК 51 можуть формуватися ТЛЛ-схемами або n-МОП-схемами. Припустимо використання як джерела сигналів для МК 51 схем з відкритим колектором або відкритим стоком. Однак при цьому час зміни вхідного сигналу при переході з 0 у 1 виявиться сильно затягнутим.

Особливості роботи портів. Звертання до портів введення-виведення можливо з використанням команд, що оперують із байтом, окремим бітом і з використанням команд, і довільною комбінацією біт. При цьому в тих випадках, коли порт є одночасно операндом і місцем призначення результату, пристрій керування автоматично реалізує спеціальний режим, що називається «читання – модифікація – запис». Цей режим обігу припускає введення сигналів не із зовнішніх виведень порту, а з його

регістра-защіпки, що дозволяє виключити неправильне зчитування раніше виведеної інформації. Подібний механізм звертання до портів реалізований у таких командах:

ANL – логічне І, наприклад, ANL P1,A;

ORL – логічне АБО, наприклад, ORL P2,A;

XRL – виключаюче АБО, наприклад, XRL P3,A;

JBC – перехід, якщо в адресованому біті одиниця й наступне скидання біта, наприклад, JBC P1.1, LABEL;

CPL – інверсія біта, наприклад, CPL P3.3;

INC – інкремент біта, наприклад, INC P2;

DEC – декремент порту, наприклад, DEC P2;

DJNZ – декремент порту й перехід, якщо його вміст не дорівнює нулю, наприклад, DJNZ P3, LABEL;

MOV PX.Y, C – передача біта перенесення у біт Y порту X;

SET PX.Y – установка біта Y порту X;

CLR PX.Y – скидання біта Y порту X;

За цими командами спочатку зчитується байт із порту, а потім записується новий байт у регістр-защіпку.

Два МК обмінюються даними в асинхронному режимі. Дані передаються й приймаються через порти P0 і P2, а керуючі сигнали й біт контролю на парність, (якщо цей контроль існує) – по лініях порту P1. Приймання – передача ведеться під керуванням 5 сигналів:

- «ГОТОВИЙ» – сигнал виставляється мікроконтролером-приймачем і свідчить про готовність мікроконтролера до прийому нової порції даних;
- «ДАНІ НА ШИНІ» – сигнал виставляється мікроконтролером приймачем. Свідчить про те, що нові дані виставлені на шини портів;
- «ПОМИЛКА» – сигнал виставляється МК-приймачем. Свідчить про те, що при перевірці прийнятих даних виявлена помилка й необхідно повторити передачу;
- «ПІДТВЕРДЖЕННЯ ПОМИЛКИ» – сигнал виставляється МК-джерелом і свідчить про те, що МК одержав сигнал «Помилка» і у новому циклі старі дані будуть виставлені знову на шини;
- «КІНЕЦЬ ПЕРЕДАЧІ» – сигнал виставляється МК-джерелом і свідчить про те, що необхідна інформація передана повністю.

Дані для передачі перебувають у РПД МК. Поточна адреса передаваного байта перебуває у регістрі R0. Прийняті дані розташовуються у РПД МК, починаючи з адреси, зазначеної у регістрі R0. При складанні програми необхідно враховувати, що при виконанні обміну даних всі переривання дозволені й МК може перейти до підпрограми обслуговування переривання.

При прийманні – передачі інформації здійснюється контроль на парність. Біт парності передається через нульовий біт порту P1.

Порядок виконання роботи

З а в д а н н я. Скласти програму роботи МК з передачі інформації й виконати її.

Приклад виконання програми

```
ORG 00H                 установлення початкової адреси
ORL IE, #85H            дозвіл переривань
ORL TCON, #05H          установлення виду переривання:
                        по фронту
MOV R0, #20H            установлення початкової адреси
                        переданої інформації
M1:  GET P1.1           видача сигналів М: «ГОТОВИЙ»
JNB P1.2, M             очікування сигналу
                        «ДАНІ НА ШИНІ»
MOV B, P0               пересилання даних у старший байт
SETB P1.1               видача сигналу «ГОТОВИЙ»
     W:  JNB P1.2,W         очікування сигналу
                             «ДАНІ НА ШИНІ»
MOV A, P0               пересилання у молодший байт
     CPL A                  інверсія молодшого байта
     XRL A, B               порівняння молодшого й
                            старшого байтів
JNZ ERR                 молодший байт не дорівнює 0,
                        перехід по мітці
MOV @R0, B              пересилання старшого байта у регістр R0
```

```
INC R0                  інкремент R0
      JMB M1                перехід на мітку M1
      ERR: SETB P1.3        видача сигналу«ПОМИЛКА»
      W1:  JNB P1.4, W1     отриманий сигнал
                            «ПІДТВЕРДЖЕННЯ ПОМИЛКИ»,
                            немає – перехід по мітці
JMB M1                  перехід до повторної передачі
                        поточного байта
END                     Кінець програми.
```

Варіанти завдань

Варіант 1. При прийомі-передачі інформації здійснюється контроль на парність. Біт парності передається через нульовий біт порту P1. Скласти програму роботи мікроконтролера з прийому інформації.

Варіант 2. При прийомі-передачі інформації здійснюється контроль на парність. Біт парності передається через нульовий біт порту P1. Скласти програму роботи мікроконтролера з передачі інформації.

Варіант 3. Дані передаються двома інтервалами через порт P0. Спочатку передається пряме значення байта, а потім інверсне. Скласти програму роботи мікроконтролера з прийому інформації.

Варіант 4. Дані передаються двома інтервалами через порт P0. Спочатку передається пряме значення байта, а потім інверсне. Скласти програму роботи мікроконтролера з передачі інформації.

Варіант 5. Дані передаються двома значеннями. Через порт P0 передається пряме значення, а через порт P2 інверсне. Скласти програму роботи мікроконтролера з прийому інформації.

Варіант 6. Дані передаються двома значеннями. Через порт P0 передається пряме значення, а через порт P2 інверсне. Скласти програму роботи мікроконтролера з передачі інформації.

Варіант 7. Дані передаються потетрадно. Через 0–3 біти порту P0 передається пряме значення тетради, а через 4–7 біти – інверсне. У першому циклі передається молодша тетрада, у другому старша. Скласти програму роботи мікроконтролера з прийому інформації.

Варіант 8. Дані передаються потетрадно. Через 0–3 біти порту P0 передається пряме значення тетради, а через 4–7 біти – інверсне. У першому циклі передається молодша тетрада, у другому – старша. Скласти програму роботи мікроконтролера з передачі інформації.

Варіант 9. Дані передаються послідовно через нульовий біт порту P1. Спочатку передається 8 біт інформації, а потім біт контролю на парність. Скласти програму роботи мікроконтролера з прийому інформації.

Варіант 10. Дані передаються послідовно через нульовий біт порту P1. Спочатку передається 8 біт інформації, а потім біт контролю на парність. Скласти програму роботи мікроконтролера з передачі інформації.

Зміст звіту

- текст програми та її алгоритм;
- результати роботи програми.

Контрольні запитання

1. Призначення портів введення-виведення P0 і P3.
2. Наведіть структуру портів P1, P2 і P3.
3. Призначення структурних елементів портів регістра-защіпки.
4. Які альтернативні функції можуть бути реалізовані портом P3 і як вони задіються?
5. У чому відмінність порту P0 від портів P1, P2, P3? Наведіть приклади запису інформації у порти P0 і P1.
6. Як здійснюється запис інформації у порти? Наведіть приклади команд запису інформації у порти P2 і P3.
7. За допомогою яких команд здійснюється звертання до портів введення-виведення?
8. Наведіть команди, які здійснюють спеціальний режим «читання – модифікація – запис» і поясніть механізм його роботи.
9. Наведіть приклади команд роботи з портами.
10. Назвіть фізичні рівні сигналу в порту P1 і способи їхнього посилення.
11. Наведіть регістри, які можуть працювати з бітами.
12. Назвіть фізичні рівні сигналів у портах P0, P2 і P3.

3.10. Лабораторна робота 10.
Дослідження організації й застосування підпрограм

Мета роботи – дослідити організацію роботи підпрограм, написаних мовою Асемблер для програмування мікроконтролера К1816ВЕ51.

Порядок підготовки до роботи

- вивчити рекомендовану літературу [20] і методичні вказівки;
- познайомитися із принципами побудови й викликом підпрограм мовою Асемблер.

Опис лабораторної установки

- ПЕОМ;
- програма емулятора мікроконтролера МК 51.

Основні відомості

При розробці МК-систем можуть бути використані два способи організації прикладних програм: монолітний і модульний. При першому способі вся прикладна програма МК розробляється як єдине ціле, а при другому будується з окремих програмних блоків, кожен з яких реалізує деяку процедуру оброблення даних або керування. Взаємозв'язок блоків визначається розроблювачем при монтажі із цих блоків БСА й закінченої прикладної програми.

Окремі фрагменти прикладної програми МК можуть бути отримані у вигляді лінійної послідовності блоків, інші (багаторазово використовувані) звичайно оформляються у вигляді підпрограм, до яких прикладна програма, названа основною, має можливість звернутися у міру необхідності. Підпрограма повинна мати такі властивості: виконувати закінчену процедуру оброблення даних, мати тільки один вхід і один вихід та не мати ефект після дії, при якому поточне виконання підпрограми робило б вплив на її наступні виконання.

Виклик підпрограми. Звертання до підпрограми здійснюється за командою виклику CALL MARK, де MARK – символічне ім'я процедури. Ім'я процедури використовується як мітка, що вказує на одну з команд (найчастіше першу) підпрограми. Для МК 51 мнемонічне значення CALL

є узагальненим і транслюється в одну з команд ACALL або LCALL залежно від адресної відстані підпрограми, яка викликається.

За командою CALL у стеку зберігається значення лічильника команд, і повернення з підпрограми здійснюється у те місце основної програми, звідки був здійснений виклик (до команди основної програми, що слідує за командою CALL). Для цього будь-яка підпрограма повинна закінчуватися командою повернення RET, що здійснює відновлення вмісту лічильника команд зі стека.

Досить часто виникає необхідність такої організації обчислювального процесу, при якій підпрограма викликає іншу підпрограму, та, у свою чергу, викликає наступну і т. д. Цей процес називається вкладенням підпрограм. Кількість підпрограм, які можуть бути викликані таким чином (глибина вкладеності підпрограм), обмежується тільки ємністю стека.

Збереження параметрів основної програми. Іноді при звертанні до підпрограми виникає необхідність зберегти не тільки адресу повернення в основну програму, але й вміст окремих робочих регістрів. Зручним способом для цього є перемикання банку регістрів. Наприклад, якщо основна програма використовує банк регістрів 0, то підпрограма може використовувати банк регістрів 1. *Однак перемикання банку регістрів не забезпечує збереження вмісту акумулятора,* що приводить до необхідності створювати в одному з робочих регістрів або у РПД «копію» акумулятора.

Параметризовані підпрограми. Для успішної роботи будь-якої підпрограми необхідно однозначно визначити спосіб передачі у неї вихідних даних і спосіб виведення результату її роботи. Підпрограма, якій потрібна додаткова інформація у вигляді параметрів її настроювання або операндів, називається параметризованою. Прикладом параметризованої підпрограми може служити підпрограма тимчасової затримки, якщо основній програмі потрібна реалізація тимчасових затримок різної тривалості. Основна програма при цьому повинна забезпечити передачу в підпрограму вставок, що забезпечують необхідний час затримки.

Одержали поширення три способи передачі параметрів: через пам’ять, через регістри загального призначення й через регістр ознак.

При передачі вхідних параметрів через пам'ять основна програма обов'язково містить команди завантаження деяких комірок пам'яті, а підпрограма – команди зчитування із цих комірок. При передачі вихідних параметрів підпрограма повинна завантажити деякі комірки пам'яті, а основна програма – зчитати. Передача параметрів через регістри здійснюється аналогічним образом. Третій спосіб передачі параметрів – через регістр ознак – зручно використовувати при передачі вихідних параметрів (наприклад, у підпрограмах порівняння чисел). У цьому випадку підпрограма повинна встановити (або скинути) відповідні ознаки, а основна програма – проаналізувати їхнє значення. У МК 48 для цієї мети доцільно використати прапори, що зберігаються у PSW (C і F0). Набагато більшими можливостями для передачі параметрів через ознаки володіє МК 51, в якому є 128 прапорів користувача, доступних для модифікації й аналізу. Крім перерахованих способів передачі параметрів (загальних для МК 48 і МК 51), у МК 51 є ще можливість передачі параметрів через стек. Цей спосіб, зокрема, дозволяє використовувати як параметр вміст лічильника команд.

Використання процедур, оформлених у вигляді підпрограм, при розробці програмного забезпечення має ряд переваг. Насамперед, відносно прості модулі, виділені зі складної програми, можуть програмуватися декількома розроблювачами з метою скорочення часу проектування. Ще більш важливим є те, що будь-яка підпрограма допускає автономне налагодження. Це, як правило, багаторазово скорочує час налагодження всього прикладного програмного забезпечення. І нарешті, механізм використання підпрограм, що реалізують необхідний набір процедур, зменшує довжину прикладної програми, внаслідок чого зменшується ємність пам'яті програм, що вимагається.

Істотним є й та обставина, що налагоджені процедури організовуються розроблювачами в бібліотеки параметризуючих підпрограм і можуть бути багаторазово використані у проектній роботі. Відзначимо, що бібліотека параметризованих підпрограм будується на основі команд виклику підпрограм (табл. 3.10) і угоди про єдиний спосіб обміну параметрами.

Таблиця 3.10 – Група команд виклику підпрограм і повернення з них

Назва команди	Мнемокод	Т	Б	Ц	Операція
Довгий виклик підпрограми	LCALL ad 16	12	3	2	(PC) ← (PC) + 3, (SP) ← (SP) + 1, ((SP)) ← (PC_{0-7}), (SP) ← (SP) + 1, ((SP)) ← (PC_{8-15}), (PC) ← ad 16
Абсолютний виклик підпрограми	ACALL ad 11	6	2	2	(PC) ← (PC) + 2, (SP) ← (SP) + 1, ((SP)) ← (PC0-7), (SP) ← (SP) + 1, ((SP)) ← (PC_{8-15}), (PC_{0-10}) ←ad ll
Повернення з підпрограми	RET	1	1	2	(PC_{8-15}) ← ((SP)), (SP) ← (SP) – 1, (PC_{0-7}) ← ((SP)), (SP) ← (SP) - 1,
Повернення з підпрограми оброблення переривання	RETI	1	1	2	(PC_{8-15}) ← ((SP)), (SP) ← (SP) – 1, (PC_{0-7}) ← ((SP)), (SP) ← (SP) – 1

Порядок виконання роботи

1. Включити комп'ютер і завантажити програму емулятора МК51.

2. Відповідно до варіанта завдання скласти програму мовою Асемблер.

3. Використовуючи програму емулятора МК51, зробити налагодження, виконання програми й перевірити результати її виконання.

Приклад виконання програми

Завдання. Занести послідовно у порти P1 і P2 мікроконтролера вміст молодших байтів лічильників у двійково-десятковому форматі (у P1 – сотні, у P2 – десятки й одиниці).

```
START:
MOV R0, TL0      (R0) ← (TL0)    Пересилання молодшого байта
                                 таймера в регістр R0
CALL 50H         (PC) ← (50H)    Виклик підпрограми
MOV P1, R6       (P1) ← (R6)     Пересилання вмісту R6 у порт P1
MOV P2, R5       (P2) ← (R5)     Пересилання вмісту R5 у порт P2
MOV R0, TL1      (R0) ← (TL1)    Пересилання молодшого байта
                                 TL1 таймера у регістр R0
CALL 50H                         (PC) ← (50H)  Виклик підпрограми
MOV P1, R6       (P1) ← (R6)     Пересилання вмісту R6 у порт P1
MOV P2, R5       (P2) ← (R5)     Пересилання вмісту  R5 у порт P2
```

JMP START		Безумовний перехід на мітку START
ORG 50H		Перехід на мітку за адресою 50H
MOV R7, #08H	(R7) ← 08H	Пересилання 08H у регістр R7
MOV A, #00H	(A) ← 00H	Пересилання 00H в акумулятор
MOV R5, #00H	(R5) ← 00H	Пересилання 00H у регістр R5
MOV R6, #00H	(R6) ← 00H	Пересилання 00H у регістр R6
CIKL:		Мітка
MOV A, R0	(A) ← (R0)	Пересилання вмісту R0 в акум.
RLC A	$(A_{n+1}) \leftarrow (A_n)$	Зрушення вмісту акум. уліво
MOV R0, A	(R0) ← (A)	Пересилання вмісту акумулятора у регістр R0
MOV A, R5	(A) ← (R5)	Пересилання вмісту R6 у порт P1
ADDC A, R5	(A) ← (A) + (C) + (R5)	Додавання вмісту регістра R5, C, акум. і занесення результату в акум.
DA A		Десяткова корекція акумулятора
MOV R5, A	(R5) ← (A)	Пересилання вмісту акум. у R5
MOV A, R6	(A) ← (R6)	Пересилання вмісту R6 в акум.
ADDC A, R6	(A) ← (A) + (C) + (R6)	Додавання вмісту регістра R6, C, акум. і занесення результату в акум.
DA A		Десяткова корекція акумулятора
MOV R6, A	(R6) ← (A)	Пересилання вмісту акумулятора у регістр R6
DEC R7	(R7) ← (R7)-1	Віднімання 1 із регістра R7
CJNE R7, #00H, CIKL		Умовний перехід на мітку CIKL
RET	((PC) ← (SP))	Повернення з підпрограми
END		Кінець програми

Висновок: подана вище програма дозволяє досліджувати організацію й застосування підпрограм у мікроконтролері К1816ВЕ51.

Варіанти завдань

Варіант 1. У порти мікроконтролера P0–P3 надходять двійково-десяткові дані. Перевести дані у шістнадцятковий формат і розмістити у РПД послідовно з адреси 30H.

Варіант 2. Видати послідовно у порти P1 і P2 мікроконтролера вміст молодших байтів лічильників у двійково-десятковому форматі (у P1 – сотні, у P2 – десятки й одиниці).

Варіант 3. У порти P0–P3 надходять шістнадцяткові дані. Занести в РПД, починаючи з адреси 40H, кількість одиниць, що надійшли у кожен порт.

Варіант 4. Для кожного з регістрів R0, R3 і регістра-розширника B послідовно видати у порти інформацію про вміст регістрів:

у P0 – пряме значення байта; у P1 – інверсне значення байта;

в P2 – кількість нулів у байті; у P3.0 – прапор контролю парності.

Варіант 5. У кожен із портів P0–P2 надходять дані від двох чотирирозрядних датчиків. Видати у порт P3 суму шести датчиків, підключених до портів P0–P2.

Варіант 6. Записати у регістри R3, R7 і регістр-розширник B добуток їх старшої й молодшої тетради відповідно.

Варіант 7. Видати у порти P0–P2 кількість одиниць, що втримуються у регістрах R0, R7 і регістрі-розширнику B відповідно.

Варіант 8. Зчитати з інтервалом у 10 мілісекунд 5 значень із порту P0, використовуючи підпрограму затримки на 10 мілісекунд.

Варіант 9. На вхід зовнішнього переривання ~INT0 мікроконтролера через схему «АБО–НІ» підключено 8 джерел переривань ДП0–ДП7. Скласти підпрограму обслуговування переривання, що визначає номер джерела переривання й переходить за відповідною адресою. У випадку одночасного надходження декількох запитів вступає в дію система пріоритетів, що має вигляд у порядку убування: ДП3, ДП5, ДП7, ДП4, ДП0, ДП1, ДП2, ДП6. Адреси початку підпрограм установити самостійно.

Варіант 10. Скласти підпрограму, що за сигналом зовнішнього переривання INT1 видає у порти P0–P2 поточну суму таймерів-лічильників.

Зміст звіту:

- текст програми;
- алгоритм і результати роботи.

Контрольні запитання

1. Назвіть команди звертання до підпрограм.
2. За якою командою у стеці зберігається значення лічильника команд?
3. Якою командою закінчується підпрограма?
4. Що таке вкладення підпрограм?
5. Як здійснюється збереження параметрів основної програми?
6. Що таке параметризовані підпрограми?
7. Укажіть способи передачі параметрів параметризованих підпрограм.
8. Наведіть особливості параметризованих підпрограм.
9. Поясніть роботу команди ACALL ad 11 і особливості її застосування.
10. Викладіть роботу команди RET і особливості її застосування при організаціїї підпрограм.
11. Поясніть роботу команди LJMP ad 16.
12. Поясніть призначення, будову і принцип дії стека.
13. Поясніть призначення, принцип дії і роль указівника стека при організації підпрограм.
14. Поясніть роботу команд LCALL ad 16 та особливості її застосування при організації підпрограм.
15. Викладіть роботу команди RETI при обслуговуванні переривань.
16. Поясніть роботу команд: AJMP ad 11; SJMP rel.
17. Поясніть призначення і особливості роботи підпрограми обслуговування переривання.
18. Викладіть роботу команди RETI і особливості її застосування при організації підпрограм.
19. Поясніть призначення системи пріоритетів та її принцип дії у випадку одночасного надходження декількох зовнішніх запитів на переривання.

4. ЗАСТОСУВАННЯ МІКРОКОНТРОЛЕРІВ

Сьогодні мікропроцесорні пристрої широко застосовуються для вирішення найрізніших завдань практично в усіх областях діяльності людини. Можливі області застосування МК пов'язані з їх архітектурою і розрядністю:

- 8-розрядні мікроконтролери сімейства МК51 застосовуються для керування різними технологічними процесами у виробництві, випробуваннях і дослідженнях устаткування, у пристроях керування транспортом, у службовій, комерційній, торговій і побутовій апаратурі, пристроях автоматичного керування вимірювальними приладами;
- 16-розрядні МП використовуються у системах зв'язку, збору і оброблення інформації, контрольно-розподільчих системах, у системах навігації, аналого-цифрових перетворювачах, у мікро-ЕОМ широкого призначення;
- 32-розрядні застосовуються в мікро- і міні-ЕОМ широкого призначення, МПС цільового призначення, спеціалізованих процесорах, цифрових фільтрах, автокореляторах.
- 64-розрядні МП – в мультимедійних, у 2D і 3D-графічних системах.

4.1. Застосування мікроконтролерів в електромеханічних системах

На сьогодні мікроконтролери широко використовуються у керуванні різними електромеханічними системами, використовуваними у транспорті, в енергетиці, різних галузях промисловості, електропобутовій техніці та ін.

У [21] наведені класифікація і опис задач електромеханіки і транспорту, які розв'язуються із застосуванням мікроконтролерних пристроїв. Там же наведені приклади застосування мікроконтролерів у транспорті. Так, *для керування потужними транспортними дизельними установками від* 250 *до* 3000 кВт *і більше* використовується електронний регулятор фірми –«Хайнцман». Там же вказані найбільш відомі фірми-виробники

таких регуляторів, як, «Бош», «Хайнцман», «Барбет», «Вудвард» і «Колман». Основою регуляторів такого класу є швидкодіючий, потужний МП. Власна програма регулятора зберігається у постійній flash-пам'яті.

Регулятор визначає робочий стан дизеля за сигналами, які надходять від різних датчиків. У двигуна можуть контролюватися температура охолоджуючої рідини, тиск масла, кількість оборотів та інші параметри [21].

У [27] наведений приклад і функціональна схема з використанням мікроконтролера у структурі системи *керування процесом рекуперативного гальмування*, побудованої для електрорухомого складу постійного струму, експлуатованого на залізницях України. Ця система дозволяє істотно зменшити споживання енергії.

Наведена [21] система керування технологічним процесом без введення ЕОМ у контур регулювання, в якій керуюча програма знаходиться в ЕОМ, забезпечує задану послідовність виконання технологічних операцій, задану тривалість їх виконання у часі, темп прискорення і уповільнення, обробку сигналів від технологічних датчиків, а також датчиків аварійних ситуацій з відповідним відключенням. Виконавчим органом є електродвигун з його власною системою керування.

Там же наведені приклади застосування МК у системах керування технологічними процесами з частковим і повним включенням ЕОМ у контури регулювання технологічних параметрів і електроприводу.

Для реалізації системи векторного керування асинхронними і синхронними електроприводами такими фірмами, як «SIEMENS», «ABB», «MITSUBISHI» та ін., використовуються спеціалізовані мікросистеми фірм «TEXAS INSTRUMENT» або аналог «DEVICES» [28].

Для цифрового керування електродвигунами і для вирішення ряду інших задач фірма «TEXAS INSTRUMENT» розробила сімейство цифрових сигнальних процесорів (ЦСП) TMS320C20x (платформа C2000) з розвиненою периферією і невисокою вартістю. Розроблені фірмою три платформи – C2000, C5000, C6000 – за запатентованою технологією виробництва кремнію TimeLine™ з роздільною здатністю 0,18 мкм дозволяють забезпечити весь діапазон можливих застосувань ЦСП. Цим забезпечується можливість найширшого вибору процесора за критерієм «продуктивність / вартість / споживана потужність». Ці мікросистеми дозволяють

реалізувати складні обчислювальні алгоритми з необхідною швидкодією. Інтервал повторення основних обчислень складає 25 мкс. *Платформа C5000* орієнтована на застосування у портативних пристроях і у мобільному зв'язку. Використання 0,18 мкм технології дозволяє досягти продуктивності (до 800 MIPS) при зниженні енергоспоживання до 0,05 мВт/ MIPS і вартості окремих ЦСП до 5$.

Платформа C6000 покликана забезпечити максимальну продуктивність у системах керування, що вимагають граничних швидкостей обчислень як з фіксованою, так і плаваючою крапками. Вона включає дві гілки 32-розрядних ЦСП з фіксованою і плаваючою крапками. До сімейств процесорів з фіксованою крапкою належать C62x з продуктивністю від 1200 до 2400 MIPS і нові процесори сімейства C64x з продуктивністю від 4800 до 8800 MIPS. Процесори сімейства C67x належать до пристроїв з рухомою крапкою і продуктивністю до одного мільярда операцій з рухомою крапкою у секунду (1 GFLOPS) при тактовій частоті 167 МГц. Висока продуктивність процесорів цієї платформи досягається за рахунок нової архітектури VelociTI™ з «дуже довгим командним словом» (VLIW) і з новітніми апаратними рішеннями. На виконання видається вісім 32-розрядних команд, кожна з яких виконується одним з восьми незалежних функціональних пристроїв, згрупованих у 2 блоки. Процес виконання команд конвеєризований і розпадається на етапи вибірки, розпаковування, декодування і виконання. *C6000* застосовується у модемних пулах, базових станціях і відеосистемах.

4.2. Застосування мікроконтролерів в електроапаратобудуванні

МП пристрої використовуються у керуванні електричними апаратами, комплектними розподільними пристроями (КРП), технологічними процесами виробництва, випробувань і досліджень, гнучкими системами релейного захисту (ГСРЗ) та ін. [29, 30–32, 37–43]. Нижче наводяться приклади використання МП в комплектних розподільних пристроях з високовольтними електричними апаратами [31], у системах автоматичного керування процесами випробувань і досліджень електричних апаратів [29, 30] і у ГСРЗ [32].

У [31] наведено КРП з мікропроцесорним керуванням і релейним захистом, що випускає АТ «Рівненський завод високовольтних апаратів» (РЗВА). У шафах КРП серії КУ-10Ц як основну комплектуючу апаратуру застосовують:

- високовольтний вакуумний вимикач ВВКЕ-10 (VMIS);
- трансформатор струму ТЛК-10,ТВЛ;
- трансформатори струму нульового захисту ТЗЛМ;
- трансформатори напруги ЗНОЛ-06, НОЛ-08, НАМИ;
- обмежувачі перенапруження ОПНС, розрядники РВО;
- запобіжники силові ПКНТ;
- запобіжники трансформаторів напруги ПКН.

КРП серії НКАИ670049.003 з мікропроцесорним керуванням мають порівняно з КРП інших серій аналогічного призначення ряд переваг, а саме:

1. Наочність процесу роботи КРП за рахунок більшої кількості вимірювань і сигналізації, а також показу інформації на динамічних екранах, які дають можливість оператору своєчасно реагувати для запобігання аварії.

2. Дистанційне керування як терміналами релейного захисту, так і первинним устаткуванням підстанцій (порівняно з місцевим керуванням у разі використання традиційного устаткування).

3. Постійну діагностику устаткування, що дозволяє проводити передаварійну профілактику устаткування (порівняно з поставарійним або періодичним технічним обслуговуванням традиційного устаткування).

4. Можливість покрокового нарощування системи як релейного захисту, так і систем вимірювання і керування, зміни їх функцій шляхом перепрограмування.

5. Можливість реєстрації і збереження всіх величин, контрольованих параметрів у передаварійних і аварійних режимах роботи, що дозволяє провести точний поставарійний комп'ютерний аналіз причин аварії (така можливість повністю відсутня у традиційному устаткуванні).

6. Можливість реалізації ряду допоміжних функцій керування і контролю.

Наведені вище переваги забезпечуються застосуванням у новому КРП мікропроцесора типу «SPAC» фірми АВВ – Чебоксари (Росія) або

REF-542 ABB. Докладніший виклад особливостей застосування МП «SPAC» і REF–542 ABB в КРП наведений у [31].

Застосування МП у гнучких системах релейного захисту (ГСРЗ) забезпечує ефективне запобігання і (або) локалізацію аварій [32].

Кінцевою метою функціонування релейного захисту (РЗ) є забезпечення безаварійності об'єктів захисту (ОЗ) (електричних станцій, ліній електропередачі, електроенергетичних установок і т. п.), тобто можливості системи РЗ шляхом відключення ОЗ своєчасно запобігати розвитку аварійних ситуацій, небезпечних для устаткування і обслуговуючого персоналу. Попереднє покоління пристроїв РЗ було створено на базі електромеханічних реле, напівпровідникових елементів і аналогових інтегральних мікросхем (ІМС). Створені на основі таких непрограмованих елементів, вони функціонально є кінцеві автомати другого роду з незмінною (жорсткою) архітектурою і знаходять застосування у цей час при реалізації простих алгоритмів виявлення пошкоджень [32]. На відміну від них *ГСРЗ мають можливість* перепрограмування на реалізацію тих або інших функцій без зміни складу комплексу технічних засобів і реалізації алгоритмів виявлення пошкоджень підвищеної складності з використанням принципів адаптації і автоматизації процесів діагностики і настроюванню апаратури. Це дозволяє знизити збиток від пошкодження ОЗ, підвищити якість електроенергії і скоротити витрати на обслуговування, контроль, розробку і проектування РЗ. Наприклад, застосування МП релейного захисту на базі МП серії К589 дозволяє забезпечити комплексний захист генератора (КЗГ) автономної електростанції [32].

Указаний захист здійснює виявлення перевантажень первинного двигуна (наприклад, турбіни) генератора, зовнішніх і внутрішніх коротких замикань (КЗ), зниження напруги, переходу генератора у руховий режим. Перевантаження генератора, що викликає перегрівання його обмоток, виявляється за параметром «квадрат діючого значення струму». Цей же КЗГ контролює струми всіх трьох фаз. Допустима тривалість перевантаження визначається фазою з максимальним значенням струму. Цей же інтегральний параметр використовується для захисту від зовнішніх коротких замикань.

Перевантаження привідного двигуна виявляється за двома параметрами – частотою напруги на виведеннях генератора і активною потужністю, що віддається ним. Якщо зниження частоти не перевищує заданого

значення, допустима тривалість перевантаження визначається тільки активною потужністю. Для забезпечення можливості збереження генератора в роботі у разі перевантажень у комплексному захисті генератора (КЗГ) передбачені два ступені розвантаження (шляхом відключення невідповідальних споживачів).

Зниження напруги виявляється за параметром «квадрат діючого значення напруги», а перехід генератора у режим двигуна (що є неприпустимим навантаженням для паралельно працюючих з ним генераторів) за параметром «зворотна активна потужність». Детальніше питання застосування МП і МК у ГСРЗ викладені в [32].

4.3. Застосування МК51 при випробуваннях електричних апаратів

Аналіз вимог, що висуваються до електричних апаратів захисту (автоматичних вимикачів (АВ) і швидкодіючих запобіжників (ШЗ) та ін.) та методів їх випробувань, указує на вельми широкий перелік параметрів, які повинні перевірятися і досліджуватися при проведенні різних випробувань і досліджень. До таких параметрів належать: номінальний струм, струм перевантаження, струм короткого замикання, напруга на дузі, Джоулевий інтеграл, час відключення, температура на виводах, температура у центрі плавкого елемента, температура контактів, швидкість руху дуги у дугогасних решітках та ін. Все це вказує на необхідність використання вельми широкого спектра, відповідних датчиків, що дозволяють з необхідною точністю відстежувати зміну цих параметрів у процесі досліджень [29].

При проведенні комутаційних досліджень на постійному струмі таких електричних апаратів, як швидкодіючі запобіжники, автоматичні вимикачі та інші використовуються експериментальні установки, що включають головний ланцюг і ланцюг керування. Схема однієї з таких установок подана на рис. 4.1. Головний ланцюг установки складають ударний генератор (УГ) ($U_{н}$ = 880 В, $I_{уд}$ = 70 кА), регульовані реактори L, регульований опір $R_{а}$, захисний вимикач (ЗВ), вмикаючий апарат (ВА), макет апарата (МА). Проведення досліджень здійснюється за допомогою пульта електронного керування (ПЕК) і електромеханічного або електронного осцилографа (ЕО).

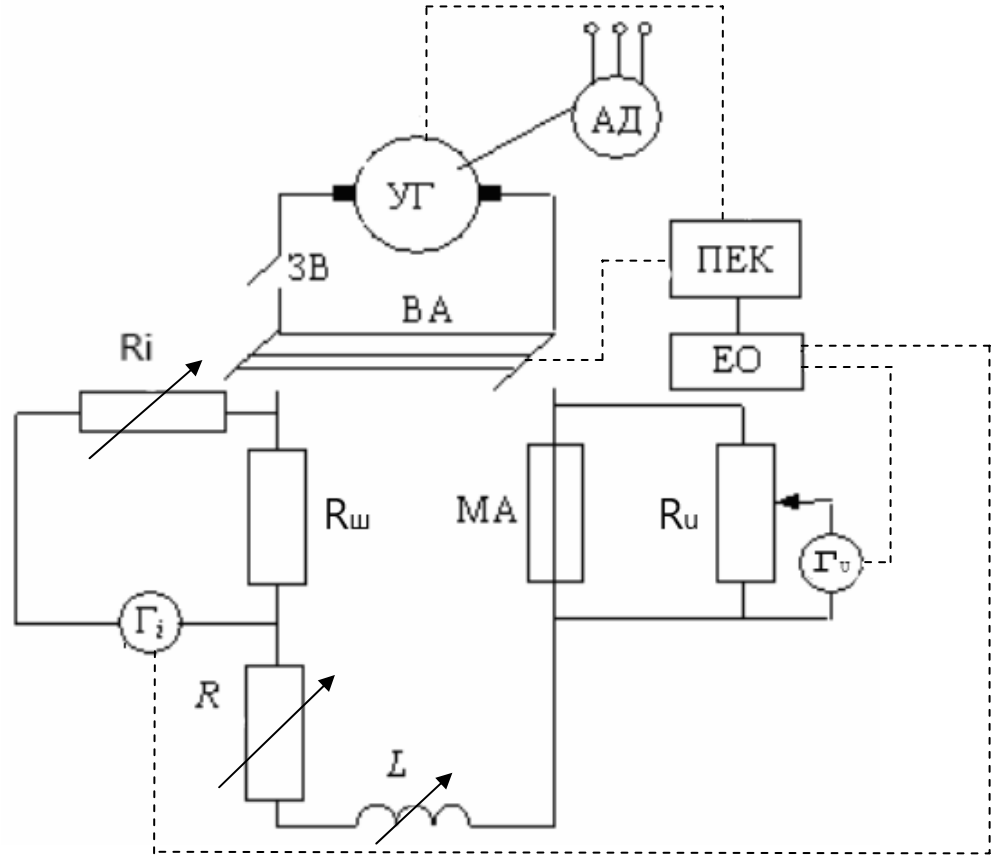


Рисунок 4.1 – Схема стенда для комутаційних досліджень ЕА

Вимірювання струмів проводиться за допомогою шунта з опором $R_{ш} = 0{,}7 \cdot 10^{-5}$ Ом. Напруга на дузі вимірюється за схемою дільника напруги. Криві струму і напруги у стандартних експериментах записувалися на світлочутливий папір за допомогою світлопроменевого осцилографа. У цьому випадку обробка осцилограм проводиться графічним методом, що вимагає великих трудових витрат і часу та знижує точність вимірювань.

При дослідженні мало вивчених процесів для того, щоб забезпечити реєстрацію можливих гострих піків перенапруження, для запису кривих струму і напруги необхідно використовувати електронний осцилограф і здійснювати фотографування за допомогою фотоприставки. У випадках, коли потрібна підвищена точність, обробка осцилограми проводиться за допомогою вимірювального мікроскопа. Все це також призводить до додаткових матеріальних, часових і трудових витрат. Скоротити

терміни проведення комутаційних досліджень, підвищити точність вимірювань, знизити їх вартість можна, застосувавши розроблену і викладену [29] автоматизовану систему керування технологічним процесом випробувань або досліджень (АСК ТПВ) із застосуванням однокристального мікроконтролера.

Структурна схема АСК ТПВ, подана на рис. 4.2, виконана на базі мікроконтролера серії МК1816ВЕ51 (МК51) [29].

Схема включає:

- датчики контрольованих параметрів (струму, напруги, температури, Джоулевого інтеграла) Д1–Д4 (первинні перетворювачі);
- нормуючі підсилювачі П1–П4;
- чотириканальний комутатор аналогових сигналів типу КР590КИ6;
- аналого-цифровий перетворювач типу К1113ПВ1;
- мікроконтролер, що містить вбудований генератор тактових сигналів, пам’ять команд, ОЗП, вбудовані 3 порти і послідовний канал зв’язку;
- компаратори К1–К4 типу К554 СА3, виходи яких за «АБО» об’єднані з вихідними сигналами керуючого мікроконтролера;
- пристрої зв’язку, узгодження і обміну ПЗО1–ПЗО4, які включають виконавчі пристрої силової установки, що задають режим випробувань або досліджень.

Через послідовний інтерфейс RS232C АСКТПВ пов’язана з ПЕОМ, яка може змінювати режими випробувань або досліджень, а також приймати, запам’ятовувати, відображати і документувати результати випробувань або досліджень.

До об’єкта дослідження підключені відповідні датчики. Датчики контрольованих параметрів Д1–Д4 є первинними перетворювачами струму, напруги, температури, Джоулевого інтеграла у напругу. Нормуючі підсилювачі погоджують вихідну напругу датчиків з необхідним вхідним сигналом АЦП 0 – 10 В і забезпечують низький вихідний опір.

Комутатор аналогових сигналів перемикає один із входів на вихід залежно від керуючого коду, що надійшов від мікроконтролера.

АЦП є швидкодіючим десятирозрядним перетворювачем вхідної напруги в паралельний двійковий код. Запуск перетворювача проводить-

ся мікроконтролером, закінчення перетворення викликає сигнал готовності, який є командою для прочитування даних.

Мікроконтролер як МП пристрій відповідно до записаної у пам'ять програми керує процесом досліджень або випробувань шляхом опитування із заданою періодичністю датчиків Д1–Д4 згідно з алгоритмом керування. Вихідні сигнали датчиків унаслідок їх різної фізичної природи можуть потребувати посилення і проміжного перетворення на АЦП або на схемах формувачів сигналів (ФС), які найчастіше виконують функції гальванічної розв'язки і формування рівнів двійкових сигналів стандарту ТТЛ.

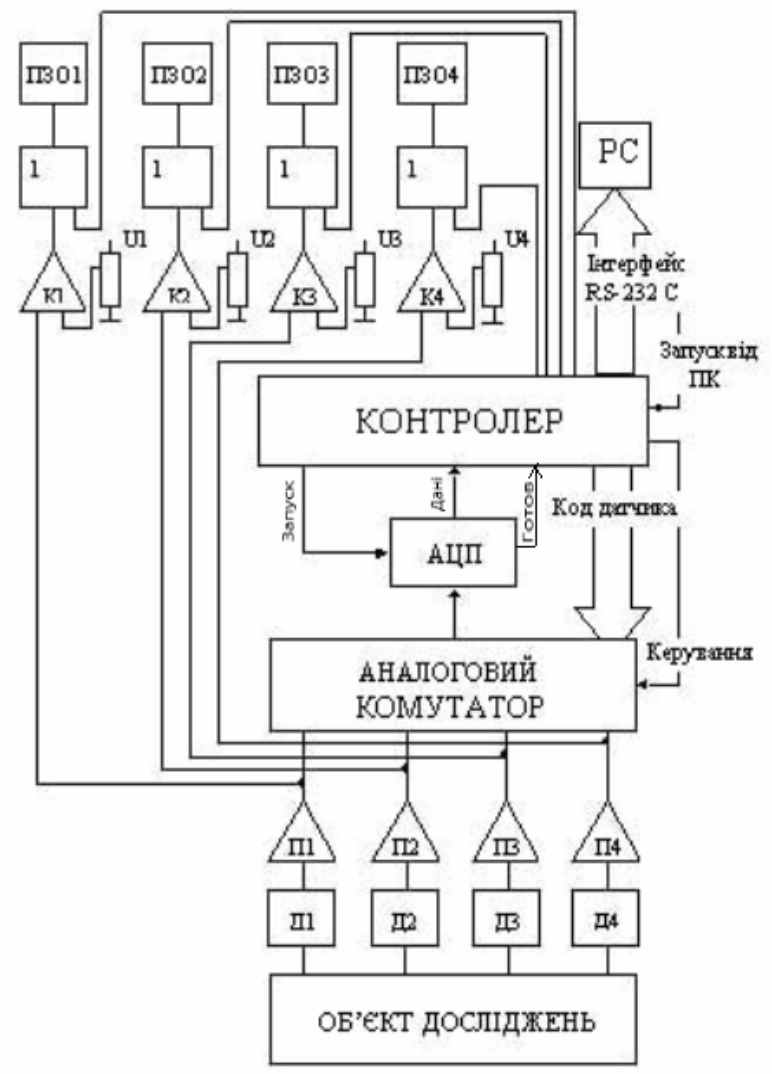


Рисунок 4.2 – Структурна схема автоматизованої системи керування технологічним процесом досліджень електричних апаратів

Мікроконтролер з необхідною періодичністю оновлює керуючі слова на своїх вихідних портах. Деяка частина керуючого слова може інтерпретуватися як сукупність прямих двійкових сигналів керування (СК), що через схеми формувачів сигналів (підсилювачі потужності, реле, оптрони та ін.) або пристрої зв'язку з об'єктом (ПЗО1–ПЗО4) надходять на виконавчі механізми (ВМ). Компаратори К1–К4 є паралельним апаратним контуром для захисту від аварійних режимів. ПЗО1–ПЗО4 є підсилювачами потужності, які керують виконавчими механізмами або пристроями силової установки.

Обґрунтування вибору мікропроцесорної системи наведено на прикладі для досліджень швидкодіючих запобіжників.

Мікропроцесор для описаної вище системи вибирається виходячи з характеру таких досліджуваних процесів і умов досліджень:

- швидкості протікання процесів;
- кількості досліджуваних параметрів і частоти опитування датчиків;
- завдань з переробки інформації;
- умов експлуатації і вимог щодо надійності.

Аналіз вихідних даних цього завдання показує, що його рішення може бути здійснено на базі мікроконтролера серії МК51.

Система на базі цього мікроконтролера здатна опитувати датчики з частотою 100 мкс, тобто за час відключення запобіжника $t_в \le 10$ мкс система встигне опитати датчики 100 разів, чого цілком достатньо для зняття і побудови характеристик запобіжника з необхідною точністю.

Найбільш прийнятним для дослідження характеристик швидкодіючих запобіжників і автоматичних вимикачів є мікроконтролер типу МК1816ВЕ51, що має такі технічні показники:

- тип – паралельний;
- розрядність паралельно оброблюваної інформації – 8 двійкових розрядів;
- форма подання чисел – двійковий додатковий код;
- методи адресації – регістрова, пряма, непряма – регістрова, безпосередня;
- одиниця, що адресується, – байт;

- кількість команд – 111, включаючи команди арифметичних і логічних операцій, стекових операцій, складання слів двійкової довжини, операції керування;
- формат команд – однобайтова, двобайтова, трибайтова;
- час виконання команд – 1–4 мкс;
- 32 РОН і набір регістрів спеціальних функцій;
- 128 визначуваних користувачем програмно-керованих прапори;
- послідовний інтерфейс;
- чотири 8-розрядні програмовані канали введення-виведення;
- два 16-бітові багаторежимні таймери / лічильники;
- система переривання з п’ятьма векторами і двома рівнями з програмною установкою пріоритету;
- місткість внутрішнього ОЗП – 128 байт, ПЗП – 4 кбайт.

Діалог з МК здійснюється за допомогою послідовного інтерфейсу RS-232C через ПЕОМ або пульт керування.

Важливою особливістю арифметико-логічного пристрою (АЛП) мікроконтролера сімейства МК51 є його здатність оперувати не тільки байтами, але і бітами. Окремі програмно-доступні біти можуть бути встановлені, скинуті, інвертовані, передані, перевірені і використані у логічних операціях. Це дозволяє при керуванні об’єктами часто застосовувати алгоритми, що містять операції над вхідними і вихідними булевими змінними. АЛП являє собою паралельний 8-розрядний пристрій, що забезпечує виконання арифметичних і логічних операцій, а також операцій зсуву, тощо. АЛП може оперувати чотирма типами інформаційних об’єктів: булевими (біт), цифровими (4 біт), байтними (8 біт) і адресними (16 біт). У АЛП виконується 51 різна операція пересилання або перетворення цих даних. Оскільки використовується 11 режимів адресації (7 для даних і 4 для адрес), то шляхом комбінування «операція / режим адресація» базове число команд 111 розширюється до 255. В АЛП реалізується механізм каскадового виконання простих операцій для реалізації складних команд, наприклад, таких, як команда умовної передачі керування за результатами порівняння.

Робота схеми здійснюється за розробленим алгоритмом і програмою. Алгоритм роботи схеми наведений на рис. 4.3.

Розроблена АСКТПВ дозволяє скоротити терміни проведення досліджень, підвищити достовірність і економічну ефективність.

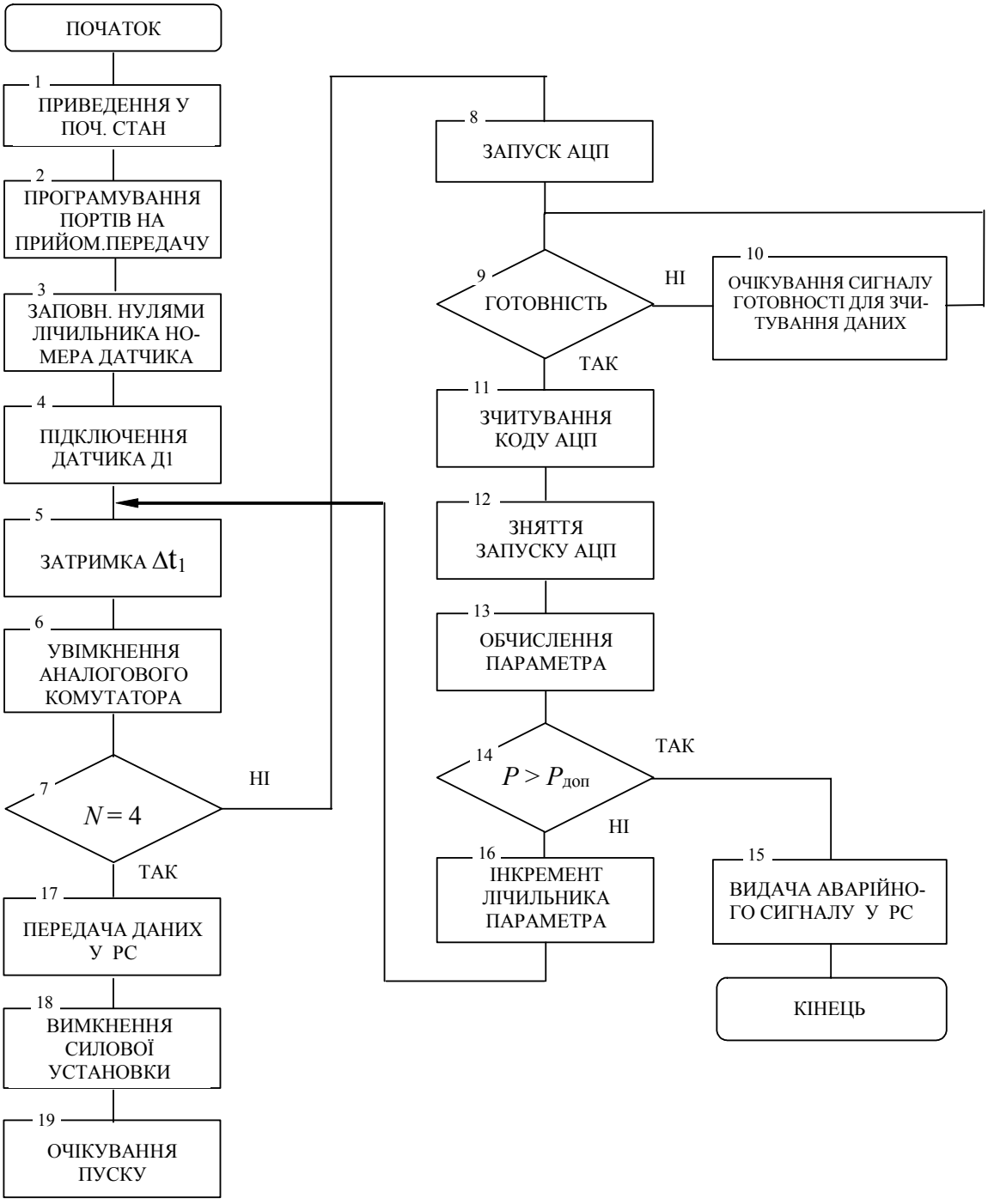


Рисунок 4.3 – Алгоритм роботи схеми автоматизованої системи керування технологічним процесом дослідження швидкодіючих запобіжників

4.4. Структурна схема АСКТПВ з паралельними АЦП

При дослідженні багатьох різних за природою фізичних процесів (горіння і гасіння електричної дуги, нагрівання, електромагнітне поле та ін.), що відбуваються при комутації в електричних апаратах захисту, виникає необхідність у визначенні їх характеристик і параметрів. Ці процеси при відключенні аварійних струмів вельми короткочасні і мають тривалість від 1 до 10 мс. Для дослідження таких процесів можуть бути використані установки, наведені в [29, 33, 34], схема однієї з яких описана у п. 4.2 і наведена на рис. 4.1. Керування і проведення досліджень у таких установках здійснюється за допомогою пульта електронного керування і електромеханічного або електронного осцилографа, що надалі при графічній обробці осцилограм призводить до похибок, додаткових матеріальних, часових і трудових витрат і робить їх малоефективними. Запропонована вище у п. 4.3 (рис. 4.2) структурна схема АСКТПВ на базі МК51 з послідовним АЦП у ряді випадків також не забезпечує ефективного розв'язання цих задач, зважаючи на низьку швидкодію МК, велику кількість датчиків і частоти їх опитування. Це завдання можна вирішити, використовуючи схему АСКТПВ з паралельними АЦП, включивши до неї швидкодіючий МК серії MCS251 [30].

Для захисту електроустановок в аварійних режимах найширше використовуються такі електричні апарати захисту, як автоматичні вимикачі і швидкодіючі запобіжники. Тому розробку структурної схеми АСКТПВ доцільно виконати для одного з цих апаратів захисту. АСКТПВ для інших апаратів може відрізнятися тільки кількістю і найменуваннями контрольованих параметрів і відповідними датчиками, які будуть підключені до досліджуваного апарата.

Розглянемо на прикладі технічного завдання на створення мікроконтролерної системи керування стендом для випробувань швидкодіючих запобіжників з такими початковими даними:

- кількість контрольованих параметрів (датчиків) – 6, зокрема:
- струм (захист);
- напруга (захист);
- температура у центрі плавкого елемента (захист);
- температура на виводах;

- Джоулевий інтеграл і інтеграл горіння дуги;
- тривалість одного досліду – 4 мкс;
- кількість розрядів перетвореної інформації – 8;
- зовнішній інтерфейс обміну – RS232C;
- кількість опитувань датчиків (не менше) – 1 000.

Вирішити це завдання і скоротити терміни проведення комутаційних досліджень, підвищити точність вимірювань, знизити їх вартість дозволяє АСКТПВ з паралельними АЦП, яка розроблена на базі високопродуктивного МК сімейства MCS251 8XC251SB і подана на рис. 4.4. Схема АСКТПВ включає таке:

- датчики контрольованих параметрів (струму, напруги, температури у центрі і на виводах, Джоулевого інтеграла, інтеграла горіння дуги) Д1–Д6;
- нормуючі підсилювачі П1–П6;
- 8-канальний комутатор аналогових сигналів;
- АЦП типу К1108ПВ1 (А, Б);
- МК, що містить вбудований генератор тактових сигналів, пам’ять команд, ОЗП, вбудовані 4 порти і послідовний канал зв’язку;
- компаратори К1–К3 типу КР554СА3, виходи яких по «АБО» об’єднані з вихідними сигналами керуючого мікроконтролера;
- пристрої зв’язку з об’єктом ПЗО, які включають виконавчі пристрої (ВП) силової установки та задають режим випробувань або досліджень.

Через послідовний інтерфейс RS232C АСКТПВ з’єднана з ЕОМ, яка може змінювати режими випробувань або досліджень, а також приймати, запам’ятовувати, відображати і документувати результати випробувань або досліджень. Для виходу на послідовний інтерфейс необхідно вирішити такі проблеми: узгодження рівнів сигналів RS232 і МК (TTL); підтримання стандартної швидкості прийому передачі; підтримання стандартних форматів посилань; підтримання стандартних протоколів обміну.

Вибір мікроконтролера

Основним елементом системи керування є мікроконтролер 80C251SB фірми Intel. Цей мікроконтролер вибраний виходячи з таких умов: мікроконтролер цього типу (MCS 251) є подальшою розробкою

широко відомого у світі мікроконтролера серії MCS51, програмно сумісний зверху, але зі значно вищою швидкодією (одна операція виконується за 100 нс).

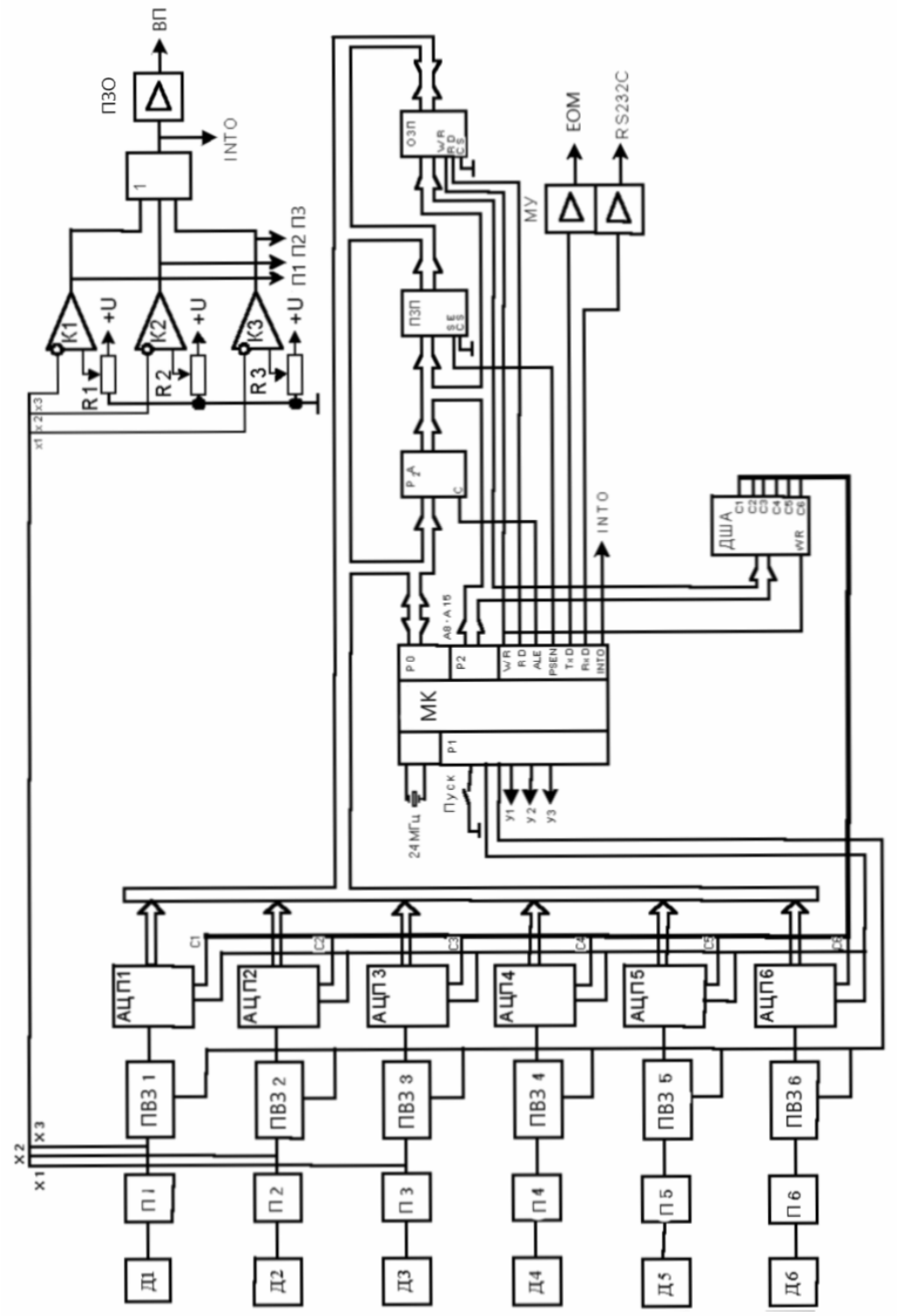


Рисунок 4.4 – Структурна схема АСКТПВ з паралельними АЦП

Мікроконтролер вибирають виходячи з умови забезпечення тривалості циклу АЦП – перетворення вхідних сигналів і запису їх у пристрій (ОЗП) мікроконтролером, що оперативно запам'ятовує. Загальна кількість операцій дорівнює 5 · 6 + 5 = 35 команд. Для забезпечення необхідної швидкодії схеми вимірювання (відповідно до завдання 4 мкс) тривалість однієї команди розраховується як $t_к$ / 35 = 114 нс. Таку швидкодію може забезпечити МК 80С251SB.

Опис архітектури мікроконтролера

8XC251SB – перший МК у сімействі MCS251 компанії Intel. Нове сімейство 8-бітових мікроконтролерів підвищує функціональність і продуктивність широко поширених мікроконтролерів MCS51 при збереженні сумісності на рівні двійкових кодів. Завдяки сумісності контактів з 8XC51F, МК 8XC251SB може служити засобом підвищення продуктивності існуючих апаратно-програмних систем. До типових областей застосування 8XC251SB можна віднести системи керування.

Усім МК сімейства MCS251 властиві такі загальні особливості:

- 24-бітова лінійна адресація до 16 Мбайт пам'яті;
- ЦПУ регістрової архітектури з регістрами, що адресуються як байти, слова і подвійні слова;
- сторінковий режим, прискорюючий вибірку команд із зовнішньої пам'яті;
- конвеєр команд;
- розширена система команд, що включає 16-бітові арифметичні і логічні команди;
- 64-кбайтовий зовнішній стек;
- мінімальний час виконання команд (2 такти у порівнянні з 12 тактами у МК MCS51);
- двійкова сумісність з МК MCS51;

Перелічимо деякі переваги, пов'язані з цими особливостями:

- збереження програм, написаних для МК MCS51;
- значно вища швидкість оброблення, ніж у МК MCS51 при тій же тактовій частоті;
- підтримка програм і даних великого розміру;
- підвищена продуктивність програм на мові *С*.

Функціональна блок-схема МК 8XC251SB, подана нижче на рис. 4.5, має таку структуру. Ядро процесора, загальне для всіх мікроконтролерів MCS251. Окремі контролери сімейства відрізняються за набором периферійних блоків на кристалі, портів введення-виведення, за зовнішньою системною шиною, розміром ОЗП на кристалі, а також за типом і обсягом внутрішньої пам'яті програм. До складу периферійних пристроїв 8XC251SB входять виділений сторожовий таймер, таймер-лічильник, матриця програмованих лічильників і порт послідовного введення-виведення. МК 8XC251SB має чотири 8-бітові порти введення-виведення, P0–P3. Кожну лінію порту введення-виведення можна окремо запрограмувати або як сигнал загального призначення, або як сигнал спеціальної функції, який підтримує або зовнішню шину, або вбудований периферійний пристрій. Порти P0 і P2 утворюють зовнішню шину, що має 16 ліній для мультиплексування 16-бітової адреси і 8-бітових даних. (МК 8XC251SB дозволяє також конфігурувати 17-й біт зовнішньої адреси). Порти P1 і P3 утворюють сигнали керування шиною і периферією.

МК 8X251SB, функціональна блок-схема якого подана на рис. 4.5, має два режими зниженого енергоспоживання. У холостому режимі тактові сигнали ЦПП зупиняються, але підтримується синхронізація периферії. У режимі мікроспоживання внутрішній тактовий генератор зупиняється, і весь кристал переходить у статичний стан. За дозволеним перериванням або апаратним скиданням кристал може вийти з холостого режиму або режиму низького споживання і повернутися у нормальний режим.

Мікроконтролери MCS251 мають розширену систему команд, доповнену новими операціями, режимами адресації і операндами.

Багато команд можуть працювати з 8-, 16- і 32-бітовими операндами що забезпечують зручне і ефективне програмування на мовах високого рівня типу С. Включені такі нові можливості, як команда TRAP, новий режим адресації зі зсувом і ряд команд умовного переходу. Аналіз опису системи команд і її порівняння з системою команд мікроконтролерів MCS51 показує, що 8XC251SB можна конфігурувати для роботи у двійковому або початковому режимі. У будь-якому режимі 8XC251SB може виконувати всі команди архітектури MCS51 і MCS251.

Проте початковий режим найбільш ефективний для виконання команд архітектури MCS251, а двійковий – для команд архітектури MCS51.

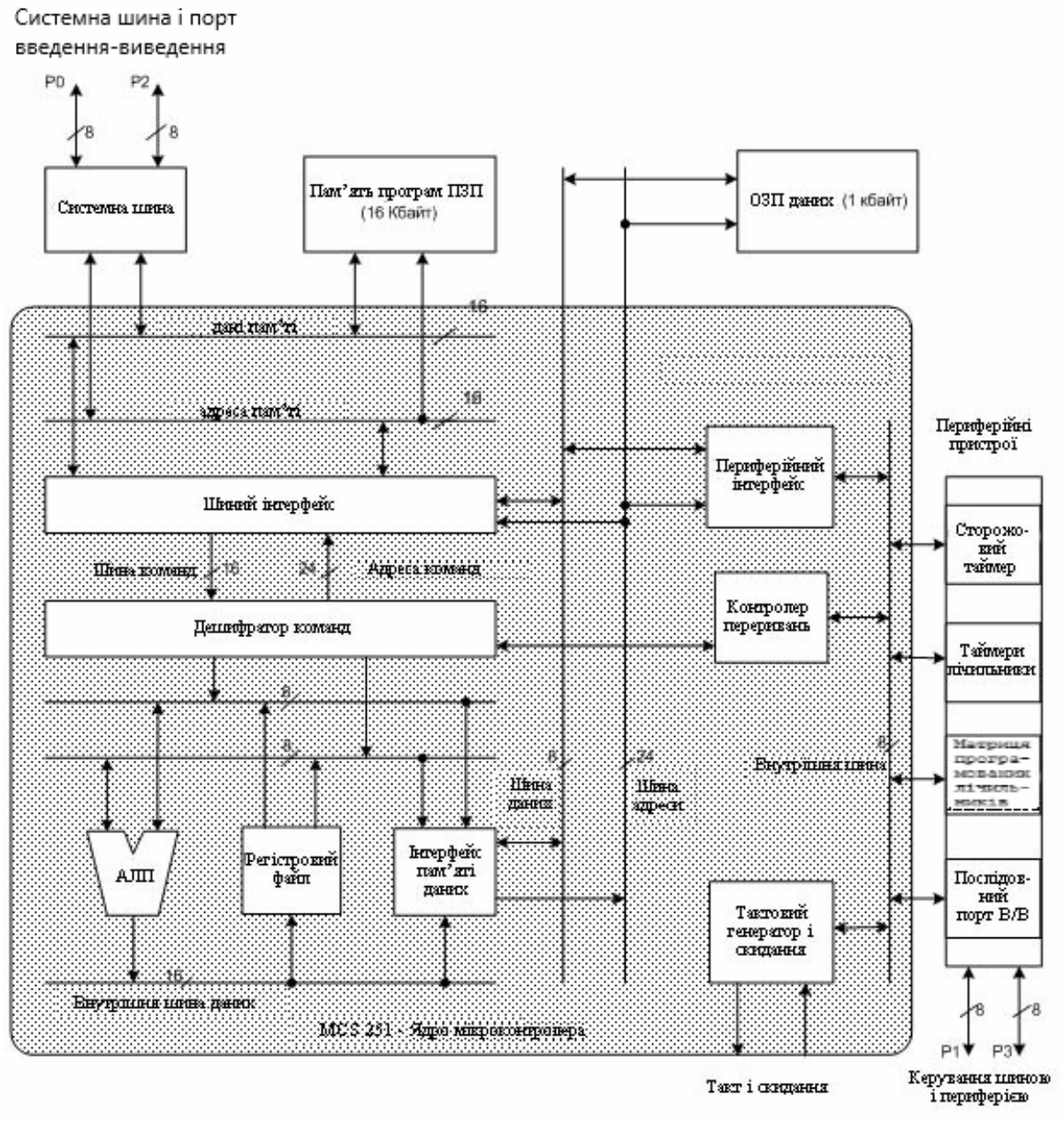


Рисунок 4.5 – Функціональна блок-схема МК 8XC251SB

У двійковому режимі об'єктні коди MCS51 можна виконувати на мікроконтролері 8XC251SB без перекомпіляції.

Якщо система спочатку була спроектована для MCS51, а новий МК 8XC251SB виконуватиме код, написаний для MCS51, то продуктивність буде вища, якщо 8XC251SB працює в двійковому режимі. Об'єктний код, написаний для MCS51, працює швидше на 8XC251SB. Проте, якщо велику частину коду переписати в новій системі команд, то продуктивність

буде вища, якщо 8XC251SB працює в початковому режимі. В цьому випадку 8XC251SB може працювати значно швидше, ніж мікроконтролер MCS51.

Мікроконтролери MCS251 для зберігання команд і даних використовують єдину, лінійну 16-Мбайтову область адресації пам'яті. МК 8XC251SB може адресувати до 128 кбайт фізичної зовнішньої пам'яті. Регістри спеціальних функцій і регістровий файл мають окремі області адресації.

Вибір АЦП

У схемі, що розробляється, застосовується 8-бітове перетворення. Кількість розрядів АЦП вибрана виходячи з наведеної похибки аналогових датчиків Д1–Д6, що не перевищує 0,5 %. Похибка перетворень АЦП Δ визначається значенням молодшого розряду одержуваного коду і дорівнює $1/2^n$, де n – кількість розрядів. Вона не повинна перевищувати похибку вхідного аналогового сигналу.

При $n = 8$, $\Delta = 1/256 = 0{,}004 = 0{,}4$ %, що цілком прийнятно для нашого випадку. Вибирати АЦП з великою розрядністю не має сенсу, оскільки молодші розряди не будуть достовірними. Крім того, швидкодіючі АЦП (у нашому випадку час перетворення менше 1 мкс) достатньо вартісні, і зі збільшенням розрядності ціна їх різко зростає. У цій схемі потрібно шість АЦП для забезпечення одночасного зняття аналогового сигналу з подальшим послідовним прочитуванням результатів і записом їх в ОЗП. Цим вимогам задовольняє АЦП К1108ПВ1 (А, Б). Мікросхема 10-розрядного швидкодіючого функціонально закінченого АЦП послідовного наближення К1108ПВ1 (А, Б) призначена для перетворення аналогового сигналу в двійковий паралельний цифровий код.

До складу функціональної схеми перетворювача входять: джерело опорної напруги (ДОН), генератор тактових імпульсів (ГТІ), вихідний регістр з трьома логічними станами і функцією зберігання інформації протягом одного циклу перетворення, вихідний регістр (ВРг), регістр послідовного перетворення (РПП), цифро-аналоговий перетворювач (ЦАП), багатовхідний компаратор напруг (КН) з вхідним віднімаючим пристроєм, дешифратор рівнів струму й ін.

Мікросхема розрахована на перетворення однополярної вхідної напруги у діапазоні від 0 до 3 В, що подається на вхід через зовнішній

операційний підсилювач (П) або пристрій вибірки і зберігання (ПВЗ) при максимальній частоті перетворення 1,1 МГц для 10-розрядного режиму і 1,33 МГц для 8-розрядного режиму.

Для роботи АЦП К1108ПВ1 потрібно декілька зовнішніх керамічних конденсаторів і джерело напруги U_{cc1} = 5 Ве 5% і U_{cc2} = 5,2 Ве 5%. Номінальне значення напруги внутрішнього ДОН складає 2,5 В. Потужність, споживана від джерела живлення, не перевищує 0,85 Вт.

Опис роботи схеми АСК ТПВ, наведеної на рис. 4.4

Сигнали, що поступають з аналогових датчиків Д1–Д6, нормуються підсилювачами П1–П6, виконаними на базі прецизійних операційних підсилювачів КР140УД26Б. Після цього вони поступають на пристрої вибірки і зберігання ПВЗ1–ПВЗ6, виконані на базі мікросхем КР1100СК2. За командою МК ПВЗ1–ПВЗ6 одночасно запам'ятовують аналогові сигнали від відповідних підсилювачів П1–П6 на якийсь час, необхідний для роботи АЦП.

Наступна команда МК запускає аналого-цифрові перетворювачі АЦП1–АЦП6, які забезпечують паралельне у часі перетворення вхідних аналогових сигналів в 8-розрядний двійковий код. Потім знімаються сигнали керування ПВЗ1–ПВЗ6 і АЦП1–АЦП6. Перетворена інформація зберігається у вихідних регістрах АЦП1–АЦП6. У циклах адресного читання МК послідовно прочитує перетворену інформацію з АЦП1–АЦП6.

Перетворений сигнал записується в ОЗП статичного типу, виконаного на базі мікросхеми IDT7164S20T місткістю 8К на 8 біт фірми NSC, час доступу 20 нс. Сигнали дозволу роботи ПВЗ1–ПВЗ6 і АЦП1–АЦП6 видаються мікроконтролером. АЦП стробується імпульсами, виданими дешифратором адреси (ДША). Як ДША застосовується демультиплексор 3 на 8 зі сторобуванням, мікросхема КР1533ИД7 або КР1554ИД7.

Тактова частота мікроконтролера в 24 МГц забезпечується вбудованим у МК генератором при підключенні до контролера зовнішнього кварцевого резонатора. Програма роботи пристрою розміщена у програмованому постійному запам'ятовуючому пристрої (ППЗП) з ультрафіолетовим стиранням типу D27C64A1 фірми Intel місткістю 8К на 8 біт, час вибірки 20 нс. ВІС ППЗП встановлюється на сокеті, що дозволяє багато разів міняти роботу пристрою, пристосовувавши його до різних типів випробовуваних запобіжників, тим самим, забезпечуючи гнучкість побудо-

ви системи автоматичного керування. Після закінчення одного циклу випробувань МК проводить настроювання зв'язку з IBM PC, після чого відбувається передача масиву даних результатів випробувань, що містяться в ОЗП, для подальшої оброблення і аналізу.

Передача проводиться у «старт-стоповому» режимі для RS232C з контролем на парність або непарність кожної посилки і перевірки за допомогою контрольної суми масиву.

При виникненні аварійної ситуації на входах 1–3 аварійний сигнал, посилений компараторами К1–К3, подається на елемент АБО, який, у свою чергу, видає імпульс на відключення всієї установки. Як компаратори К1–К3 вибираємо широко використовувані інтегральні схеми КР554СА3 з відкритим колектором, що полегшує узгодження з логічними рівнями мікроконтролера і цифрових інтегральних схем.

Установка схеми захисту виставляється резисторами R1–R3. При аварії МК переходить у режим переривання, виконання основної програми припиняється і МК видає повідомлення про аварійну ситуацію в ЕОМ.

Основний алгоритм роботи схеми АСК ТПВ [30]

На початку алгоритму проводиться установлення початкового стану всіх керуючих сигналів: керування ПВЗ і АЦП, програмуються всі параметри для настроювання послідовного порту введення-виведення RS232C. Потім проводиться очікування команди «пуск» (2) від зовнішньої схеми. У головному циклі алгоритму (3–14) проводиться включення ПВЗ1–ПВЗ6 (3), включення АЦП1–АЦП6 (4), встановлюється лічильник параметрів $i = 1$. Виключення ПВЗ1–ПВЗ6 відбувається через час затримки, рівний 0,8 мкс, необхідний для перетворення аналогового сигналу в цифровий код, після чого знімається сигнал дозволу роботи АЦП (7). Внутрішній цикл знімання інформації і запису в ОЗП виконується 6 разів (8–12), інкрементується номер вибраного АЦП і адреса ОЗП.

Блок 13 резервує дві адреси ОЗП для збільшення (у разі потреби) кількості датчиків до восьми, блок 14 аналізує стан, коли проведено 1000 опитувань, і вихід «І» означає закінчення процесу опитування всіх параметрів.

Після цього підраховується контрольна сума всього масиву (15) за модулем 256 і записується у відповідну комірку ОЗП. Потім відбувається настроювання зв'язку з РС (17–18), відбувається передача масиву

з контролем норми передачі (19–20). Якщо інформація передана достовірно, то відбувається вихід з алгоритму, у разі недостовірності переданої інформації відбувається повторна передача масиву (21) [30] .

Алгоритм переривання [30]

Переривання через аварію відбувається по входу INT0. У блоці 1 підпрограма мікроконтролера прочитує сигнали, які вказують на причину аварії. Ознака причини аварії передається через послідовний порт введення-виведення RS232C у ПЕОМ аналогічно блокам 17–21 основного алгоритму.

Робота схеми АСКТПВ з паралельними АЦП здійснюється за викладеними вище алгоритмами, що дозволяють набагато скоротити терміни проведення досліджень комутаційних електричних апаратів захисту, підвищити достовірність результатів досліджень і їх економічну ефективність. Розроблені АСК ТПВ і алгоритми можуть використовуватися при проведенні випробувань або досліджень та інших електричних апаратів і пристроїв електропобутової техніки.

Приклади застосування МК в ЕА і ЕПТ, алгоритми і програми, наведені у додатку 6 та більш детально у публікаціях автора [29, 30, 37– 43].

Контрольні запитання і завдання

1. Назвіть області застосування МК і завдання, які розв’язуються за їх допомогою. Наведіть приклади для електромеханічних систем.
2. Викладіть приклади застосування МК в електроапаратобудуванні і електропобутовій техніці.
3. Складіть структурну схему АСКТПВ для опитування 5 датчиків 20 разів за період (f = 50 Гц) на базі МП КР580, алгоритм її роботи та програму.
4. Складіть структурну схему АСКТПВ для опитування 6 датчиків 25 разів за період (f = 60 Гц) на базі МК51 і наведіть алгоритм.
5. Наведіть приклади застосування МК у керуванні технологічним процесом з включенням ЕОМ у контур регулювання технологічного параметра.
6. Наведіть приклади застосування МК у керуванні технологічним процесом без включення ЕОМ у контур регулювання.

7. Викладіть мету і особливості застосування МК у комплектних розподільних пристроях (КРП).

8. За якими показниками і як здійснюється вибір мікроконтролера?

9. Складіть структурну схему АСКТПВ з паралельними АЦП і наведіть приклад розрахунку її швидкодії.

10. Викладіть порядок розрахунку швидкодії структурної схеми АСКТПВ з послідовним АЦП при опитуванні 6 датчиків з частотою опитування 200 разів за період ($f = 50$ Гц).

11. Назвіть області використання 8-, 16-, 32- і 64-розрядних мікропроцесорів та мікроконтролерів і наведіть приклади.

12. Наведіть приклади застосування МК у керуванні електромеханічними системами транспорту.

13. Для чого застосовують МК у системах керування технологічними процесами з частковим і повним включенням ЕОМ у контур регулювання при проведенні випробувань і досліджень ЕА і ЕБТ?

14. Викладіть призначення і дайте характеристику сімейства цифрових сигнальних процесорів (ЦПС) TMS320C20X (платформа C2000), розроблених фірмою «TEXAS INSTRUMENT».

15. Де використовуються і які мають особливості ЦПС з платформами C5000 і C6000, яка їх продуктивність і за рахунок чого вона досягається?

16. Яка основна комплектуюча апаратура застосовується у КРП серії КУ-10Ц з мікропроцесорним керуванням?

17. Особливості та переваги застосування МП і МК у гнучких системах релейного захисту в порівнянні з аналоговими елементами і мікросхемами.

18. Призначення і вибір датчиків (первинних перетворювачів), нормуючих підсилювачів (нормувачів), мультиплексорів (аналогових електронних комутаторів або цифрових), компараторів, АЦП і ЦАП, пристроїв вибірки і зберігання, пристроїв зв’язку з об’єктом керування.

19. Призначення і функціонування послідовного інтерфейсу RS232C.

20. Як проводиться розрахунок швидкодії АСКТПВ на базі МП або МК, і які шляхи її підвищення?

21. Наведіть приклади застосування МП і МК в електричних апаратах.

ДОДАТОК 1

Таблиця Д 1.1 – Система команд МК «Електроніка МС2702» (МП КР580)

Мнемокод	Кільк. байтів	Код	Алгоритм	Коментар
1. Команди пересилання (не виробляють ознаки)				
MOV rd, rs	1	01DDDSSS	(rd) ← (rs)	Вміст rs пересилається у rd
MOV M, rs	1	01110SSS	(M) ← (rs)	Вміст rs пересилається за адресою, що знаходиться у M
MOV rd, M	1	01DDD110	(rd) ← (M)	Вміст комірки, адреса якої у M, пересилається у rd
MVI r, <b2>	2	00DDD110	(r) ← <b2>	Другий байт команди пересилається у r
MVI M, <b2>	2	36H	(M) ← <b2>	Другий байт команди пересилається в комірку за адресою що знаходиться у M
LXI B, <b2><b3>	3	01H	(C) ← <b2> (B) ← <b3>	Вміст <b2> пересилається у (C). Вміст <b3> пересилається у (B)
LXI D, <b2><b3>	3	11H	(E) ← <b2> (D) ← <b3>	Вміст <b2> пересилається у (E). Вміст <b3> пересилається у (D)
LXI H, <b2><b3>	3	21H	(L) ← <b2> (H) ← <b3>	Вміст <b2> пересилається у (L). Вміст <b3> пересилається у (H)
LXI SP, <b2><b3>	1	31H	(SP)МЛ ← <b2> (SP)СТ ← <b3>	Вміст <b2> пересилається в молодший розряд указівника стека (SP). Вміст <b3> пересилається у старші розряди указівника стека (SP)
LDAX B	1	0AH	(A) ← [(B), (C)]	Вміст (B)(C) записати в акумулятор
LDAX D	1	1AH	(A) ← [(D), (E)]	Вміст(D)(E) записати в акумулятор
STAX B	1	02H	[(B), (C)] ← (A)	Вміст акумулятора переслати за адресою що знаходиться у (B)(C)
STAX D	1	12H	[(D), (E)] ← (A)	Вміст акумулятора переслати за адресою що знаходиться у (D)(E)
LHLD <b2><b3>	3	2AH	L ← <b3> <b2> H ← <b3> <b2>	Вміст комірок пам’яті з адресою <b3><b2> пересилається у (H)(L)
SHLD <b2><b3>	3	22H	[<b3><b2>] ← (L) [<b3><b2>] ← (H)	Вміст (H)(L) пересилається у комірку пам’яті, адреса якої у <b3><b2>

Продовження табл. Д 1.1

Мнемокод	Кільк. байтів	Код	Алгоритм	Коментар
XCHG	1	EBH	(H) ↔ (D) (L) ↔ (E)	Обмін вмістом між (H)(L) і (D)(E)
XTHL	1	E3H	SP ↔ (L) [SP+1] ↔ (H)	Обмін вмістом між (H)(L) і [SP + 1], SP
SPHL	1	F9H	(SP) ← (H)(L)	16-байтове число (H)(L) пересилається у (SP)
PCHL	1	E9H	(PC) ← (H)(L)	Вміст (H)(L) пересилається у лічильник команди
STA <b2><b3>	3	32H	[<b3><b2>] ← (A)	Вміст акумулятора пересилається у комірку пам'яті, адреса якої у <b3><b2>
LDA <b2><b3>	3	3AH	(A) ← [<b3><b2>]	Вміст комірок пам'яті з адресою <b3><b2> пересилається в акумулятор
2. Команди розгалуження				
JMP <b2><b3>	3	C3	без умов PC ← <b3> <b2>	Безумовний перехід до команди за адресою <b3> <b2>
JC <b2><b3>	3	DA	|C| = 1	Якщо (C) = 0, виконується наступна команда
JNC <b2><b3>	3	D2	C = 0	Якщо (C) = 1, виконується наступна команда
JZ <b2><b3>	3	CA	Z = 1	Якщо (Z) = 0, виконується наступна команда
JNZ <b2><b3>	3	C2	Z = 0	Якщо (Z) = 1, виконується наступна команда
JP <b2><b3>	3	F2	S = 0	Якщо (S) = 1, виконується наступна команда
JM <b2><b3>	3	FA	S = 1	Якщо (S) = 0, виконується наступна команда
JPE <b2><b3>	3	EA	P = 1	Якщо (P) = 0, виконується наступна команда
JP0 <b2><b3>	3	E2	P = 0	Якщо (P) = 1, виконується наступна команда
CALL <b2><b3>	3	CD	без умов PC ← <b3> <b2>	Виклик підпрограми (ПП)

Продовження табл. Д 1.1

Мнемокод	Кільк. байтів	Код	Алгоритм	Коментар
CC <b2><b3>	3	DC	PC ← <b3><b2> C = 1	Якщо (C) = 1, перехід на ПП за адресою, зазначеною у <b3> <b2>
CNC <b2><b3>	3	DA	C = 0 PC ← <b3> <b2> [SP–2][SP–1] <(PC)	Якщо (C) = 0, перехід на ПП за адресою, зазначеною у <b3> <b2>
CZ <b2><b3>	3	CC	Z = 1	Якщо (Z) =1, перехід на ПП за адресою, зазначеною у <b3> <b2>
CNZ <b2><b3>	3	C4	Z = 0	Якщо (Z) = 0, перехід на ПП за адресою, зазначеною у <b3> <b2>
CM <b2><b3>	3	FC	S = 1	Якщо (S) =1, перехід на ПП за адресою, зазначеною у <b3> <b2>
CP <b2><b3>	3	F4	S = 0	Якщо (S) = 0, перехід на ПП за адресою, зазначеною у <b3> <b2>
CPE <b2><b3>	3	EC	P = 1	Якщо (P) = 1, перехід на ПП за адресою, зазначеною у <b3> <b2>
CP0 <b2><b3>	3	E4	P = 0	Якщо (P) = 0, перехід на ПП за адресою, зазначеною у <b3> <b2>
RET	1	C9	(PC) ← [SP] [SP +1], (SP) = (SP) + 2	Безумовне повернення із ПП
RC	1	D8	C = 1 (PC) ← [SP] [SP + 1]	C = 1, повернення із ПП на команду, адреса якої записана у стеці
RZ	1	C8	Z = 1	Повернення із ПП при (Z) = 1
RNZ	1	C0	Z = 0	(Z) = 0, повернення із ПП на команду
RM	1	F8	S = 1	(S) = 1, повернення із ПП на команду
RP	1	F0	S = 0	(S) = 0, повернення із ПП на команду
RPE	1	E8	P = 1	(P) = 1, повернення із ПП на команду
RPO	1	E0	P = 0	(P) = 0, повернення із ПП на команду
RST	1			ПОВТОРНИЙ ЗАПУСК

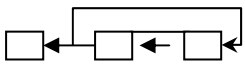
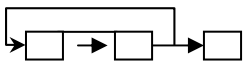
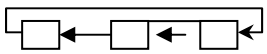
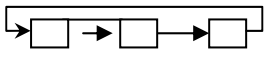
Продовження табл. Д 1.1

Мнемо-код	Кільк. байтів	Код	Ознаки					Алгоритм	Коментар
			C	Z	S	P	C'		
3. Арифметико-логічні команди									
ADD rs	1	10000SSS	+	+	+	+	+	(A) ← A) + (rs)	Вміст акумулятора складається з rs
ADC rs	1	100001SSS	+	+	+	+	+	(A) ← (A) + (rs) + (C)	Вміст акумулятора складається з rs і бітом C
SUB r	1	10010SSS	+	+	+	+	+	(A) ← (A)– (rs)	Із вмісту акумулятора віднімається вміст rs
SBB r_s	1	10011SSS	+	+	+	+	+	(A) ← (A) – (rs) – (C)	Із вмісту акумулятора віднімається вміст rs та біт C
ANA rs	1	10100SSS	0	+	+	+	–	A ← (A) ∧ (rs)	Порозрядна кон'юкція
XRA rs	1	10101SSS	0	+	+	+	–	A ← (A) ∀ (rs)	Порозрядне заперечення рівнозначності
ORA rs	1	10110SSS	0	+	+	+	–	A ← (A) ∨ (rs)	Порозрядна диз'юнкція
CMP rs	1	10111SSS	+	1	+	+	+	(A) – (rs)	Порівняння за допомогою внутрішнього вирахування
ADD M	1	86H	+	+	+	+	+	(A) ← (A) + (M)	Вміст акумулятора складається із вмістом комірки за адресою у M
ADC M	1	8EH	+	+	+	+	+	(A) ← (A) + (M) + (C)	Вміст акумулятора складається із вмістом комірки за адресою у M і бітом C
SUB M	1	96H	+	+	+	+	+	(A) ← (A) – (M)	Із вмісту акумулятора віднімається вміст комірки за адресою у M
SBB M	1	9EH	+	+	+	+	+	(A) ← (A) – (M) – (C)	Із вмісту акумулятора віднімається вміст комірки за адресою у M і біт C
ANA M	1	A6H	0	+	+	+	–	A ← (A) ∧ (M)	Порозрядна кон'юкція

Продовження табл. Д 1.1

Мнемо-код	Кільк. байтів	Код	Ознаки					Алгоритм	Коментар
			C	Z	S	P	C'		
XRA M	1	AEH	0	+	+	+	-	A ← (A) ∀ (M)	Порозрядне заперечення рівнозначності
ORA M	1	B6H	0	+	+	+	-	A ← (A) ∨ (M)	Порозрядна диз'юнкція
CMP M	1	BEH	+	+	+	+	+	(A) – (M)	Порівняння за допомогою внутрішнього вирахування
ADI<b2>	2	C6H	+	+	+	+	+	A ← (A) + <b2>	Вміст акумулятора складається із вмістом 2-го байта команди
ACI<b2>	2	CEH	+	+	+	+	+	(A) ← (A) + + <b2> + (C)	Вміст акумулятора складається із вмістом 2-го байта команди і бітом C
SUI<b2>	2	D6H	+	+	+	+	+	(A) ← (A) – <b2>	Із вмісту акумулятора віднімається вміст 2-го байта команди
SBI<b2>	2	DEH	+	+	+	+	+	(A) ← (A) – – <b2> – (C)	Із вмісту акумулятора віднімається вміст 2-го байта команди і біт C
ANI<b2>	2	E6H	0	+	+	+	–	(A) ← (A) ∧ <b2>	Порозрядна кон'юнкція
XRI<b2>	2	EEH	0	+	+	+	–	(A) ← (A) ∀ <b2>	Порозрядне заперечення рівнозначності
ORI<b2>	2	F6H	0	+	+	+	–	(A) ← (A) ∨ <b2>	Порозрядна диз'юнкція
CPI<b2>	2	FEH	+	+	+	+	+	(A) – <b2>	Порівняння за допомогою внутрішнього вирахування
INR rd	1	000DD100	–	+	+	+	+	(rd) ← (rd) +1	Вміст rd збільшується на 1
DCR rd	1	00DDD101	–	+	+	+	+	(rd) ← (rd) – 1	Вміст rd зменшується на 1
INX B	1	03H	–	–	–	–	–	(B)(C) ← (B)(C) + 1	Вміст (B)(C) збільшується на 1

Продовження табл. Д 1.1

Мнемо-код	Кільк. байтів	Код	Ознаки					Алгоритм	Коментар
			C	Z	S	P	C'		
INX D	1	13H	–	–	–	–	–	(D)(E) ← (D)(E) + 1	Вміст (D)(E) збільшується на 1
INX H	1	23H	–	–	–	–	–	(H)(L) ← (H)(L) + 1	Вміст (H)(L) збільшується на 1
INX SP	1	33H	–	–	–	–	–	(SP) ← (SP) + 1	Вміст (SP) збільшується на 1
INR M	1	34H	–	+	+	+	+	(M) ← (M) + 1	Вміст комірки за адресою у M збільшується на 1
DCX B	1	0BH	–	–	–	–	–	(B)(C) ← (B)(C) – 1	Вміст (B)(C) зменшується на 1
DCX D	1	1BH	–	–	–	–	–	(D)(E) ← (D)(E) – 1	Вміст (D)(E) зменшується на 1
DCX H	1	2BH	–	–	–	–	–	(H)(L) ← (H)(L) – 1	Вміст (H)(L) зменшується на 1
DCX SP	1	3BH	–	–	–	–	–	(SP) ← (SP) – 1	Вміст (SP) зменшується на 1
DCR M	1	35H	–	+	+	+	+	(M) ← (M) – 1	Вміст комірки за адресою у M зменшується на 1
RLC	1	07H	+	–	–	–	–	$A_{m+1} \leftarrow A_m$ $A_0 \leftarrow A_1$; $(C) \leftarrow A_7$	
RRC	1	0FH	+	–	–	–	–	$A_m \leftarrow A_{m+1}$ $A_7 \leftarrow A_0$; $(C) \leftarrow A_0$	
RAL	1	17H	+	–	–	–	–	$A_{m+1} \leftarrow A_m$ $(C) \leftarrow A_7$; $A_0 \leftarrow (C)$	
RAR	1	1FH	+	–	–	–	–	$A_m \leftarrow A_{m+1}$ $A_7 \leftarrow (C)$; $(C) \leftarrow A_0$	
CMA	1	2FH	–	–	–	–	–	$(A) \leftarrow (\overline{A})$	Вміст акумулятора інвертується
CMC	1	3EH	+	–	–	–	–	$(C) \leftarrow (\overline{C})$	Вміст біта перенесення C інвертується

Продовження табл. Д 1.1

Мнемо-код	Кільк. байтів	Код	Ознаки					Алгоритм	Коментар
			C	Z	S	P	C'		
STC	1	37H	+	–	–	–	–	(C) ← 1	Біт перенесення C встановлюється у 1
DAD B	1	09	+	–	–	–	–	(H)(L) ← (H)(L) + + (B)(C)	Подвійне додавання (H)(L) з (B)(C)
DAD D	1	19	+	–	–	–	–	(H)(L) ← (H)(L) + + (D)(E)	Подвійне додавання (H)(L) з (B)(C)
DAD H	1	29	+	–	–	–	–	(H)(L) ← (H)(L) + + (H)(L)	Подвійне додавання (D)(E) з (H)(L)
DAD SP	1	39	+	–	–	–	–	(H)(L) ← (H)(L) + + (SP)	Подвійне додавання (SP) з (H)(L)
DAA	1	27	+	+	+	+	+	Десяткова корекція акумулятора	
4. Команди введення-виведення, звертання до стека й керування									
IN<b2>	2	B	–	–	–	–	–	(A) ← (Дані)	Тут <b2> є адресою пристрою введення
OUT<b2>	2	D3	–	–	–	–	–	(Дані) ← (A)	Тут <b2> є адресою пристрою виведення
PUSH B	1	C5	–	–	–	–	–	[SP – 1] ← (B) [SP – 2] ← (C) (SP) ← (SP) – 2	Вміст (B, C) пересилається у стек, указівник стека зменшується на 2
PUSH D	1	D5	–	–	–	–	–	[SP – 1] ← (D) [SP – 2] ← (E) (SP) ← (SP) – 2	Вміст (D, E) пересилається у стек, указівник стека зменшується на 2
PUSH H	1	E5	–	–	–	–	–	[SP – 1] ← (H) [SP – 2] ← (L) (SP) ← (SP) – 2	Вміст (H, L) пересилається у стек, указівник стека зменшується на 2
PUSH PSW	1	F5	–	–	–	–	–	[SP – 1] ← (A) [SP – 2] ← (F) (SP) ← (SP) – 2	Вміст (A, F) пересилається у стек, указівник стека зменшується на 2
POP B	1	C1	–	–	–	–	–	(C) ← [SP] (B) ← [SP – 1] [SP] ← [SP] + 2	Вміст (B, C) пересилається у стек, указівник стека збільшується на 2

Закінчення табл. Д 1.1

Мнемо-код	Кільк. байтів	Код	Ознаки					Алгоритм	Коментар
			C	Z	S	P	C’		
POP D	1	D1	–	–	–	–	–	(E) ← [SP] (D) ← [SP + 1] [SP] ← [SP] + 2	Вміст (D, E) пересилається у стек, указівник стека збільшується на 2
POP PSW	1	F1	+	+	+	+	+	(F) ← [SP] (A) ← [SP + 1] [SP] ← [SP] + 2	Вміст(A, F) пересилається у стек, указівник стека збільшується на 2
POP H	1	E1	–	–	–	–	–	(L) ← [SP] (H) ← [SP + 1] [SP] ← [SP] + 2	Вміст (H, L) пересилається у стек, указівник стека збільшується на 2
XTHL	1	E3	–	–	–	–	–	SP ↔ (L) – – [SP + 1] ↔ (H)	Обмін вмістом (H)(L) і [SP + 1], SP
XCHG	1	EB	–	–	–	–	–	(H) ↔ (D) – – (L) ↔ (E)	Обмін вмістом (H)(L) і (D)(E)
EI	1	FB	–	–	–	–	–		Переривання дозволено
DI	1	F3	–	–	–	–	–		Переривання заблоковано
NOP	1	0	–	–	–	–	–		Операція не виконується
HLT	1	76	–	–	–	–	–		Відбувається зупинка можливості наступного запуску при сприйнятті запитів переривання

ДОДАТОК 2

Приклади виконання завдань

Завдання 1. Викласти загальні відомості про інтерфейс КР580ВВ55. Запрограмувати інтерфейс (табл. Д 2.1, варіант 10) у режимі 0, канали *А* – введення, *В* – виведення, розряди каналу *С* 0–3 – виведення, 4–7 – введення.

Загальні відомості. Інтерфейс – сукупність апаратних і програмних засобів, а також ліній зв'язку, що забезпечують сполучення й роботу мікропроцесора з різними зовнішніми пристроями.

Структурна схема паралельного програмованого інтерфейсу (ППІ) КР580ВВ55 (рис. Д 2.1) містить у собі: двонаправлений 8-розрядний буфер даних (БФД), який з'єднує ВІС із системною шиною даних (D0–D7); три восьмирозрядних канали введення/виведення *А, В, С* для обміну інформацією; схему вибору каналів і керування; схему керування каналом.

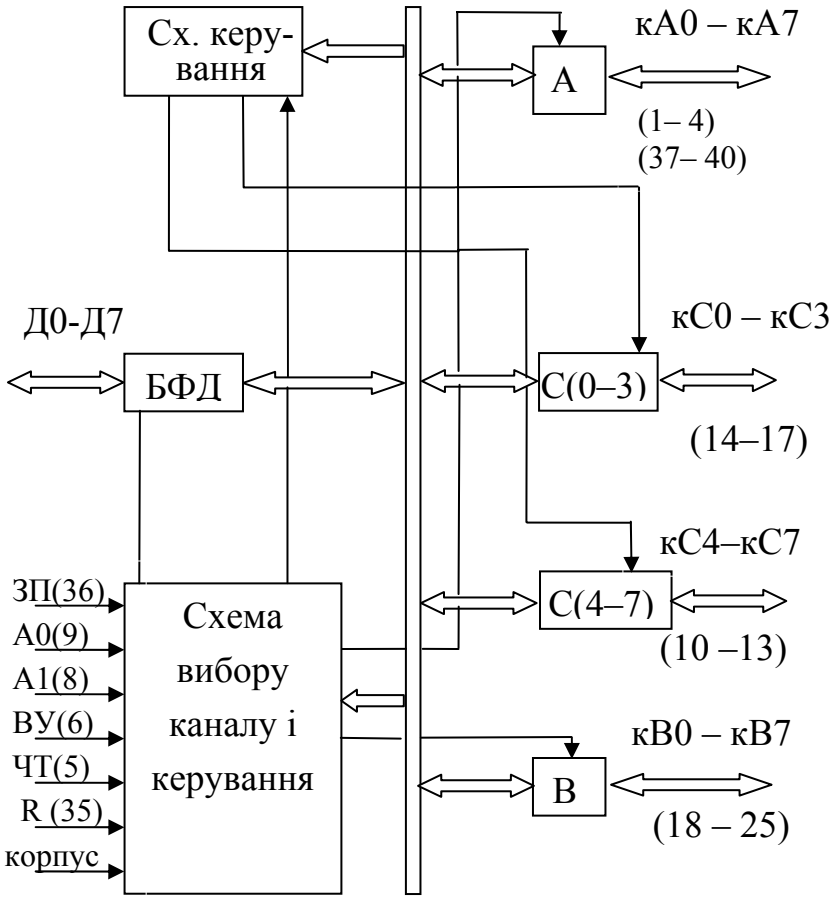


Рисунок Д 2.1 – Структурна схема паралельного інтерфейсу КР580ВВ55

Паралельний інтерфейс, що програмується (ППІ) КР580ВВ55, являє собою однокришталевий пристрій введення-виведення паралельної інформації різного формату. ППІ містить три 8-розрядні порти *A, B, C,* що можуть бути використані як для введення, так і для виведення інформації.

Порт *C* має дві відмінні особливості. Перша полягає у тому, що він має можливість роздільного установлення його розрядів у «0» або у «1», що дозволяє використовувати порт *C* для передачі керуючих сигналів. Друга особливість полягає у тому, що порт *C* розділяється на два 4-розрядних канали, що можуть використовуватись окремо.

У цьому випадку старші розряди *C*4–*C*7 об'єднуються з портом *A*, утворюючи групу *A*, молодші розряди *C*0–*C*3 – з портом *B*, утворюючи групу В. Кожну групу можна використовувати як 12-розрядні порти. Обмін інформацією МП КР580ВМ80А з портом введення-виведення здійснюється через акумулятор за командами IN «порт» і OUT «порт». Регістри ВІС КР580ВВ55 використовуються як порти введення-виведення, що програмуються шляхом запису інформації у регістр керуючого слова (РКС). Структура керуючих слів зображена на рис. Д 2.2 та рис. Д 2.3. Формат керуючого слова на рис. Д 2.2 використовується для настроювання ППІ на потрібний режим, а на рис. Д 2.3 для встановлення розрядів порта *C*. Кожен з 8-ми розрядів порта *C* може бути встановлений у «1» або у «0» за командою OUT «порт». При значенні D7 = 1 в керуючий регістр ППІ записується керуюче слово, що визначає потрібний режим роботи, а при D7 = 0, виконується встановлення розрядів порту *C*.

Напрямок передачі інформації та запис керуючих слів визначається сигналами керування роботою ППІ, які наведені у табл. Д 2.1.

Основні функції інтерфейсу:

1) буферування інформації; 2) дешифрування адреси або вибір пристрою; 3) дешифрування команди; 4) керування й синхронізація.

Режими роботи інтерфейсу:

• *режим 0* – основний режим, у якому здійснюється введення-виведення даних через три восьмирозрядних канали, причому канал *C* може розбиватися на два чотирирозрядні підканали і використатися для введення або виведення інформації. Цей режим використається при синхронному обміні або при програмній реалізації синхронного обміну.

• *режим 1* – канали *A* і *B* використовуються для введення-виведення даних. Канал *C* використовується для керування цим обміном. Цей режим забезпечує односпрямований обмін інформацією МП із пристроєм введення-виведення (ПВВ) за стробом готовності.

• *режим 2* – канал *A* працює у режимі двонаправленої шини. П'ять розрядів кн. *C* (*C*7–*C*3) використаються для керування. Цей режим забезпечує двонаправлену передачу інформації з порту *A* до зовнішнього пристрою й навпаки.

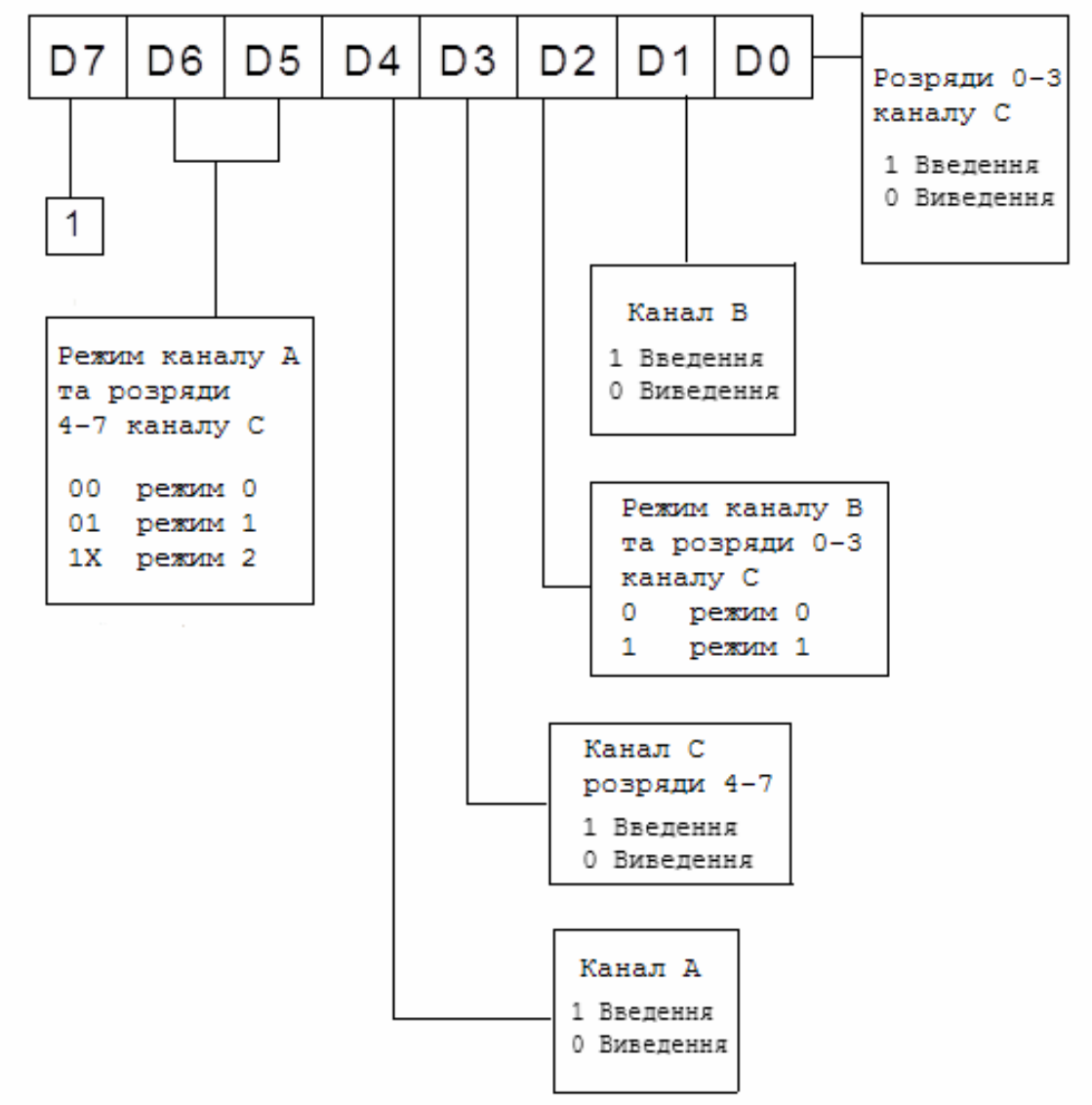


Рисунок Д 2.2 – Формат керуючого слова ППІ КР580ВВ55 для вибору режимів

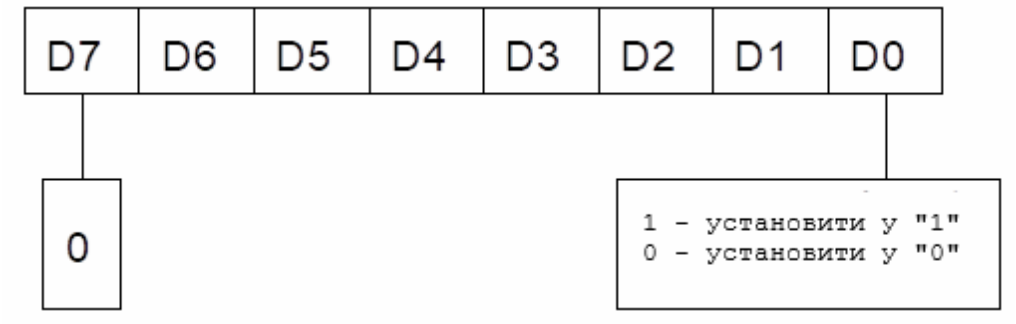


Рисунок Д 2.3 – Формат керуючого слова для встановлення розрядів порта *C*

Таблиця Д 2.1 – Сигнали керування роботою ППІ

C3	RD	WR	A1	A0	Напрямок передачі інформації
Введення з ППІ у МП					
0	0	1	0	0	РА – ШД
0	0	1	0	1	РВ – ШД
0	0	1	1	0	РС – ДВ
Виведення з МП у ППІ					
0	1	0	0	0	ШД – РА
0	1	0	0	1	ШД – РВ
0	1	0	1	0	ШД – РС
0	1	0	1	1	ШД – регістр керуючого слова
1	X	X	X	X	Відключення ППІ від ШД

Програмування інтерфейсу

Варіанти завдань для програмування інтерфейсу наведені у табл. Д 2.3.

Виходячи з умови завдання 1 (для варіанта 10) за форматом керуючого слова рис. Д 2.2 встановимо двійковий код керуючого слова (КС) 1001 1000 і переведемо його у 16-ковий код, що відповідає 98Н.

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	1	0	0	0
			9	8 H			

Заносимо керуюче слово 98Н в акумулятор (команда MVI A, 98H), а потім з акумулятора у регістр керуючого слова (РКС) командою OUT адр. РКС. Приклад програмування інтерфейсу наведено в табл. Д 2.2.

Таблиця Д 2.2 – Приклад програмування інтерфейсу

Адреса	Код	Команда	Алгоритм Коментар
8000	3E	MVI A, 98	(A) ← 98H
8001	98		
8002	D3	OUT адр. РКС	(адр.РКС) ← (A)
8003	Адр. РКС		
8004	робоча програма		

Таблиця Д 2.3 – Варіанти даних для виконання завдання з програмування паралельного інтерфейсу

№ за журн.	1	2	3	4	5	6	7	8	9	10
Режим	0	0	0	0	0	0	0	0	0	0
Канал А	Введення	Введення	Введення	Виведення	Виведення	Виведення	Введення	Виведення	Виведення	Введення
Канал В	Введення	Виведення	Виведення	Введення	Виведення	Введення	Введення	Виведення	Виведення	Виведення
Розряди 0..3 кн. С	Виведення	Введення	Введення	Виведення	Введення	Введення	Введення	Виведення	Виведення	Виведення
Розряди 4...7 кн. С	Виведення	Виведення	Введення	Виведення	Введення	Виведення	Виведення	Введення	Виведення	Введення
№ за журн.	11	12	13	14	15	16	17	18	19	20
Канал А	Виведення	Введення	Введення	Виведення	Введення	Виведення	Введення	Введення	Виведення	Виведення
Канал В	Введення	Виведення	Введення	Виведення	Виведення	Виведення	Виведення	Введення	Виведення	Введення
Розряди 0...3 кн. С	Виведення	Виведення	Введення	Введення	Виведення	Введення	Виведення	Введення	Виведення	Виведення
Розряди 4..7 кн. С	Виведення	Введення	Виведення	Введення	Виведення	Введення	Введення	Виведення	Введення	Виведення

Завдання 2. Викласти загальні відомості про таймер КР580ВІ53. Запрограмувати таймер: канал 2, режим каналу – 4, лічильник 2–10, режим ЧТ або ЗП – на льоту, число, що завантажується у лічильник $N = 5$.

Загальні відомості. Структурна схема таймера КР580ВІ53 наведена на рис. Д 2.4. Таймер має в одному корпусі три таймери-лічильники (три незалежних канали: КН0, КН1, КН2), які працюють незалежно один від одного і керуються програмою. Кожний канал містить у собі шістнадцятирозрядний лічильник (ЛЧ), *регістр режиму (РР), в який записується керуюче слово*, схему керування (СК), схему синхронізації (СС), вхід вибору мікросхеми ВМ, виводи входів (Вх0, Вх1, Вх2) і виходів (Вих0, Вих1, Вих2) та дозволи входів (Р0, Р1, Р2). Входи А0, А1 і схема вибірки каналів ВК служать для вибору одного із трьох каналів і регістрів режимів РР0–РР2.

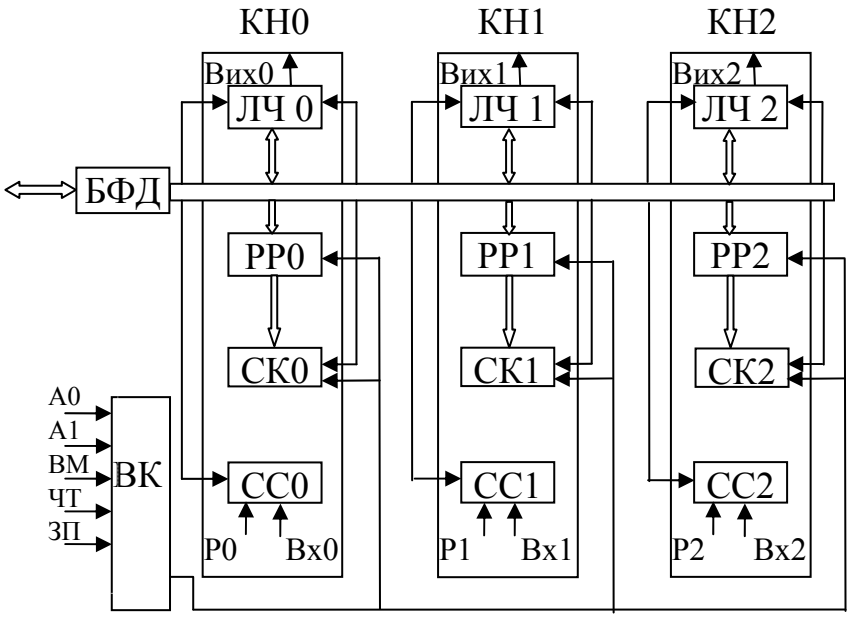


Рисунок Д 2.4 – Структурна схема таймера КР580ВІ53

Вибір одного з каналів і регістра режиму здійснюється за допомогою керуючих входів А0, А1, шляхом подачі на них відповідних двійкових кодів: 00 – лічильник ЛЧ0; 01 – лічильник ЛЧ1; 10 – лічильник ЛЧ2; 11 – регістр режиму РР.

Вхід ВМ служить для вибору мікросхеми таймера, вхід ЧТ – читання за низьким рівнем сигналу, вхід ЗП – запис за низьким рівнем сигналу. Буфер даних БФД служить для тимчасового зберігання даних.

Режими роботи таймера

Режим 0: *видача сигналу переривання за кінцевим числом.* Після встановлення режиму 0 на виході каналу з'являється рівень 0. Після завантаження числа *N* у лічильник каналу вихід залишається на рівні 0 й лічильник починає відраховувати, якщо на дозвільному вході встановлений рівень 1. Після того, як досягається встановлене число *N*, на виході встановлюється 1 і залишається доти, поки канал не буде перезавантажений новим числом або перепрограмований на новий режим роботи.

Режим 1: *програмувальний очікувальний мультивібратор*. Програмою встановлюється тривалість сигналу. У цьому режимі вихід каналу після завантаження числа *N* у лічильник установлюється рівень 0 після першого тактового сигналу, що виникає за переднім фронтом сигналу на керуючому виході. Одночасно починається рахунок. При досягненні кінцевого числа *N* на виході встановлюється 1.

Режим 2: *генератор тактових імпульсів.* У цьому режимі на виході каналу через задане число *N* періоду тактової частоти з'являється рівень 0 тривалістю в один період тактової частоти. Число періоду визначається числом *N*, записаним у лічильник каналів.

Режим 3: *генератор прямокутних сигналів.* На виході каналу буде високий рівень протягом часу, заданого числом *N*. Перша половина інтервалу – 1, друга – 0.

Режим 4: *програмувальний керований строб* (імпульс широкого функціонального призначення). Після завантаження числа *N* на виході й на дозвільному вході з'являється 1. Після чого починається рахунок і при досягненні кінцевого числа на виході з'являється 0.

Режим 5: *схемно-технічний керований строб.* Робота аналогічна режиму 4. Лічильник каналу після завантаження починає лічити по передньому фронті на керованому вході.

Програмування таймера

Варіанти завдань для програмування таймера наведені у табл. Д 2.4.

За умовою завдання 2 і форматом керуючого слова (КС) таймера, наведеним на рис. Д 2.5, складаємо 8-розрядний двійковий код керуючого

слова КС, тобто, заповнюємо розряди Д0–Д7 набором 1 і 0. Потім визначаємо відповідний йому шістнадцятковий код КС рівний 89Н і програмно заносимо його у регістр режима РР, за адресою Е3, командами MVI A, 89H і OUT E3. Після цього заноситься інформація (число N = 05) у канал лічильника ЛЧ2 за адресою Е, командами MVI A, 05 і OUT E2, де E2 – адреса каналу лічильника ЛЧ2.

У мікроконтролері МС2702 для програмованого інтервального таймера КР580ВІ53 конструктивно розроблена така адресація каналів лічильників – ЛЧ0 – Е0, ЛЧ1 – Е1, ЛЧ2 – Е2, регістра режиму – Е3.

Таблиця Д 2.4 – Варіанти даних для програмування таймера

Номер вар.	1	2	3	4	5	6	7	8	9	10
Гр. 2хА канал	2	1	0	2	0	1	1	0	2	0
Гр.2хБ канал	1	0	2	0	1	2	2	1	0	1
Режим каналу	0	1	2	3	4	5	4	3	2	1
Лічильник	2	2–10	2	2–10	2	2	2	2–10	2–10	2
Режим ЧТ або ЗП	мол. байт	ст. байт	ст. байт	мол. байт	мол. байт	мол. байт	ст. байт	на льоту	мол., ст. байт	на льоту
Номер вар.	11	12	13	14	15	16	17	18	19	20
Гр. 2хА канал	1	2	1	0	2	0	1	2	1	2
Гр.2хБ канал	0	1	2	2	0	1	2	0	2	1
Режими каналу	5	0	3	2	4	1	0	2	3	4
Лічильник	2–10	2	2–10	2	2–10	2	2–10	2	2	2–10
Режим ЧТ або ЗП	мол. байт	ст. байт	мол., ст. байт	на льоту	мол., ст. байт	мол. байт	ст. байт	мол. байт	ст. байт	на льоту

Призначення окремих розрядів КС таймера зображено на рис. Д 2.5.

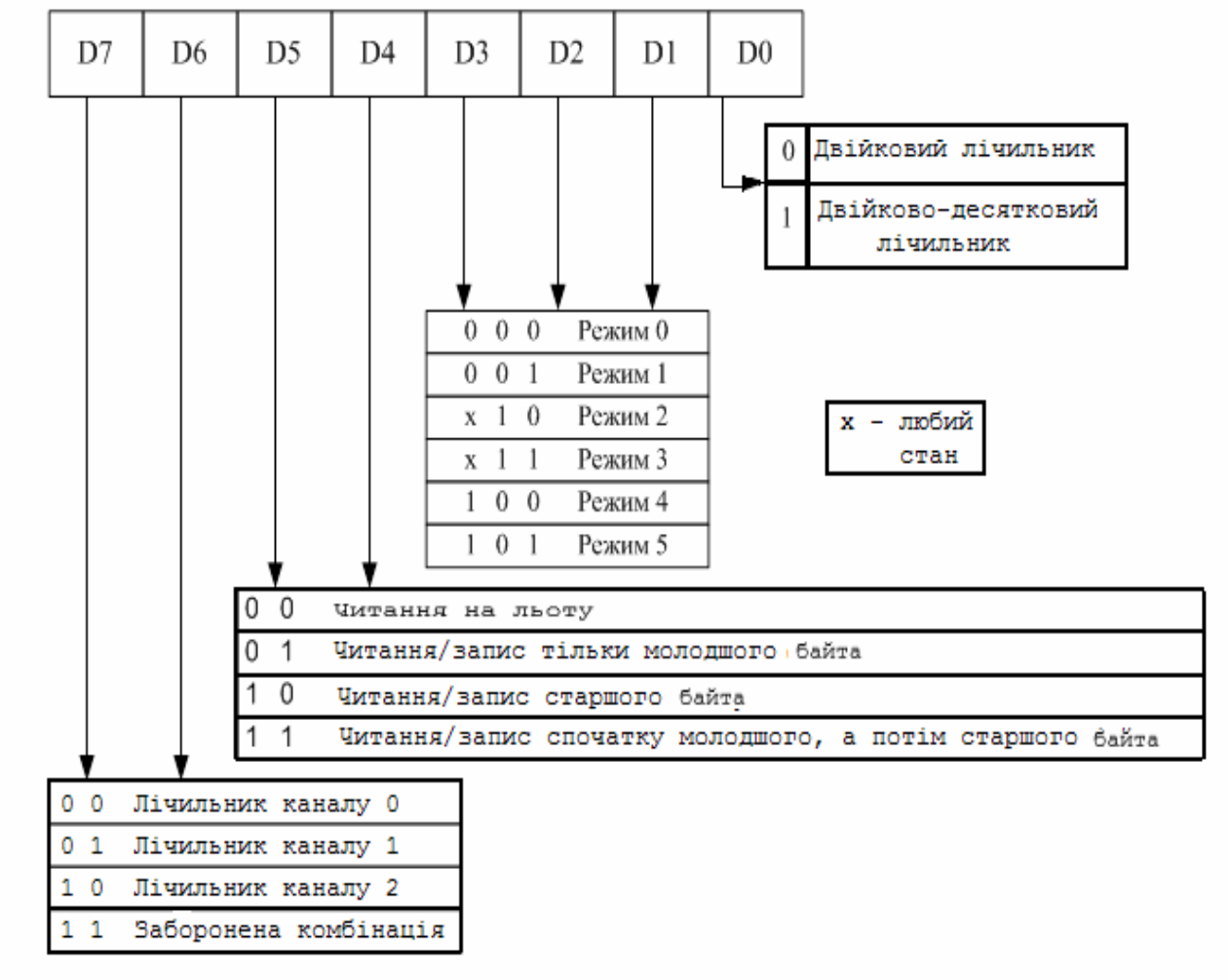


Рисунок Д 2.5 – Формат керуючого слова таймера КР580ВІ53

Відповідно до завдання 2 складаємо двійковий код керуючого слова КС і визначаємо відповідний йому шістнадцятковий код, 89Н.

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	1	0	0	1
				8	9H		

Складена програма роботи каналу 2 наведена у табл. Д 2.5.

Таблиця Д 2.5 – Програма роботи з каналом 2 таймера

Адреса	Код	Команда	Коментар
2100	3E	MVI A, 89H	(A) ← 89H
2101	89		
2102	D3	OUT E3	(E3) ← (A)
2103	E3		
2104	3E	MVI A, 05	(A) ← 05
2105	05		
2106	D3	OUT E1	(E1) ← (A)
2107	E1		

Порядок вибору каналів таймера довільний. Програмування вибраного каналу слід проводити строго за таким порядком:

1) спочатку записується керуюче слово у 16-ковому коді у РР;

2) потім заноситься інформація на адресу вибраного лічильника.

Лічильник каналів являє собою 16-розрядний лічильник з попереднім встановленням, який працює на віднімання у двійковому або двійково-десятковому коді. Оскільки лічильник працює на віднімання, то скінченним числом, на яке реагує схема, є число «00», а початковим – число, завантажене у лічильник з магістралі даних мікро-ЕОМ.

Необхідно відмітити, що адресація лічильників і регістра режиму притаманна тільки контролеру типу МС2702. Для інших систем і пристроїв на базі МПК серії 580 адресація таймера може бути (і, як правило, буває) іншою. *Існує два методи зчитування чисел.* Перший – зчитування з використанням простих операцій введення (IN, LDA) із вибраного каналу. При цьому канал призупиняє свою роботу. Другий метод – зчитування вмісту лічильника каналу без призупинення лічби, читання «на льоту». Читання при такому способі здійснюється за допомогою запису спеціальної команди на адресу регістра режиму (для МК МС 2702 – Е3).

Завдання 3. Розробити структурну схему автоматизованої системи керування технологічним процесом досліджень (АСКТПД) електричних апаратів або електропобутової техніки, з опитуванням *M* датчиків *N* раз за період, при частоті 50 Гц, на базі мікроконтролера MC2702 та скласти алгоритм і програму її роботи. Провести розрахунок швидкодії схеми.

Варіанти даних наведені у табл. Д 2.6.

Таблиця Д 2.6 – Варіанти даних

Номер варіанта	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Електричний апарат (ЕА)	ШЗ	АВ	МП	КР	РН	ШЗ	МП	КР	МП	ШЗ	АВ	МП	РН	РС
Електропобутова техніка (ЕПТ)	МХ	КП	ПМ	МВ	ІП	ПМ	ЕН	МХ	КП	ПМ	МВ	ПТ	ЕВ	ЕП
Кількість датчиків, M	5	6	7	8	3	4	8	6	7	6	5	4	5	6
Кількість опитувань, N	24	11	25	10	24	22	32	30	26	28	16	20	18	14

Примітка. У таблиці Д 2.6 прийняті такі скорочені позначення:

для ЕА: ШЗ – швидкодіючий запобіжник; АВ – автоматичний вимикач; МП – магнітний пускач; КР – контактор; РН – реле напруги; РС – реле струму;

для ЕПТ: МХ – мікрохвильова піч; КП – кондиціонер побутовий; ПМ – пральна машина; МВ – мультиварка; ІП – індукційна піч; ЕВ – електричний водонагрівач; ПС – пилотяг; ЕП – електрична плита.

Приклад змісту і структури завдання

Вступ (1 с).

1. Характеристика і аналіз об’єкта керування (ЕА або ЕПТ) та технічних вимог (3–6 с).

2. Розробка технологічного алгоритму (технологічних процесів, методів досліджень і випробувань ЕА або ЕПТ) (7–9 с).

3. Розробка і опис структурної схеми АСКТПД (2–5 с).

4. Вибір елементів структурної схеми (датчиків, комутатора, АЦП, МК, підсилювачів, компараторів, пристроїв зв’язку з об’єктом (ПЗО) та ін.) (7–9 с).

5. Побудова і опис алгоритму роботи схеми АСКТПД (2–3 с).

6. Написання програми, (1–2 с).

7. Розрахунок швидкодії АСКТПД (1–2 с).

Висновки (1 с).

Список джерел інформації (1 с).

Розробка структурної схеми АСКТПД

Структурна схема АСКТПД для 6-ти датчиків на базі мікроконтролера МС2702 наведена нижче на рис. Д 2.6.

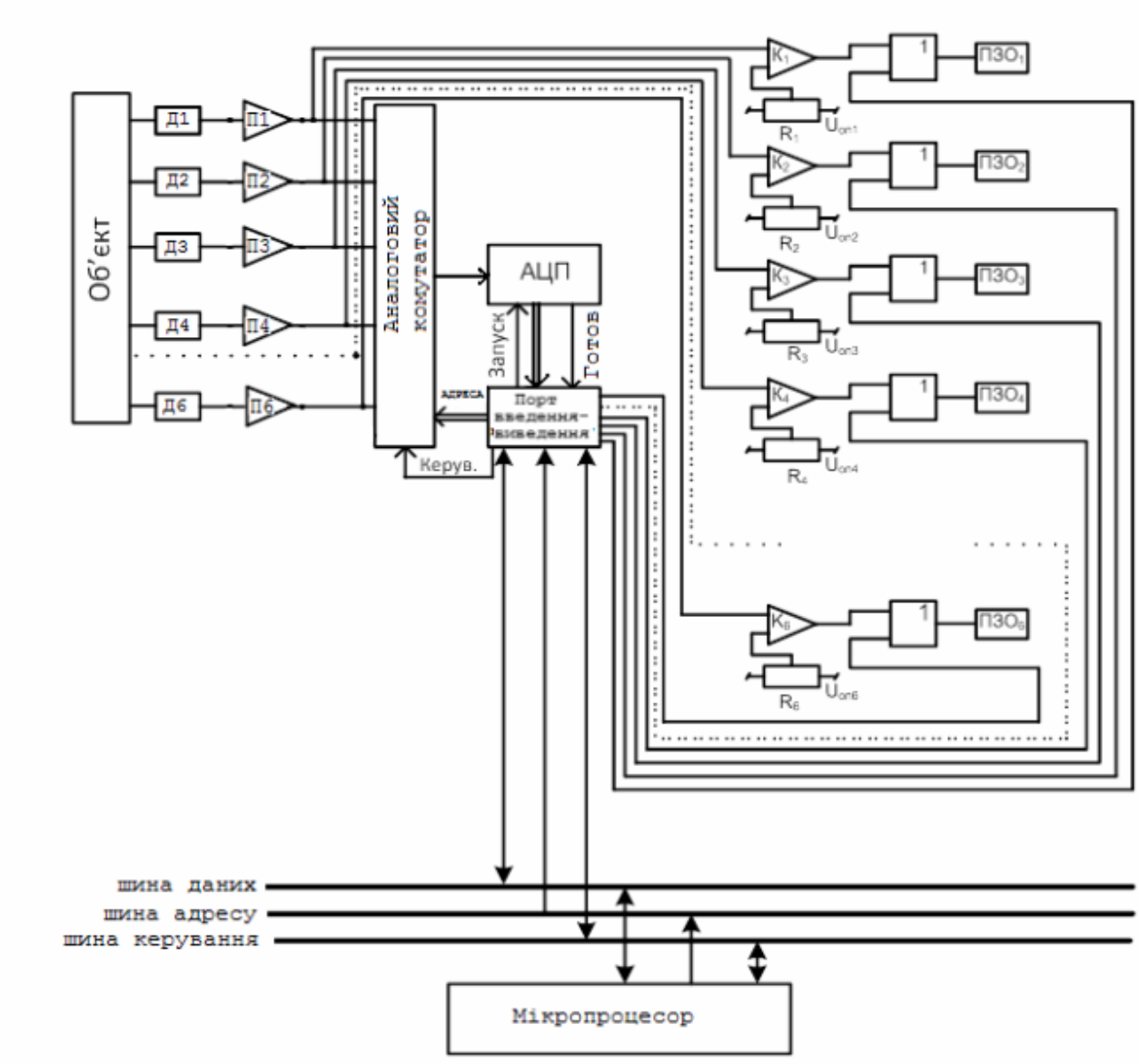


Рисунок Д 2.6 – Структурна схема АСКТПД для 6-ти датчиків на базі МК МС2702

Структурна схема включає:

- датчики (первинні перетворювачі) контрольованих параметрів Д1–Д6 (струму, напруги, температури, Джоулевого інтеграла, тиску, швидкості або інших парметрів і у більшій кількості);
- нормуючі підсилювачі П1–П6;
- багатоканальний комутатор аналогових сигналів типу КР590КИ6;
- мікроконтролер «Електроніка МС2702» що містить виконані на окремих мікросхемах і розміщені на одній платі: 8-розрядний мікропроцесор КР580; програмувальний паралельний інтерфейс КР580ВВ55; програмувальний інтервальний таймер типу КР580ВІ53, що має три таймери-лічильники; пам'ять (постійну й оперативну) із загальною ємністю 64 Кбайта; окремий пульт керування з дисплеєм і клавіатурою.
- аналого-цифровий перетворювач (АЦП) типу К1113ПВ1;
- компаратори К1–К6 типу К554СА3, виходи яких за «АБО» об'єднані з вихідними сигналами мікропроцесора керуючого мікроконтролера;
- пристрої зв'язку з об'єктом ПЗО1–ПЗО6, які включають виконавчі пристрої силової установки, що узгоджують та задають режим випробувань або досліджень.

Через інтерфейс АСКТПД можна з'єднувати із ПЕОМ та змінювати режими випробувань або досліджень, а також приймати, запам'ятовувати, відображати і документувати результати випробувань або досліджень.

До об'єкта дослідження підключені відповідні датчики. Датчики контрольованих параметрів Д1–Д6 є первинними перетворювачами струму, напруги, температури, Джоулевого інтеграла, тиску, швидкості або інших фізичних параметрів об'єкта дослідження, у напругу. У схемі АСКТПД можливе підключення значно більшої кількості датчиків як аналогових, через комутатор до АЦП, так і двійкових, безпосередньо до порту введення. Нормуючі підсилювачі погоджують вихідну напругу датчиків з необхідним вхідним сигналом АЦП 0–10 В і забезпечують низький вихідний опір.

Комутатор аналогових сигналів перемикає один із входів, залежно від керуючого коду, що надійшов від мікроконтролера, на вихід підключений до входу АЦП, що є швидкодіючим десятирозрядним перетворювачем вхідної напруги у паралельний двійковий код. Запуск АЦП проводиться мікроконтролером, закінчення перетворення викликає сигнал готовності, який є командою для прочитування даних. МК відповідно до записаної у пам'ять програми керує процесом досліджень або випробувань шляхом опитування із заданою періодичністю датчиків Д1–Д6 згідно з алгоритмом керування, наведеним на рис. Д 2.7. Вихідні сигнали датчиків унаслідок їх різної фізичної природи можуть потребувати посилення і проміжного перетворення на АЦП або на схемах формувачів сигналів (ФС), які найчастіше виконують функції гальванічної розв'язки і формування рівнів двійкових сигналів стандарту ТТЛ. МК з необхідною періодичністю оновлює керуючі слова на своїх вихідних портах. Деяка частина керуючого слова може інтерпретуватися як сукупність прямих двійкових сигналів керування (СК), що через схеми формувачів сигналів (підсилювачі потужності, реле, оптрони та ін.) або пристрої зв'язку з об'єктом (ПЗО1–ПЗО6) надходять на виконавчі механізми (ВМ). Компаратори К1–К6 є паралельним апаратним контуром для захисту від аварійних режимів. ПЗО1–ПЗО6 є підсилювачами потужності, які керують виконавчими пристроями силової установки.

Вибір МП або МК і частоти опитування датчиків проводиться виходячи з характеру таких досліджуваних процесів і умов досліджень:

– швидкості протікання процесів та завдань з переробки інформації;

– кількості досліджуваних параметрів і частоти опитування датчиків та умов експлуатації і вимог щодо надійності.

Частота опитування датчиків F_s визначається за теоремою Котельникова залежно від заданої точності і характеру вимірюваного сигналу:

$$F_s = \nu \cdot F_m,$$

де $\nu = (10 \div 50)$, F_m – максимальна частота у спектрі сигналу.

Приклад алгоритму опитування 6-ти датчиків 30 разів за період на базі мікроконтролера МС2702 наведено на рис. Д 2.7.

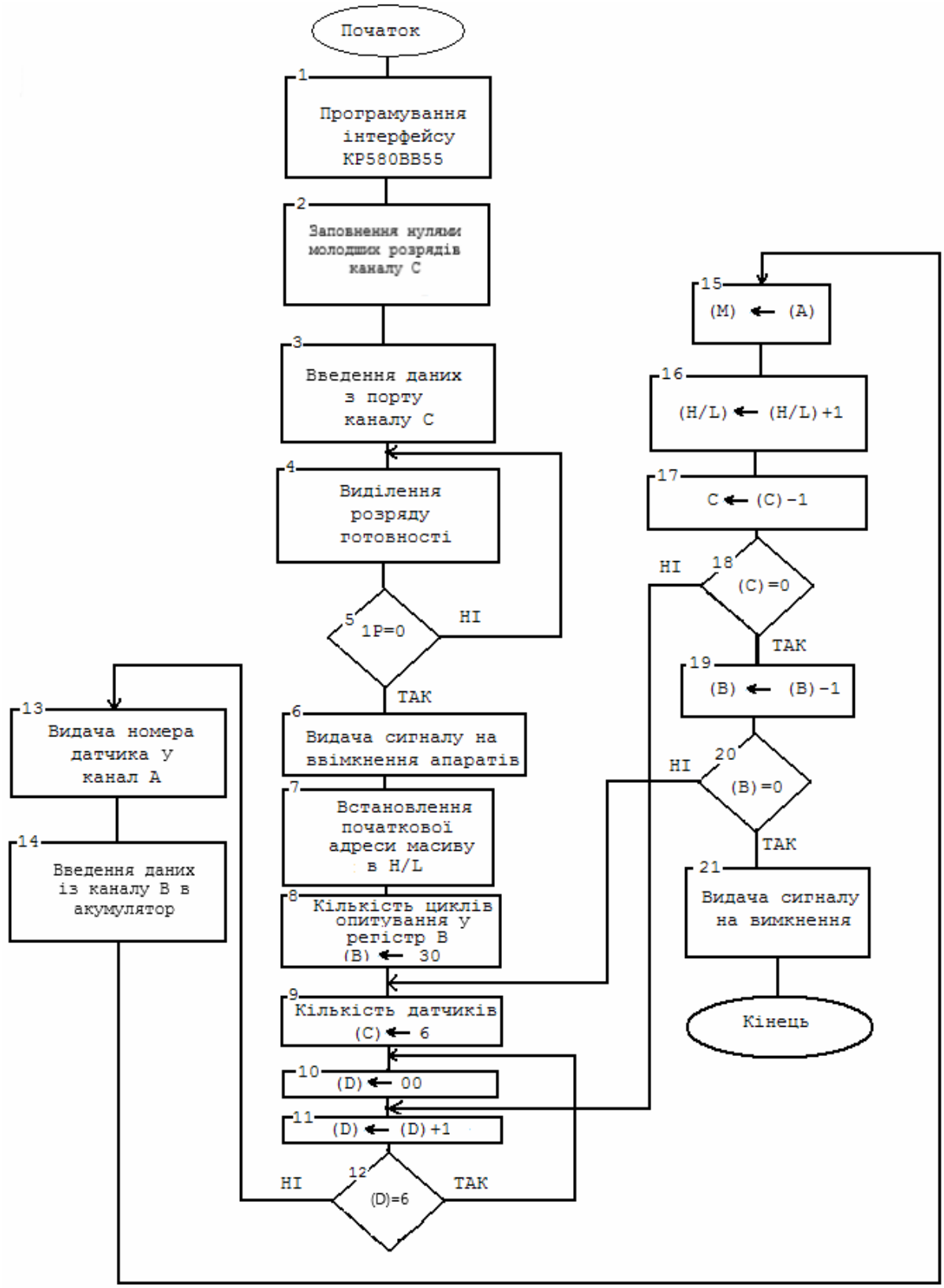


Рисунок Д 2.7 – Алгоритм опитування 6-ти датчиків 30 разів за період

Якщо для АСКТПД обрати мікроконтролер МС2702, то програма на мові Асемблер може мати вигляд наведений нижче.

Приклад програми опитування 4-х датчиків 22 рази за період

Мітка	Час	Команда	Алгоритм і коментарій команди
		MVI A, 8A	(A) ← (8A) Програмування інтерфейсу
		OUT (адр. РУС)	(РУС) ← (A) Пересилання вмісту А в РУС
		MVI A, 00	(A) ← 00 Заповнення нулями акумулятора
		OUT (адр. кн. С)	(адр. кн. С) ← (A) Пересилання вмісту А у молодші розряди канала С
A1:		IN (адр. кн. С)	Очікування сигналу готовності від оператора
		MVI B, 01	(B) ← 01 Занесення 01 в регістр В
		ANA B, 01	Операція «І» над вмістом регістра В і акум.
		JZ A1	Умовний перехід на мітку А1
		MVI A, 01	(A) ← 01 Увімкнення апаратів
		OUT (адр. кн. С)	(адр. кн. С) ← (A) Пересилання вмісту А у кн. С
		LXI H (поч. адр.),	Завантаження початкової адреси у H/L
		MVI B, 22	Кількість циклів опитування 22 у регістр В
A4:	3,5	MVI C, 04	(C)← 04 Завантаження кількості датчиків 4 у регістр С
A2:	3,5	MVI D, 00	(D)← 00, Заповнення нулями регістра D
A3:	2,5	INR D	Організація лічильника адрес датчиків у рег. D
	2,0	MOV A, D	Отриману адресу з D пересилаємо в акумулятор
	3,5	CPI 5, A2	Безпосереднє порівняння
	5,0	OUT (адр. кн.А)	Видача номера датчика у канал А
	5,0	IN (адр. кн. В)	Введення даних з порту В у акумулятор
	3,5	MOV M, A	Пересилання даних у комірку ОЗП за адресою ((M))
	2,5	INX, H	Збільшення адреси комірки ОЗП у ((M)) на 1
	2,5	DCR C	(C) ← (C) – 1 Зменшення вмісту регістра С на 1
	5,0	JNZ A3	Перевірка умови, чи всі 4 датчики опитані
	2,5	DCR B	(B) ← (B) – 1 Зменшення вмісту регістра В на 1
	5,0	JNZ A4	Перевіряємо, чи вся кількість циклів опитувань виконана
		MVI A, 00	(A) ← 00 Заповнення нулями акумулятора
		OUT (адр. кн. С)	(адр. кн. С) ← (A) Пересилання вмісту А в кн. С для видачі сигналу на відключення апаратів
		HLT	Зупинка

Приклад розрахунку швидкодії АСКТПД з одним АЦП

Завдання. Необхідно за допомогою АСКТПД зняти показання з кожного із 4-х датчиків з частотою 22 точки за період при частоті $f = 50$ Гц.

Розв'язання

1. Розраховуємо кількість звернень до датчиків

$$K = 4 \cdot 22 = 88.$$

2. Визначаємо період при частоті 50 Гц $T = 1 / f = 1 / 50 = 0{,}02$ с.

3. Визначаємо час, необхідний для опитування одного датчика і розміщення даних в ОЗП, $t_{д.}$ $t_д = 0{,}02 / 88 = 227{,}3$ мкс.

4. Із наведеної вище програми для МК MC2702 визначаємо час $t_{пр.}$ необхідний для опитування одного датчика один раз і розміщення даних в ОЗП. Для цього складемо зазначену в колонці «Час» програми, починаючи з мітки А4, тривалість виконання відповідних команд в мікросекундах (мкс) і визначимо $t_{пр} = 41{,}5$ мкс.

5. Визначаємо час переключення аналогового комутатора А612–20 $t_к = 1$ мкс та час перетворення АЦП А611–21/2 $t_{АЦП} = 15$ мкс.

6. Швидкодія АСУ ТПД у цілому t_c визначиться як $t_c = t_{пр} + t_к + t_{АЦП}$

$t_c = 41{,}5 + 1 + 15 = 57{,}5$ мкс.

Висновок. Оскільки $t_с$ значно менше $t_д = 227{,}3$ мкс, то за швидкодією система задовольняє вимогам.

У випадку, якщо $t_с$ більше $t_д$, тобто за швидкодією система не задовольняє вимогам, то для зменшення $t_с$ необхідно вибрати більш швидкодіючий і високопродуктивний мікроконтролер, або зменшити кількість датчиків і частоту їх опитувань. Якщо і після цих заходів схема АСКТПД з одним АЦП не забезпечує належної швидкодії, тоді рекомендується застосовувати схему АСКТПД з паралельними АЦП (див. підрозділ 4.4, ст. 310).

ДОДАТОК 3

Система команд PIC16X7XX

Таблиця Д 3.1 – Опис полів коду операції

Поле	Опис
F	Адреса регістра (0x00...0x7f)
W	Робочий регістр (акумулятор)
B	Адреса розряду всередині 8-розрядного регістра
K	Постійне число або значення літерала (мітка)
X	Невизначене значення (= 0 або 1) Асемблер генерує код $x = 0$, рекомендовано для сумісності з усіма інструментальними засобами програмного забезпечення
D	Указівник підсумку; $d = 0$: підсумок зберігається у W, $D = 1$: підсумок зберігається у регістрі f. За замовчуванням значення $d = 1$
label	Ім’я мітки
TOS	Верхня комірка стека Программный счетчик Регистр старших разрядов программного счетчика
PC	Програмний лічильник Регистр старших разрядов программного счетчика
PCLATH	Регістр старших розрядів програмного лічильника
GIE	Біт загального дозволу переривань
WDT	Сторожовий таймер
TO	Біт закінчення лічення сторожового таймера
PD	Біт скидання при увімкненні живлення
dest	Підсумок, регістр W або початковий регістр
[ ]	Додаткові параметри
( )	Вміст
$\rightarrow$	Призначення
< >	Указівник розряду
$\in$	Належить множині
Курсив	Термін, визначений користувачем (шрифт– *Курсив*)

Таблиця Д 3.2 – Система команд PIC16X7XX

Мнемоніка команди	Операція	Цикли	Код операції MSb LSb	Прапорці	Примітка
Байтові операції з регістрами					
ADDWF t, d	Складання ($W + f \rightarrow d$)	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF t, d	Логічне «І» (W *and* $f \rightarrow d$)	1	00 0101 dfff ffff	Z	1,2
CLRF f	Заповнити нулями f	1	00 0001 1fff ffff	Z	2
CLRW -	Заповнити нулями W	1	00 0001 0xxx xxxx	Z	
COMF t, d	Доповнення f (Логічне «НІ»)	1	00 1001 dfff ffff	Z	1,2
DECF t, d	Декремент f	1	00 0011 dfff ffff	Z	1,2
DECFSZ t, d	Декремент f, пропуск, якщо «0»	1(2)	00 1011 dfff ffff		1,2,3
INCF t, d	Інкремент f	1	00 1010 dfff ffff	Z	1,2
INCFSZ t, d	Інкремент f, пропуск, якщо «0»	1(2)	00 1111 dfff ffff		1,2,3
IORWF t, d	Логічне «АБО» (W *or* $f \rightarrow d$)	1	00 0100 dfff ffff	Z	1,2
MOVF t, d	Пересилання ($f \rightarrow W$)	1	00 1000 dfff ffff	Z	1,2
MOVWF f	Пересилання ($W \rightarrow f$)	1	00 0000 1fff ffff		
NOP -	Пуста операція	1	00 0000 0xx0 0000		
RLF t, d	Зсув вліво через перенесення	1	00 1101 dfff ffff	C	1,2
RRF t, d	Зсув вправо через перенесення	1	00 1100 dfff ffff	C	1,2
SUBWF t, d	Віднімання ($f - W \rightarrow d$)	1	00 0010 dfff ffff	C,CD,Z	1,2
SWAPF t, d	Обміняти напівбайти f	1	00 1110 dfff ffff		1,2
XOFWF t, d	Виключаюче «АБО» (W *or* $f \rightarrow d$)	1	00 0110 dfff ffff	Z	1,2
Бітові операції з регістром					
BCF t, d	Заповнити нулями біт b у f	1	01 00bb dfff ffff		1,2
BSF t, d	Встановити біт b у f	1	01 01bb dfff ffff		1,2
BTFSC t, d	Тест біта b у f, пропуск, якщо «0»	1(2)	01 10bb dfff ffff		3
BTFSS t, d	Тест біта b у f пропуск, якщо «1»	1(2)	01 11bb dfff ffff		3

Закінчення табл. Д 3.2

Мнемоніка команди	Операція	Цикли	Код операції MSb LSb	Прапорці	Примітка
Операції з константами і командами керування					
ADDLW k	Складання ($k + W \rightarrow W$)	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW k	Логічне «І» (k *and* $W \rightarrow W$)	1	11 1001 kkkk kkkk	Z	
CALL k	Виклик підпрограми k	2	10 0kkk kkkk kkkk		
CLRWDT -	Заповнити нулями *WDT*	1	00 0000 0110 0100		
GOTO k	Перехід за адресою k	2	10 1kkk kkkk kkkk		
IORLW k	Логічне «АБО» (k *or* $W \rightarrow W$)	1	11 1000 kkkk kkkk	Z	
MOVLW k	Пересилання ($k \rightarrow W$)	1	11 00xx kkkk kkkk		
RETFIE -	Повернення з переривання	2	00 0000 0000 1001		
RETLW k	Повернення з константою	2	11 01xx kkkk kkkk		
RETURN -	Повернення з підпрограми	2	00 0000 0000 1000		
SLEEP -	Зупинка	1	00 0000 0110 0011	TO,PD	
SUBLW k	Віднімання ($k - W \rightarrow W$)	1	11 110x kkkk kkkk	C,DCY	
XORLW k	Виключаюче «АБО» (k *xor*) $W \rightarrow W$)	1	11 1010 kkkk kkkk	Z	

Примітка

1. Усі команди виконуються як «читання – модифікація – запис». Якщо змінюється регістр I/O, то вихідним значенням буде величина, прочитана безпосередньо з контактів. Наприклад, якщо у регістрі даних «1», а контакти сконфігуровані як входи і зовнішній пристрій встановить низький рівень, то до регістра даних будуть записані «0».

2. Якщо команда виконана над регістром TMR0 (коли $d = 1$, результат записується до регістра таймера 0), то переддільник буде заповнений нулями.

Якщо програмний лічильник (PC) змінюється або результат перевірки умови істинний, то команда виконується за два цикли. Другий цикл виконується як команда NOP.

ДОДАТОК 4

Таблиця Д 4.1 – Команди Асемблера МК51

Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
Група команд передачі даних						
Пересилання в акумулятор з регістра (n = 0÷7)	MOV A, Rn	11101rrr	1	1	1	(A) ← (Rn)
Пересилання в акумулятор прямоадресованого байта	MOV A, ad	11100101	3	2	1	(A) ← (ad)
Пересилання в акумулятор байта з РПД (i = 0,1)	MOV A, @ Ri	1110011i	1	1	1	(A) ← ((Ri))
Завантаження в акумулятор константи	MOV A, # d	01110100	2	2	1	(A) ← #d
Пересилання в регістр із акумулятора	MOV Rn, A	11111rrr	1	1	1	(Rn) ← (A)
Пересилання в регістр прямоадресованого байта	MOV Rn, ad	10101rrr	3	2	2	(Rn) ← (ad)
Завантаження в регістр константи	MOV Rn, # d	01111rrr	2	2	1	(Rn) ← #d
Пересилання за прямою адресою акумулятора	MOV ad, A	11110101	3	2	1	(ad) ← (A)
Пересилання за прямою адресою регістра	MOV ad, Rn	10001rrr	3	2	2	(ad) ← (Rn)
Пересилання прямоадресованого байта за прямою адресою	MOV add, ads	10000101	9	3	2	(add) ← (ads)
Пересилання байта з РПД за прямою адресою	MOV ad, @ Ri	1000011i	3	2	2	(ad) ← ((Ri))
Пересилання за прямою адресою константи	MOV ad, # d	01110101	7	3	2	(ad) ← # d
Пересилання у РПД із акумулятора	MOV @ Ri, A	1111011i	1	1	1	((Ri)) ← (A)
Пересилання у РПД прямоадресованого байта	MOV @ Ri, ad	0110011i	3	2	2	((Ri)) ← (ad)
Пересилання у РПД константи	MOV @ Ri, # d	0111011i	2	2.	1	((Ri)) ← #d
Завантаження указівника даних	MOV DPTR, # d16	10010000	13	3	2	(DPTR) ← #d16
Пересилання в акумулятор байта із ПП	MOVC A, @ A + DPTR	10010011	1	1	2	(A) ← ((A) + (DPTR))
Пересилання в акумулятор байта із ПП	MOVC A, @ A + PC	10000011	1	1	2	(PC) ← (PC) + 1 (A) ← ((A) + (PC))
Пересилання в акумулятор байта із ЗПД	MOVX A, @ Ri	1110001i	1	1	2	(A) ← ((Ri))

Продовження табл. Д 4.1

Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
Пересилання в акумулятор байта з розширеної ЗПД	MOVX A,@ DPTR	11100000	1	1	2	(A) ← ((DPTR))
Пересилання у ЗПД із акумулятора	MOVX @ Ri, A	1111001i	1	1	2	((Ri)) ← (A)
Пересилання у розширену ЗПД із акумулятора	MOVX @ DPTR, A	11110000	1	1	2	((DPTR)) ← (A)
Завантаження у стек	PUSH ad	11000000	3	2	2	(SP) ← (SP) + 1 ((SP)) ← (ad)
Витяг зі стека	POP ad	11010000	3	2	2	(ad) ← (SP) (SP) ← (SP) – 1
Обмін акумулятора з регістром	XCH A, Rn	11001rrr	1	1	1	(A) ↔ (Rn)
Обмін акумулятора з прямоадресованим байтом	XCH A, ad	11000101	3	2	1	(A) ↔ (ad)
Обмін акумулятора з байтом із РПД	XCH A, @ Ri	1100011i	1	1	1	(A) ↔ ((Ri))
Обмін молодшої тетради акумулятора з молодшою прямоадр байта РПД	XCHD A, @ Ri	1101011i	1	1	1	$(A_{0–3})$ ↔ ((Ri)$(A_{0–3})$
Група команд арифметичних операцій						
Додавання акумулятора з регістром (n = 0÷7)	ADD A, Rn	00101rrr	1	1	1	(A) ← (A) + (Rn)
Додавання акумулятора з прямоадресованим байтом	ADD A, ad	00100101	3	2	1	(A) ← (A) + (ad)
Додавання акумулятора з байтом із РПД (i = 0,1)	ADD A, @ Ri	0010011i	1	1	1	(A) ← (A) + ((Ri))
Додавання акумулятора з константою	ADD A, # d	00100100	2	2	1	(A) ← (A) + # d
Додавання акумулятора з регістром і переносом	ADDC A, Rn	00111rrr	1	1	1	(A) ← (A) + (Rn) + (C)
Додавання акумулятора з прямоадресованим байтом і переносом	ADDC A, ad	00110101	3	2	1	(A) ← (A) + (ad) + (C)
Додавання акумулятора з байтом із РПД і переносом	ADDC A, @ Ri	0011011i	1	1	1	(A) ← (A) + ((Ri)) + (C)
Додавання акумулятора з константою й переносом	ADDC A, # d	00110100	2	2	1	(A) ← (A) + # d + (C)
Десяткова корекція акумулятора	DA A	11010100	1	1	1	Якщо $(A_{0–3})$ > 9 V ((AC) = 1), то $(A_{0–3})$ ← $(A_{0–3})$ + 6, потім якщо $(A_{4–7})$ > 9V((C≠1), то $(A_{4–7})$ ← $(A_{4–7})$ + 6

Продовження табл. Д 4.1

Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
Віднімання з акумулятора регістра й позички	SUBB A, Rn	10011rrr	1	1	1	(A) ← (A) – (C) – (Rn)
Віднімання з акумулятора прямоадресованого байта й позички	SUBB A, ad	10010101	3	2	1	(A) ← (A) – (C) – ((ad))
Віднімання з акумулятора байта РПД і позички	SUBB A, @ Ri	1001011i	1	1	1	(A) ← (A) – (C) – ((Ri))
Віднімання з акумулятора константи й позички	SUBB A, d	10010100	2	2	1	(A) ← (A) – (C) – # d
Інкремент акумулятора	INC A	00000100	1	1	1	(A) ← (A) + 1
Інкремент регістра	INC Rn	00001rrr	1	1	1	(Rn) ← (Rn) + 1
Інкремент прямоадресованого байта	INC ad	00000101	3	2	1	(ad) ← (ad) + 1
Інкремент байта у РПД	INC @ Ri	0000011i	1	1	1	((Ri)) ← ((Ri)) + 1
Інкремент □рямо адре даних	INC DPTR	10100011	1	1	2	(DPTR) ← (DPTR) + 1
Декремент акумулятора	DEC A	00010100	1	1	1	(A) ← (A) – 1
Декремент регістра	DEC Rn	00011rrr	1	1	1	(Rn) ← (Rn) – 1
Декремент прямоадресованого байта	DEC ad	00010101	3	2	1	(ad) ← (ad) – 1
Декремент байта в РПД	DEC @ Ri	0001011i	1	1	1	((Ri)) ← ((Ri)) – 1
Множення акумулятора на регістр B	MUL AB	10100100	1	1	4	(B)(A) ← (A) × (B)
Ділення акумулятора на регістр B	DIV AB	10000100	1	1	4	(A).(B) ← (A) / (B)
Група команд логічних операцій						
Логічне І акумулятора й регістра	ANL A, Rn	01011rrr	1	1	1	(A) ← (A) Λ (Rn)
Логічне І акумулятора й □рямо адресованого байта	ANL A, ad	01010101	3	2	1	(A) ← (A) Λ (ad)
Логічне І акумулятора й байта із РПД	ANL A, @ Ri	0101011i	1	1	1	(A) ← (A) Λ ((Ri))
Логічне І акумулятора й константи	ANL A, #d	01010100	2	2	1	(A) ← (A) Λ # d
Логічне І прямоадресованого байта й акумулятора	ANL ad, A	01010010	3	2	1	(ad) ← (ad) Λ (A)
Логічне І прямоадресованого байта й константи	ANL ad, #d	01010011	7	3	2	(ad) ← (ad) Λ # d
Логічне АБО акумулятора й регістра	ORL A, Rn	01001rrr	1	1	1	(A) ← (A) V (Rn)
Логічне АБО акумулятора й прямоадресованого байта	ORL A, ad	01000101	3	2	1	(A) ← (A) V (ad)

Продовження табл. Д 4.1

Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
Логічне АБО акумулятора й байта із РПД	ORL A, @ Ri	0100011i	1	1	1	(A) ← (A) V ((Ri))
Логічне АБО акумулятора й константи	ORL A, # d	01000100	2	2	1	(A) ← (A) V # d
Логічне АБО прямоадресованого байта й акумулятора	ORL ad, A	01000010	3	2	1	(ad) ← (ad) V (A)
Логічне АБО прямоадресованого байта й константи	ORL ad, # d	01000011	7	3	2	(ad) ← (ad) V # d
Виключаюче АБО акумулятора й регістра	XRL A, Rn	01101rrr	1	1	1	(A) ← (A) ∀ (Rn)
Виключаюче АБО акумулятора й прямоадресованого байта	XRL A, ad	01100101	3	2	1	(A) ← (A) ∀ (ad)
Виключаюче АБО акумулятора й байта із РПД	XRL A, @ Ri	0110011i	1	1	1	(A) ← (A) ∀ ((Ri))
Виключаюче АБО акумулятора й константи	XRL A, # d	01100100	2	2	1	(A) ← (A) ∀ # d
Виключаюче АБО прямоадресованого байта і акумулятора	XRL ad, A	01100010	3	2	1	(ad) ← (ad) ∀ (A)
Виключаюче АБО прямо адресованого байта й константи	XRL ad, # d	01100011	7	3	2	(ad) ← (ad) ∀ # d
Скидання акумулятора	CLR A	11100100	1	1	1	(A) ← 0
Інверсія акумулятора	CPL A	11110100	1	1	1	(A) ← ($\overline{A}$)
Зсув акумулятора вліво циклічне	RL A	00100011	1	1	1	(A_{n+1}) ← (A_n), n= 0–6, (A_0) ← (A_7)
Зсув акумулятора вліво через перенесення	RLC A	00110011	1	1	1	(A_{n+1}) ← (A_n), n = 0–6, (A_0) ← ©, © ← (A_7)
Зсув акумулятора вправо циклічне	RR A	00000011	1	1	1	(A_n) ← (A_{n+1}), n = 0–6, (A_7) ← (A_0)
Зсув акумулятора вправо через перенесення	RRC A	00010011	1	1	1	(A_n) ← (A_{n+1}), n = 0–6, (A_7) ← ©, © ← (A_0)
Обмін місцями тетрад в акумуляторі	SWAP A	11000100	1	1	1	($A_{0–3}$) ↔ ($A_{4–7}$)

Продовження табл. Д 4.1

Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
Група команд операцій з бітами						
Скидання перенесення	CLR C	11000011	1	1	1	(C) ← 0
Скидання біта	CLR bit	11000010	4	2	1	(b) ← 0
Установлення перенесення	SETB C	11010011	1	1	1	(C) ← 1
Установлення біта	SETB bit	11010010	4	2	1	(b) ← 1
Інверсія перенесення	CPL C	10110011	1	1	1	(C) ← (C)
Інверсія біта	CPL bit	10110010	4	2	1	(b) ← (b)
Логічне І біта та перенесення	ANL C, bit	10000010	4	2	2	(C) ← (C) Λ (b)
Логічне І інверсії біта та перенесення	ANL C, / bit	10110000	4	2	2	(C) ← (C) Λ ($\overline{b}$)
Логічне АБО біта й перенесення	ORL C, bit	01110010	4	2	2	(C) ← (C) V (b)
Логічне АБО інверсії біта й перенесення	ORL C, / bit	10100000	4	2	2	(C) ← (C) V ($\overline{b}$)
Пересилання біта в перенесення	MOV C, bit	10100010	4	2	1	(C) ← (b)
Пересилання перенесення у біт	MOV bit, C	10010010	4	2	2	(b) ← (C)
Група команд передачі керування						
Довгий перехід у повному обсязі пам'яті програм	LJMP ad 16	00000010	12	3	2	(PC) ← ad 16
Абсолютний перехід усередині сторінки у 2 Кбайта	AJMP ad 11	$a_{10}a_9a_8$00001	6	2	2	(PC) ← (PC) + 2, ($PC_{0–10}$) ← ad 11
Короткий відносний перехід усередині сторінки у 256 байт	SJMP rel	10000000	5	2	2	(PC) ← (PC) + 2, (PC) ← (PC) + rel
Непрямий відносний перехід	JMP @A+DPTR	01110011	1	1	2	(PC) ← (A) + (DPTR)

Продовження табл. Д 4.1

Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
Перехід, якщо акумулятор дорівнює нулю	JZ rel	01100000	5	2	2	(PC) ← (PC) + 2, якщо (A) = 0, то (PC) ← (PC) + rel
Перехід, якщо акумулятор не дорівнює нулю	JNZ rel	01110000	5	2	2	(PC) ← (PC) + 2, якщо (A) ≠ 0, то (PC) ← (PC) + rel
Перехід, якщо перенесення дорівнює одиниці	JC rel	01000000	5	2	2	(PC) ← (PC) + 2, якщо (C) = 1, то (PC) ← (PC) + rel
Перехід, якщо перенесення дорівнює нулю	JNC rel	01010000	5	2	2	(PC) ← (PC) + 2, якщо (C) = 0, то (PC) ← (PC) + rel
Перехід, якщо біт дорівнює одиниці	JB bit, rel	00100000	11	3	2	(PC) ← (PC) + 3, якщо (b) = 1, то (PC) ← (PC) + rel
Перехід, якщо біт дорівнює нулю	JNB bit, rel	00110000	11	3	2	(PC) ← (PC) + 3, якщо (b) = 0, то (PC) ← (PC) + rel
Перехід, якщо біт установлений, з наступним скиданням біта	JBC bit, rel	00010000	11	3	2	(PC) ← (PC) + 3, якщо (b) = 1, то (PC) ← (PC) + rel i (b) ← (0)
Декремент регістра й перехід, якщо не нуль	DJNZ Rn, rel	11011rrr	5	2	2	(PC) ← (PC) + 2, (Rn) ← (Rn) – 1, якщо (Rn) ≠ 0, то (PC) ← (PC) + rel
Декремент прямоадресованого байта й перехід, якщо не нуль	DJNZ ad, rel	11010101	8	3	2	(PC) ← (PC) + 2, (ad) ← (ad) – 1, якщо (ad) ≠ 0, то (PC) ← (PC) + rel
Порівняння акумулятора із прямоадресованим байтом і перехід, якщо не дорівнює	CJNE A, ad, rel	10110101	8	3	2	(PC) ← (PC) + 3, якщо (A) ≠ (ad), то (PC) ← (PC) + rel, якщо (A) < (ad), то (C) ← 1, інакше (C) ← 0
Порівняння акумулятора з константою й перехід, якщо не дорівнює	CJNE A, # d, rel	10110100	10	3	2	(PC) ← (PC) + 3, якщо (A) ≠ # d, то (PC) ← (PC) + rel, якщо (A) < # d, то (C) ← 1, інакше (C) ← 0
Порівняння регістра з константою й перехід, якщо не дорівнює	CJNE Rn, # d, rel	10111rrr	10	3	2	(PC) ← (PC) + 3, якщо (Rn) ≠ # d, то (PC) ← (PC) + rel, якщо (Rn) < # d, то (C) ← 1, інакше (C) ← 0

Закінчення табл. Д 4.1

Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
Порівняння байта у РПД із константою й перехід, якщо не дорівнює	CJNE @ Ri, # d, rel	1011011i	10	3	2	(PC) ← (PC) + 3, якщо ((Ri)) ≠ # d, то (PC) ← (PC) + rel, якщо ((Ri)) < # d, то (C) ← 1, інакше (C) ← 0
Довгий виклик підпрограми	LCALL ad 16	00010010	12	3	2	(PC) ← (PC) + 3, (SP) ← (SP) + 1, ((SP)) ← (PC_{0-7}), (SP) ← (SP) + 1, ((SP)) ← (PC_{8-15}), (PC) ← ad16
Абсолютний виклик підпрограми у межах сторінки у 2 Кбайт	ACALL ad 11	$a_{10}a_9a_8$10001	6	2	2	(PC) ← (PC) + 2, (SP) ← (SP) + 1, ((SP)) ← (PC_{0-7}), (SP) ← (SP) + 1, ((SP)) ← (PC_{8-15}), (PC_{0-10}) ← ad11
Повернення з підпрограми	RET	00100010	1	1	2	(PC_{8-15}) ← ((SP)), (SP) ← (SP) – 1, (PC_{0-7}) ← ((SP)), (SP) ← (SP) – 1
Повернення з підпрограми оброблення переривання	RETI	00110010	1	1	2	(PC_{8-15}) ← ((SP)), (SP) ← (SP) – 1, (PC_{0-7}) ← ((SP)), (SP) ← (SP) – 1
Нема операції	NOP	00000000	1	1	1	(PC) ← (PC) + 1

ДОДАТОК 5

Система команд МК сімейства AVR

Прийняті позначення

Регістр статусу (SREG):

SREG: Регістр статусу
C: Прапор перенесення
Z: Прапор нульового значення
N: Прапор негативного значення
V: Прапор-указівник переповнювання доповнення до двох
S: NEV, для перевірок зі знаком
H: Прапор напівперенесення
T: Прапор пересилання, використовуваний командами BLD і BST
I: Прапор дозволу/заборони глобального переривання

Регістри і операнди:

Rd: Регістр призначення (і джерело) в регістровому файлі
Rr: Регістр джерело у регістровому файлі
R: Результат виконання команди
K: Літерал або байт даних (8 біт)
k: Дані адреси константи для лічильника програм
b: Біт у регістровому файлі або I/O регістр (3 біта)
s: Біт у регістрі статусу (3 біта)
X, Y, Z: Регістр непрямої адресації (X = R27 : R26, Y = R29 : R28, Z = R31 : R30)
P: Адреса I/O порту
q: Зсув при прямій адресації (6 біт)

I/O регістри

RAMPX, RAMPY, RAMPZ: Регістри, зв'язані з X Y і Z регістрами, що забезпечують непряму адресацію усієї області
НОЗП мікроконтролера з об'ємом НОЗП більше 64 Кбайт.

Стек:

STACK: Стек для адреси повернення і опущених у стек регістрів
SP: Указівник стека

Прапори:

<=> Прапор, на який впливає команда
0: Очищений командою прапор
1: Встановлений командою прапор
–: Прапор, на який не впливає команда

Таблиця Д 5.1 – Опис команд

Команди умовних переходів

Тестування умови	Булевий вираз	Мнемоніка	Комплементарна умова	Булевий вираз	Мнемоніка	Коментар
Rd > Rr	$Z \cdot (N \oplus V) = 0$	BRLT*	Rd ≤ Rr	$Z + (N \oplus V) = 1$	BRGE*	Зі знаком
Rd ≥ Rr	$(N \oplus V) = 0$	BRGE	Rd < Rr	$(N \oplus V) = 1$	BRLT	Зі знаком
Rd = Rr	Z = 1	BREQ	Rd = Rr	Z = 0	BRNE	Зі знаком
Rd ≤ Rr	$Z + (N \oplus V) = 1$	BRGE*	Rd > Rr	$Z \cdot (N \oplus V) = 0$	BRLT*	Зі знаком
Rd < Rr	$(N \oplus V) = 1$	BRLT	Rd ≥ Rr	$(N \oplus V) = 0$	BRGE	Зі знаком
Rd Rr	C + Z = 0	BRLO*	Rd ≤ Rr	C + Z = 1	BRSH*	Без знака
Rd > Rr	C = 0	BRSH/ BRCC	Rd < Rr	C = 1	BRLO/ BRCS	Без знака
Rd = Rr	Z = 1	BREQ	Rd = Rr	Z = 0	BRNE	Без знака
Rd ≤ Rr	C + Z = 1	BRSH*	Rd > Rr	C + Z = 0	BRLO*	Без знака
Rd < Rr	C = 1	BRLO/ BRCS	Rd ≥ Rr	C = 0	BRSH/ BRCC	Без знака
Перенесення	C = 1	BRCS	Немає перенесення	C = 0	BRCC	Простий
Негативне значення	N = 1	BRMI	Позитивне значення	N = 0	BRPL	Простий
Переповнювання	V = 1	BRVS	Немає переповнювання	V = 0	BRVC	Простий
Нульове значення	Z = 1	BREQ	Ненульове значення	Z = 0	BRNE	Простий

Примітка. Зміна Rd і Rr при операції перед тестуванням, тобто CP Rd, Rr → CP Rr, Rd

Продовження табл. Д 5.1

Мнемоніка	Операнди	Опис	Операція	Прапори	Кіл-ть циклів
Арифметичні і логічні команди					
ADD	Rd, Rr	Add without Carry – Скласти без перенесення	Rd ← Rd + Rr	Z, C, N, V, H	1
ADC	Rd, Rr	Add with Carry – Скласти з перенесенням	Rd ← Rd + Rr + C	Z, C, N, V, H	1
ADIW	Rd, K	Add Immediate to Word – Скласти безпосереднє значення зі словом	Rdh : Rdl ← Rdh : : Rdl + K	Z, C, N, V	2
SUB	Rd, Rr	Subtract without Carry – Відняти без перенесення	Rd ← Rd – Rr	Z, C, N, V, H	1
SUBI	Rd, K	Subtract Immediate – Відняти безпосереднє значення	Rd ← Rd – K	Z, C, N, V, H	1
SBC	Rd, Rr	Subtract with Carry – Відняти з перенесенням	Rd ← Rd – Rr – C	Z, C, N, V, H	1
SBCI	Rd, K	Subtract Immediate with Carry – Відняти безпосереднє значення з перенесенням	Rd ← Rd – K – C	Z, C, N, V, H	1
SBIW	Rd, K	Subtract Immediate from Word – Відняти безпосереднє значення зі слова	Rdh : Rdl ← Rdh : : Rdl – K	Z, C, N, V	2
AND	Rd, Rr	Logical AND – Виконати логічне AND	Rd ← Rd · Rr	Z, N, V	1
ANDI	Rd, K	Logical AND with Immediate – Виконати логічне AND з безпосереднім значенням	Rd ← Rd · K	Z, N, V	1
OR	Rd, Rr	Logical OR – Виконати логічне OR	Rd ← Rd v Rr	Z, N, V	1
ORI	Rd, K	Logical OR with Immediate – Виконати логічне OR з безпосереднім значенням	Rd ← Rd v K	Z, N, V	1
EOR	Rd, Rr	Exclusive OR – Виконати OR, що виключає	Rd ← Rd ⊕ Rr	Z, N, V	1
COM	Rd	One's Complement – Виконати доповнення до одиниці	Rd ← □FF – Rd	Z, C, N, V	1
NEG	Rd	Two's Complement – Виконати доповнення до двох	Rd ← □00 – Rd	Z, C, N, V, H	1
SBR	Rd, K	Set Bits in Register – Встановити біти у регістрі	Rd ← Rd v K	Z, N, V	1

Продовження табл. Д 5.1

Мнемоніка	Операнди	Опис	Операція	Прапори	Кіл-ть циклів
CBR	Rd, K	Clear Bits in Register – Очистити біти у регістрі	• Rd ← Rd (□FF – K)	Z, N, V	1
INC	Rd	Increment – Інкрементувати	Rd ← Rd + 1	Z, N, V	1
DEC	Rd	Decrement – Декрементувати	Rd ← Rd – 1	Z, N, V	1
TST	Rd	Test for Zero or Minus – Перевірити на нуль або мінус	Rd ← Rd · Rd	Z, N, V	1
CLR	Rd	Clear Register – Очистити регістр	Rd ← Rd ⊕ Rd	Z, N, V	1
SER	Rd	Set all bits in Register – Встановити всі біти регістра	Rd ← □FF	Немає	1
CP	Rd, Rr	Compare – Порівняти	Rd – Rr	Z, C, N, V, H	1
CPC	Rd, Rr	Compare with Carry – Порівняти з урахуванням перенесення	Rd – Rr – C	Z, C, N, V, H	1
CPI	Rd,K	Compare with Immediate – Порівняти з константою	Rd – K	Z, C, N, V, H	1
Команди переходів					
RJMP	k	Relate Jump – Перейти відносно	PC ← PC + k + 1	Немає	2
IJMP		Indirect Jump – Перейти побічно	PC ← Z	Немає	2
JMP	k	Jump – Перейти	PC ← k	Немає	3
RCALL	k	Relative Call to Subroutine – Викликати підпрограму відносно	PC ← PC + k + 1	Немає	3
ICALL		Indirect Call to Subroutine – Викликати підпрограму побічно	PC ← Z	Немає	3
CALL	k	Long Call to a Subroutine – Виконати довгий виклик підпрограми	PC ← k	Немає	4
RET		Return from Subroutine – Повернутися з підпрограми	PC ← STACK	Немає	4
RETI		Return from Interrupt – Повернутися з переривання	PC ← STACK	I	4
CPSE	Rd,Rr	Compare Skip if Equal – Порівняти і пропустити, якщо рівно	If Rd = Rr then PC ← PC + 2 (or 3),	Немає	1 / 2 / 3
SBRC	Rr,b	Skip if Bit in Register is Cleared – Пропустити, якщо біт у регістрі очищений	If Rr (b) = 0 then PC ← PC + 2 (or 3)	Немає	1 / 2 / 3

Продовження табл. Д 5.1

Мнемоніка	Операнди	Опис	Операція	Прапори	Кіл-ть циклів
SBRS	Rr, b	Skip if Bit in Register is Set – Пропустити, якщо біт у регістрі встановлений	If Rr (b) = 1 then PC ← PC + 2 (or 3)	Немає	1 / 2 / 3
SBIC	P, b	Skip if Bit I/O Register is Cleared – Пропустити якщо біт у регістрі I/O очищений	If I/O (P, b) = 0 then PC ← PC + 2 (or 3)	Немає	1 / 2 / 3
SBIS	P, b	Skip if Bit I/O Register is Set – Пропустити, якщо біт у регістрі I/O встановлений	If I/O (P, b) = 1 then PC ← PC + 2 (or 3)	Немає	1 / 2 / 3
BRBS	s, k	Branch if Bit in SREG is Set – Перейти, якщо біт у регістрі статусу встановлений	If SREG (s) = 1 then PC ← PC + k +1	Немає	1 / 2
BRBC	s, k	Branch if Bit in SREG is Cleared – Перейти, якщо біт у регістрі статусу очищений	If SREG (s) = 0 then PC ← PC + k +1	Немає	1 / 2
BREQ	K	Branch if Equal – Перейти, якщо рівно	If Rd = Rr (Z = 1) then PC ← PC + k +1	Немає	1 / 2
BRNE	K	Branch if Not Equal – Перейти, якщо не рівно	If Rd = Rr (Z = 0) then PC ← PC + k +1	Немає	1 / 2
BRCS	k	Branch if Carry Set – Перейти, якщо прапор перенесення встановлений	If C = 1 then PC ← PC + k +1	Немає	1 / 2
BRCC	k	Branch if Carry Cleared – Перейти, якщо прапор перенесення очищений	If C = 0 then PC ← PC + k +1	Немає	1 / 2
BRSH	k	Branch if Same or Higher (Unsigned) – Перейти якщо рівно або більше (без знака)	If Rd → Rr (C = 0) then PC ← PC + k +1	Немає	1 / 2
BRLO	k	Branch if Lower (Unsigned) – Перейти, якщо менше (без знака)	If Rd ← Rr (C = 1) then PC ← PC + k +1	Немає	1 / 2
BRMI	k	Branch if Minus – Перейти якщо мінус	If N = 1 then PC ← PC + k +1	Немає	1 / 2

Продовження табл. Д 5.1

Мне-моніка	Опе-ранди	Опис	Операція	Прапо-ри	Кіл-ть циклів
BRPL	k	Branch if Plus – Перейти, якщо плюс	If N = 0 then PC ← PC + k +1	Немає	1 / 2
BRGE	k	Branch if Greater or Equal (Signed) – Перейти, якщо більше або рівно (з урахуванням знака)	If Rd ≥ Rr (N ⊕ V= 0) then PC ← PC + k +1	Немає	1 / 2
BRLT	k	Branch if Less Then (Signed) – Перейти, якщо менше ніж (зі знаком)	If Rd < Rr (N ⊕ V = 1) then PC ← PC + k +1	Немає	1 / 2
BRHS	k	Branch if Half Carry Flag is Set – Перейти, якщо прапор напівперенесення встановлений	If H = 1 then PC ← PC + k +1	Немає	1 / 2
BRHC	k	Branch if Half Carry Flag is Cleared – Перейти, якщо прапор напівперенесення очищений	If H = 0 then PC ← PC + k +1	Немає	1 / 2
BRTS	k	Branch if T Flag is Setи – Перейти, якщо прапор встановлений	If T = 1 then PC ← PC + k +1	Немає	1 / 2
BRTC	k	Branch if T Flag is Cleared – Перейти, якщо прапор очищений	If T = 0 then PC ← PC + k +1	Немає	1 / 2
BRVS	k	Branch if Overflow Set – Перейти, якщо переповнювання встановлене	If V = 1 then PC ← PC + k +1	Немає	1 / 2
BRVC	k	Branch if Overflow Cleared – Перейти, якщо переповнювання очищене	If V = 0 then PC ← PC + k +1	Немає	1 / 2
BRIE	k	Branch if Global Interrupt is Enabled – Перейти, якщо глобальне переривання дозволено	If I = 1 then PC ← PC + k +1	Немає	1 / 2
BRID	k	Branch if Global Interrupt is Disabled – Перейти, якщо глобальне переривання заборонено	If I = 0 then PC ← PC + k +1	Немає	1 / 2

Продовження табл. Д 5.1

Мнемоніка	Операнди	Опис	Операція	Прапори	Кіл-ть циклів
Команди пересилання даних					
ELPM		tended Load Program Memory – Розширене завантаження пам'яті програм	$R_0 \leftarrow (Z + RAMPZ)$	Немає	3
MOV	Rd, Rr	Copy Register – Копіювати регістр	Rd ← Rr	Немає	1
LDI	Rd, K	Load Immediate – Завантажити безпосереднє значення	Rd ← K	Немає	1
LDS	Rd, k	Load Direct from RAM – Завантажити безпосередньо з СОЗУ	Rd ← (k)	Немає	3
LD	RdX	LD – Load Indirect – Завантажити побічно	Rd ← (X)	Немає	2
LD	Rd, X +	Load Indirect and Post-Increment – Завантажити побічно інкрементувати згодом	Rd ← (X), X ← X + 1	Немає	2
LD	Rd, – X	Load Indirect and Pre-Decrement – Завантажити побічно, декрементувавши заздалегідь	X ← X – 1, Rd ← (X)	Немає	2
LD	Rd, Y	Load Indirect – Завантажити побічно	Rd ← (Y)	Немає	2
LD	Rd, Y +	Load Indirect and Post-Increment – Завантажити побічно, інкрементувавши згодом	Rd ← (Y), Y ← Y + 1	Немає	2
LD	Rd, – Y	Load ndirect and Pre-Decrement – Завантажити побічно, декрементувавши заздалегідь	Y ← Y – 1, Rd ← (Y)	Немає	2
LDD	Rd, Y + q	Load Indirect with Displacement – Завантажити непрямим зсувом	Rd ← (Y + q)	Немає	2
LD	RdZ	Load Indirect – Завантажити побічно	Rd ← (Z)	Немає	2
LD	Rd, Z +	Load Indirect and Post-Increment – Завантажити побічно, інкрементувавши згодом	Rd ← (Z), Z ← Z + 1	Немає	2
LD	Rd, –Z	Load Indirect and Pre-Decrement – Завантажити побічно, декрементувавши заздалегідь	Z ← Z – 1, Rd ← (Z)	Немає	2
LDD	Rd, Z + q	Load Indirect with Displacement – Завантажити непрямим змішенням	Rd ← (Z + q)	Немає	2

Продовження табл. Д 5.1

Мнемоніка	Операнди	Опис	Операція	Прапори	Кіл-ть циклів
STS	k, Rr	Store Direct to RAM – Загородити безпосередньо у СОЗУ	(k) ← Rr	Немає	3
ST	X, Rr	Store Indirect – Записати побічно	(X) ← Rr	Немає	2
ST	X +, Rr	Store Indirect and Post-Increment – Записати побічно, інкрементувавши згодом	X ← X + 1	Немає	2
ST	– X, Rr	Store Indirect and Pre-Decrement – Записати побічно, декрементувавши заздалегідь	X ← X – 1, (X) ← Rr	Немає	2
ST	Y, Rr	Store Indirect – Записати побічно	(Y) ← Rr	Немає	2
ST	Y +, Rr	Store Indirect and Post-Increment – Записати побічно, інкрементувавши згодом	(Y) ← Rr, Y ← Y + 1	Немає	2
ST	–Y, Rr	Store Indirect and Pre-Decrement – Записати побічно, декрементувавши заздалегідь	Y ← Y – 1, (Y) ← Rr	Немає	2
STD	Y + q, Rr	Store Indirect with Displacement – Записати побічно зі зсувом	(Y + q) ← Rr	Немає	2
ST	Z, Rr	Store Indirect – Записати побічно	(Z) ← Rr	Немає	2
ST	Z +, Rr	Store Indirect and Post-Increment – Записати побічно, інкрементувавши згодом	(Z) ← Rr, Z ← Z + 1	Немає	2
ST	– Z, Rr	Store Indirect and Pre-Decrement – Записати побічно, декрементувавши заздалегідь	Z ← Z – 1, (Z) ← Rr	Немає	2
STD	Z + q, Rr	Store Indirect with Displacement – Записати побічно зі зсувом	(Z + q) ← Rr	Немає	2
LPM		Load Program Memory – Завантажити байт пам’яті програм	R0 ← (Z)	Немає	3
IN	Rd, P	Load an I/O Port to Register – Завантажити дані з порту I/O у регістр	Rd ← P	Немає	1
OUT	P, Rr	Store Register to I/O port – Записати дані з регістра у порт I/O	P ← Rr	Немає	1
PUSH	Rr	Push Register on Stack – Помістити регістр у стек	STACK ← Rr	Немає	2
POP	Rd	Pop Register from Stack – Завантажити регістр зі стека	Rd ← STACK	Немає	2

Продовження табл. Д 5.1

Мне-моніка	Опе-ранди	Опис	Операція	Прапо-ри	Кіл-ть циклів
Бітові команди і команди тестування бітів					
LSL	Rd	Logical Shift Left – Логічно зсунути вліво	Rd (n + 1) ← Rd (n), Rd (0) ← 0, C ← Rd (7)	Z, C, N, V, H	1
LSR	Rd	Logical Shift Right – Логічно зсунути управо	Rd (n) ← Rd (n+1), Rd (7) ← 0, C ← Rd (0)	Z, C, N, V	1
ROL		Rotate Left trough Carry – Зрушити вліво через перенесення	Rd (0) ← C, Rd (n+1) ← Rd (n), C ← Rd (7)	Z, C, N, V, H	1
ROR	Rd	Rotate Right trough Carry – Зрушити вправо через перенесення	Rd (7) ← C, Rd (n) ← Rd (n + 1), C ← Rd (0)	Z, C, N, V	1
ASR	Rd	Arithmetic Shift Right – Арифметично зрушити вправо	Rd (n) ← Rd (n + 1), n = 0...6	Z, C, N, V	1
SWAP	Rd	Swap Nibbles – Поміняти нібли місцями	Rd (3...0) ← → Rd (7...4)	Немає	1
BSET	s	Flag Set – Встановити прапор	SREG (s) ← 1	SREG (s)	1
BCLR	s	Flag Clear – Очистити прапор	SREG (s) ← 0	SREG (s)	1
SBI	P, b	Set bit to I/O Register – Встановити біт у регістр I/O	I / O (P, b) ← 1	Немає	2
CBI	P, b	Clear Bit in I/O Register – Очистити біт у регістрі I/O	I / O (P, b) ← 0	Немає	2
BST	Rd, b	Bit Store from Register to T – Переписати біт з регістра у прапор T	T ← Rd (b)	T	1
BLD	Rd,b	Bit Load from T to Register – Загрузити T флаг у біт регістра	Rd (b) ← T	Немає	1
SEC		Set Carry Flag – Встановити прапор перенесення	C ← 1	C	1
CLC		Clear Carry Flag – Очистити прапор перенесення	C ← 0	C	1

Закінчення табл. Д 5.1

Мнемоніка	Операнди	Опис	Операція	Прапори	Кіл-ть циклів
SEN		Set Negative Flag – Встановити прапор негативного значення	N ← 1	N	1
CLN		Clear Negative Flag – Очистити прапор негативного значення	N ← 0	N	1
SEZ		Set Zero Flag – Встановити прапор нульового значення	Z ← 1	Z	1
CLZ		Clear Zero Flag – Очистити прапор нульового значення	Z ← 0	Z	1
SEI		Set Global Interrupt Flag – Встановити прапор глобального переривання	I ← 1	I	1
CLI		Clear Global Interrupt Flag – Очистити прапор глобального переривання	I ← 0	I	1
SES		Set Signed Flag – Встановити прапор знака	S ← 1	S	1
CLS		Clear Signed Flag – Очистити прапор знака	S ← 0	S	1
SEV		Set Overflow Flag – Встановити прапор переповнювання	V ← 1	V	1
CLV		Clear Overflow Flag – Очистити прапор переповнювання	V ← 0	V	1
SET		Set T Flag – Встановити прапор T	T ← 1	T	1
CLT		Clear T Flag – Очистити прапор T	T ← 0	T	1
SEH		Set Half Carry Flag – Встановити прапор напівперенесення	H ← 1	H	1
CLH		Clear Half Carry Flag – Очистити прапор напівперенесення	H ← 0	H	1
NOP		No Operation – Виконати порожню команду		Немає	1
SLEEP		Sleep – Встановити режим SLEEP	Дивись опис команди	Немає	1
WDR		Watchdog Reset – Скидання сторожового таймера	Дивись опис команди	Немає	1

ДОДАТОК 6

Приклади структурних схем, алгоритмів і програм застосування мікроконтролерів у схемах автоматизованого керування ЕА і ЕПТ

Приклад Д 6.1. Програма автоматизованого керування дослідженням швидкодіючого запобіжника на мові Асемблер МК51

```
        ORG     00
        JMP     START

        ORG     080H
START:  JB      P1.6,   START       ;очікування пуску
        CLR     P1.3                ;установка вихідного
        SETB    P1.4                ;стану
        SETB    P1.5
        SETB    P1.6

        MOV     A,      P2          ;включення ПЗО
        ORL     A,      #011110000B
        MOV     P2,     A

        MOV     20H,    #00         ;заповнення нулями лічильника

LAB2:   MOV     P0,     #0FFH       ;підготовка портів
        SETB    P2.0
        SETB    P2.1

        MOV     A,      P1          ;підключення датчика
        ANL     A,      #011111000B
        ORL     A,      20H
        MOV     P1,     A

        NOP                         ;затримка
        NOP

        SETB    P1.3                ;включення комутатора
        CLR     P1.4                ;пуск АЦП

LAB1:   JB      P1.5,               ;очікування готовності АЦП

        MOV     022H,               ;зчитування коду АЦП
```

```
        MOV     A,              ;0-7 біт – ячейка 022H
        ANL     A,              8,9 біт – ячейка 021H
        MOV     021H,

        SETB    P1.4            ;зняття пуску АЦП
        CLR     P1.5

        MOV     A,              ;перехід на підпрограми
        CGNE    A,              ;обчислення параметрів
        CALL    PROG0           ;відповідно до номера
        JMP     LAB6            ;датчика
LAB3:   CGNE    A,
        CALL    PROG1
        JMP     LAB6
LAB4:   CGNE    A,
        CALL    PROG2
        JMP     LAB6
LAB5:   CGNE    A,
        CALL    PROG3
LAB6:
        MOV     A,      020H    ;вибірка із таблиці  ст. байт
        ADD     A,      #030H   ;P і Pmax
        MOV     R0,     A
        MOV     A,      $R0

        MOV     R1,     A
        MOV     R2,     A
        MOV     A,      020H
        ADD     A,      #040H
        MOV     R0,     A
        MOV     A,      $R0

        CLR     C               ;порівняння ст. байт P і Pmax
        SUBB    A,      R1
        JC      LAB7

        MOV     A,      020H    ;вибірка із таблиці молодший байт
        ADD     A,      #031H   ;P і Pmax
        MOV     R0,     A
        MOV     A,      $R0
        MOV     R1,     A
        MOV     R3,     A
```

```
        MOV     A,      020H
        ADD     A,      $041H
        MOV     R0,     A
        MOV     A,      $R0

        CLR     C               ;порівняння молодший байт P і Pmax
        SUBB    A,      R1
        JC      LAB7

        CLR     TI              ;передача параметра по RS-232
        MOV     SBUF,   R2
LAB10:  JNB     TI,     LAB10
        CLR     TI

        MOV     SBUF,   R3
LAB11:  JNB     TI,     LAB11

        MOV     A,      020H    ;цикл
        INC     A,
        CGNE    A,      #04, LAB2

        MOV     A,      P2      ;якщо цикл закінчено, то
        ANL     A,      #00001111 ;вимкнення ПЗО
                        B
        MOV     P2,     A

        JMP     START           ;повернення на початок

LAB7:   MOV     A,      P2      ;вимкнення ПЗО,
        ANL     A,      #00001111 ;якщо P > Pmax
                        B
        MOV     P2,     A

        CLR     TI              ;передача повідомлення про
        MOV     SBUF,   #0FFH   ;аварію по RS-232
LAB8:   JNB     TI,     LAB8
        CLR     TI
        MOV     SBUF,   #0FFH
LAB9:   JNB     TI,     LAB9

        JMP     START           ;повернення на початок

        END
```

Приклад Д 6.2. Розробка структурної схеми мікроконтролерного стенда та алгоритму і програми автоматизації випробування і дослідження мікрохвильових печей.

Структурна схема. Для автоматичного дослідження НВЧ печей з мікроконтролерним керуванням запропонована структурна схема на базі мікроконтролера КМ1816ВЕ51. Восьмирозрядний високопродуктивний однокристальний мікроконтролер КМ1816ВЕ51 виконаний за високоякісною *n*-МОП технологією є програмно сумісним з іншими мікроконтролерами сімейства MCS-51 [20].

Структурна схема для автоматизованого дослідження мікрохвильових печей, яка зображена на рис. Д 6.1, включає такі елементи:

− мікрохвильову піч з елементами приготування їжі (магнетрон, гриль, конвектор);

− двійкові датчики типу так / ні Д1–Д4;

− датчики контрольованих параметрів (ваги, температури, пару, вологості) Д5–Д8 (первинні перетворювачі);

− нормуючі підсилювачі П1–П4;

− чотириканальний комутатор аналогових сигналів типу КМ 590КМ6;

− аналого-цифровий перетворювач (АЦП) типу К 1113 ПВ1;

− мікроконтролер, що містить вбудований генератор тактових сигналів, пам’ять команд, ОЗП, вбудовані 4 порти і послідовний канал зв’язку.

Через послідовний інтерфейс RS232 схема пов’язана з ПЕОМ, яка може змінювати режими випробувань або досліджень, а також приймати, запам’ятовувати, відображати і документувати результати досліджень.

До об’єкта дослідження підключені відповідні датчики. Датчики контрольованих параметрів Д5–Д8 є первинними перетворювачами ваги, температури, пару, вологості у напругу. Нормуючі підсилювачі погоджують вихідну напругу датчиків з необхідним вхідним сигналом АЦП 0–10 В і забезпечують низький вихідний опір.

Комутатор аналогових сигналів перемикає один із входів на вихід залежно від керуючого коду, що поступив від мікроконтролера. Аналоговий сигнал з виходу комутатора поступає на АЦП, який забезпечує перетворення його у цифровий код. Таке перетворення реалізовано найбільш швидкодіючим апаратним засобом на основі ВІС АЦП, що підключається до порту МК.

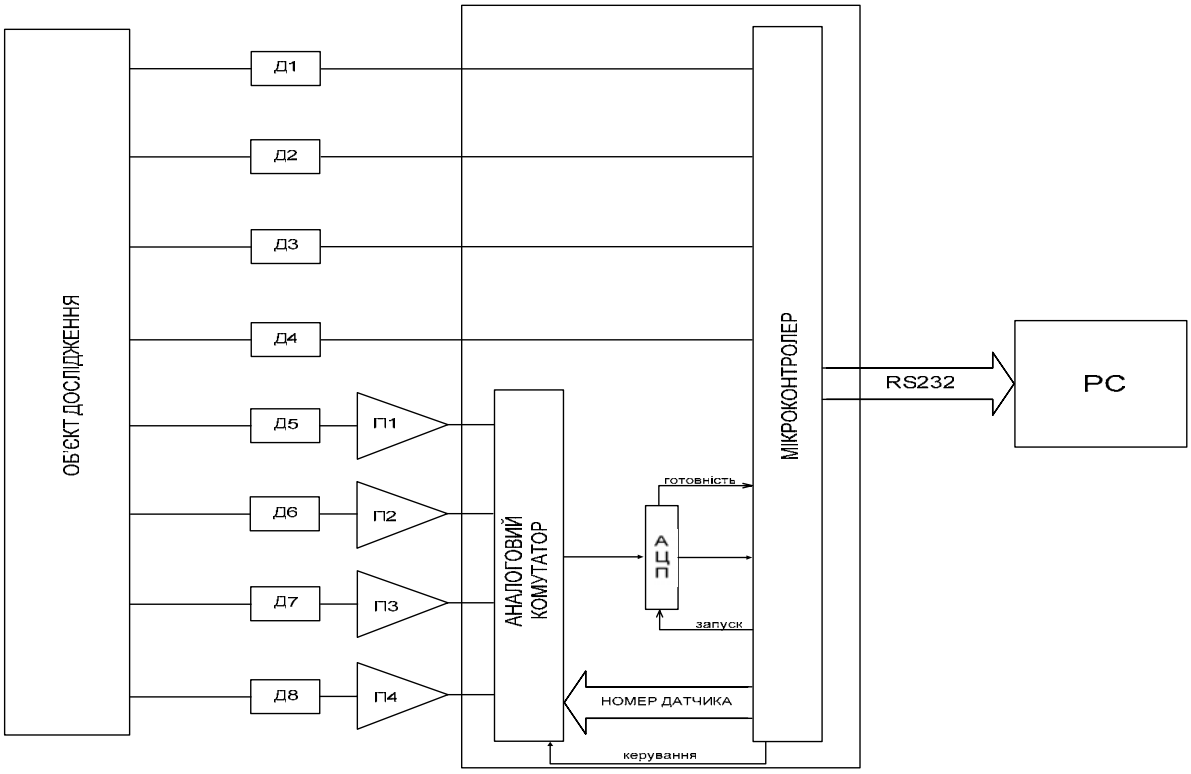


Рисунок Д 6.1 – Структурна схема автоматизованого дослідження мікрохвильових печей

Схему підключення портів мікроконтролера зображено на рис. Д 6.2.

Незадіяні порти можуть бути надалі використані для розширення функціональних можливостей мікроконтролера.

Як датчик температури можуть застосовуватися термопари. Наприклад, хромель-алюмельові термопари, які відрізняються невисокою вартістю, призначені для вимірювання температури в діапазоні від –270 °С до +1372 °С. Чутливість цих термопар складає 41 мкВ / °С. Для вимірювання ваги продукту застосовуються тензометричні або ємносні датчики, наприклад, тензо датчик балочного типу СВ1, фірми «Прибор». Діапазон вимірювання складає від 150 Г до 50 кг, номінальний вихідний сигнал 1 мВ / Г ± 1 %, нелінійність 0,02 %, гістерезис 0,02 %, вхідний опір 420 ± 30 Ом.

Для вимірювання вологості застосовуються датчики абсолютної або відносної вологості. Наприклад, датчик вологості HIH-3602-L фірми Honeywell, виконаний у корпусі TO-39 з щілинним отвором. Діапазон вимірювання датчика складає 0...l00 % RH, лінійність 0,50 ± % RH, гістерезис 1,20 ± % RH, $U_{живл}$ = 4,0 В, $I_{живл}$ = 0,20 мА.

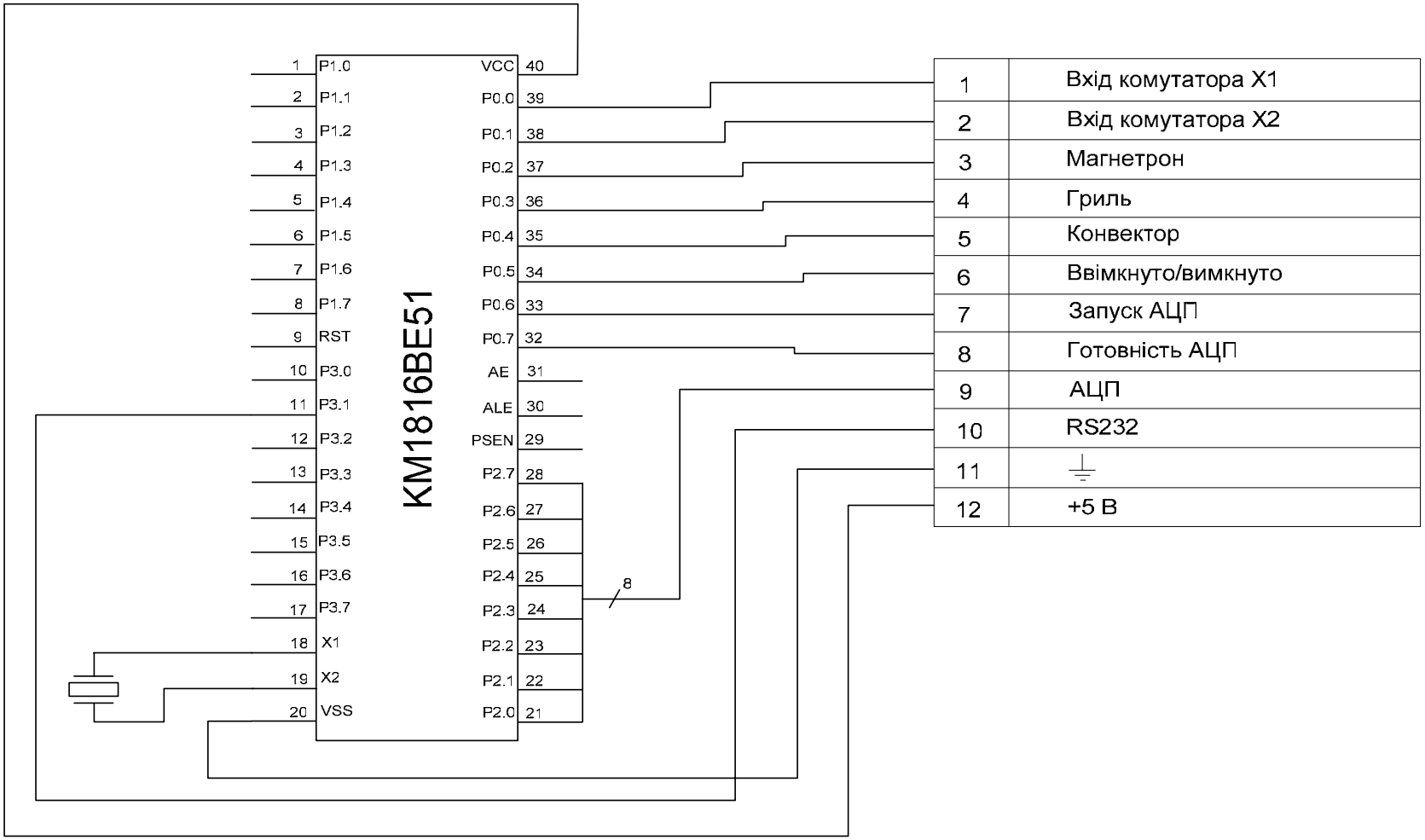


Рисунок Д 6.2 – Схема підключення портів МК КМ1816ВЕ51

Датчиком пари у мікрохвильових печах застосовується піроелектричний датчик. Вихідні сигнали датчиків унаслідок їх різної фізичної природи можуть потребувати посилення і проміжного перетворення на АЦП або на схемах формувачів сигналів, які найчастіше виконують функції гальванічної розв'язки і формування рівнів двійкових сигналів стандарту ТТЛ.

Алгоритм роботи програми. Для проведення дослідження з необхідною точністю на початку програми константі TIME привласнюється значення інтервалу часу, через який опитуються датчики і елементи мікрохвильової печі для приготування їжі. Введення константи на початку програми необхідно для того, щоб при зміні часу опитування у програміста на виникали складності у зміні тексту програми.

Далі заповнюємо нулями чотири порти мікроконтролера і перевіряємо чи включена піч. Подальше виконання програми відбуватиметься тільки тоді, коли на відповідний розряд порту мікроконтролера буде подано сигнал логічної «1». При включенні печі мікроконтролер посилає сигнал керування на аналоговий комутатор. Згідно з одержаним сигналом комутатор сполучає виведення датчика ваги (Д5) з виведеннями відповідного порту мікроконтролера.

Далі на АЦП подається сигнал запуску, після зчитування і перетворення сигналу датчика, АЦП посилає сигнал готовності на мікроконтролер. Дані, прийняті з датчика, видаються зовнішньому пристрою через універ-сальний асинхронний приймач-передавач (УАПП) персональному комп'ютеру для подальшого зберігання та оброблення. Після передачі даних мікроконтролер проводить опитування приладів мікрохвильової печі, які використовуються у цей час для приготування їжі, а саме магнетрона, гриля і конвектора.

Дані про їх роботу через УАПП передаються у комп'ютер (PC).

Після цього програма аналогічно тому, як знімались дані з датчика ваги, зчитує дані з датчиків температури, пари і вологості. Далі мікроконтролер перевіряє ввімкнена чи вимкнена мікрохвильова піч. Якщо мікрохвильова піч ввімкнена, то програмно запускається таймер на час, який, указаний у константі TIME. Після закінчення відліку заданого часу мікроконтролер знову проводить перевірку роботи магнетрона, гриля і конвектора, та знімає дані з датчиків (окрім датчика ваги). Датчик ваги повторно не перевіряється, тому що у процесі приготування їжі вага продукту практично не змінюється, тобто істотних змін у програму приготування конкретної страви не вносять. Вихід з циклу і завершення роботи програми відбувається при відключенні печі.

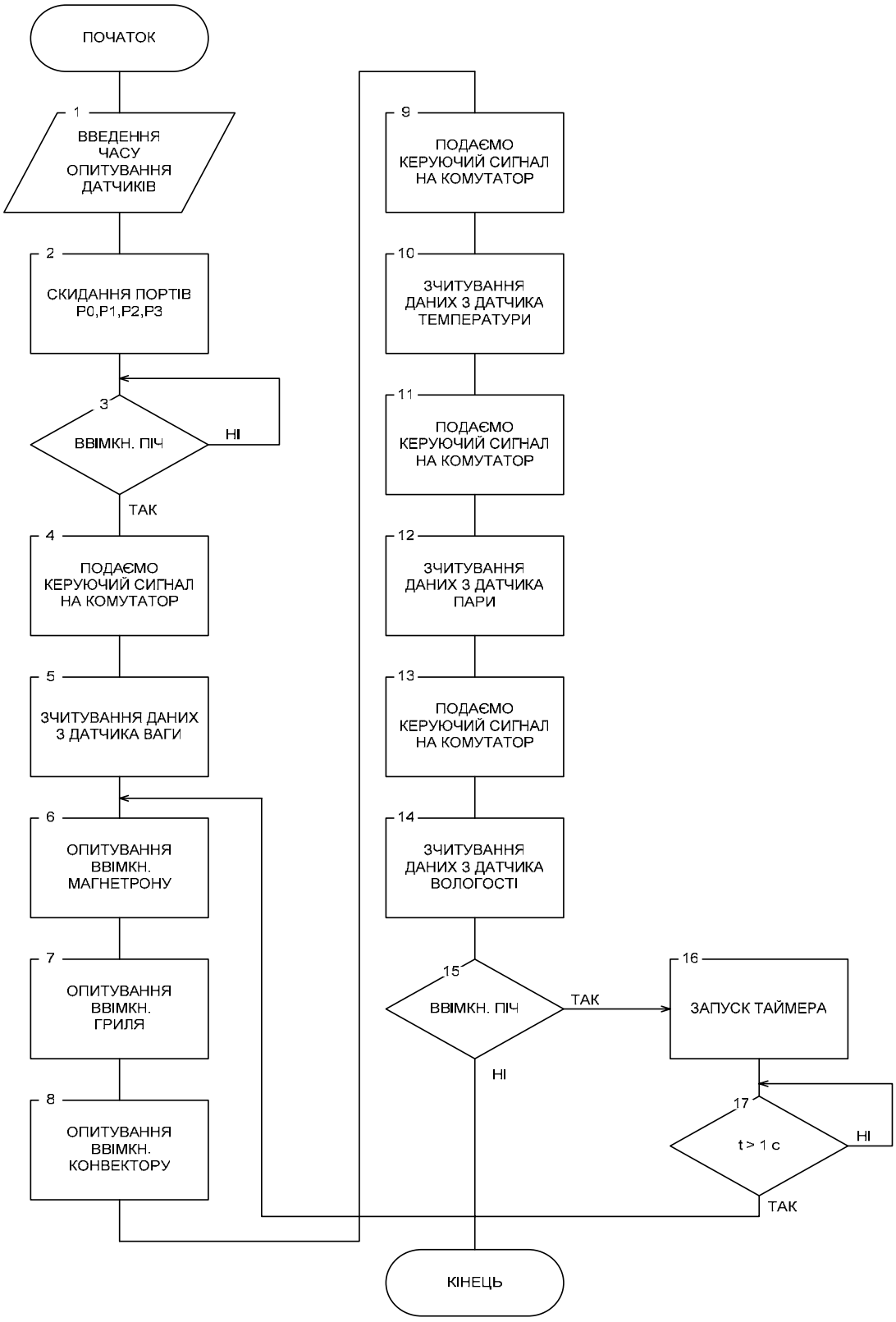


Рисунок Д 6.3 – Алгоритм роботи програми

Текст програми для дослідження роботи мікрохвильової печі має такий вигляд:

```
TIME EQU #FCh              ; визначення константи TIME
                           ; для відліку заданого часу
ANL P0,#00000000b          ; скидаємо порт P0
ANL P1,#00000000b          ; скидаємо порт P1
ANL P2,#00000000b          ; скидаємо порт P2
ANL P3,#00000000b          ; скидаємо порт P3
WAITC: JNB P3.0,WAITC      ; очікування включення печі
CLR P0.0                   ; подаємо керуючий сигнал
CLR P0.1                   ; на комутатор
ACALL ACP                  ; виклик підпрограми АЦП
ACALL RS_DAT4IKI           ; виклик підпрограми RS232
EXIT_TAIMER: MOV P0,#00000000h
ACALL RS_PRIBORI           ; виклик підпрограми RS232
SETB P0.0                  ; подаємо керуючий сигнал
CLR P0.1                   ; на комутатор
ACALL ACP                  ; виклик підпрограми АЦП
ACALL RS_DAT4IKI           ; виклик підпрограми RS232
CPL P0.0                   ; подаємо керуючий сигнал
SETB P0.1                  ; на комутатор
ACALL ACP                  ; виклик підпрограми АЦП
ACALL RS_DAT4IKI           ; виклик підпрограми RS232
SETB P0.0                  ; подаємо керуючий сигнал
SETB P0.1                  ; на комутатор
ACALL ACP                  ; виклик підпрограми АЦП
ACALL RS_DAT4IKI           ; виклик підпрограми RS232
JNB P3.0,EXIT              ; перевірка включення печі
MOV TMOD,#01010001b        ; налаштування таймера
MOV TH0,TIME               ; завантаження часу в таймер
MOV TL0,#03h
ORL TCON,#50h              ; запуск таймера
WAIT: JBC TCON.5, M1       ; перевірка переповнювання таймера
SJMP WAIT                  ; цикл, якщо TF=0
```

```
EXIT: END                          ; кінець програми
ACP: ORG 4Eh                       ; підпрограма АЦП
SETB P0.6                          ;подаємо сигнал запуску на АЦП
WAIT_ACP: JNB P0.7,WAIT_ACP    ;сигнал готовності АЦП
CLR P0.6                           ; скидаємо сигнал запуску АЦП
CLR P0.7                           ; скидаємо сигнал готовності АЦП
RET                                ; вихід з підпрограми
RS_DAT4IKI: CLR TI                 ; підпрограма RS232 для датчиків
MOV SBUF,P2                        ;завантажуємо у SBUF вміст P2
WAIT_RS: JB TI,M7                  ; очікування видачі даних
SJMP WAIT_RS
M7: RET                            ; вихід з підпрограми
RS_PRIBORI: CLR TI                 ; підпрограма RS232 для приладів
MOV SBUF,P0                        ; завантажуємо у SBUF вміст P0
WAIT_RS_2: JB TI,M8                ;очікування видачі даних
SJMP WAIT_RS_2
M8:  RET                           ; вихід з підпрограми
```

Висновки. Розроблений стенд, алгоритм його роботи і програма автоматизованого дослідження мікрохвильових печей дозволяють суттєво скоротити терміни і витрати на проведення їх випробувань і досліджень та підвищити точність отримуваних результатів.

Приклад Д 6.3. Розробка розчіплювача для автоматичних вимикачів на базі мікроконтролера MSP430F

Вступ. Захист низьковольтних електричних мереж постійного і змінного струму від ненормальних режимів роботи сьогодні виконується в основному автоматичними вимикачами. Швидкий ріст потужності сучасних мереж, їх високий рівень автоматизації викликають підвищення вимог до якості захисту, побудованого на автоматичних вимикачах.

Функцію захисту в автоматичних вимикачах (АВ) виконують звичайно різноманітні типи розчіплювачів [43], які при виникненні ненормальних режимів роботи впливають на механізм АВ, викликаючи розмикання контактів і відключення кола, яке захищається. Час спрацьовування

багатьох захисних автоматичних вимикачів, що випускаються в цей час, визначається електромагнітними розчеплювачами у режимі короткого замикання та тепловими розчіплювачами у режимі перевантаження. Останні являють собою біметалічну пластину, яка складається із двох металів з різними коефіцієнтами теплового розширення. Пластини жорстко з'єднані між собою гарячою прокаткою або зварюванням. При токових перенавантаженнях нагрів біметалічного елемента приводить до його вигину убік пластини з меншим температурним коефіцієнтом розширення. Пластина впливає на рейку механізму вільного розчеплювання. При цьому контакти розмикаються під дією пружини, що відключає. Тепловий розчеплювач характеризується тепловою інерцією. Швидкодія біметалічної пластини прямо пропорційна значенню струму. Після замикання автоматичного вимикача, який спрацював, час наступного спрацьовування розчіплювача зменшується.

Недоліком таких розчіплювачів є нестабільність часових характеристик, а також сильна залежність часу дії розчеплювачів від їхньої початкової температури й від температури навколишнього середовища.

Однією з найважливіших вимог до АВ у мережах до 1000 В є забезпечення необхідної швидкодії при аварійних відключеннях. Цей час залежить від величини напруги і при фазній напрузі 220 В не повинен перевищувати 0,4 с. Норми перевірки АВ, які існують сьогодні, потребують перевірки і забезпечення певної кратності струму короткого замикання за відношенням до номінальних струмів розчіплювачів автоматичних вимикачів. Така перевірка встановлює міру надійності відключення пошкоджень, але не гарантує швидкого їх відключення. Між тим проведені дослідження виявили залежність ступеня дії електричного струму не тільки від величини напруги, а й від тривалості його дії.

Аналізуючи часо-струмові характеристики вітчизняних АВ для мереж до 1000 В, можна зробити висновок, що фактор часу відключення ушкоджень не завжди вважався пріоритетним. Крім того, оскільки для кожного виду захисту використовувався певний розчіплювач – це призводило до збільшення габаритів АВ і як наслідок – його подорожчання та зменшення його надійності.

Метою роботи є розробка мікроконтролерного розчіплювача, який зможе замінити електромагнітний і тепловий розчіплювачі без зміни габаритних розмірів АВ. Упровадження такого розчіплювача знизить матеріалоємність, трудомісткість, собівартість, енергоспоживання і витрати при експлуатації, а також підвищить надійність АВ та їх захисну швидкодію.

Структурна схема мікроконтролерного розчіплювача

При необхідності контролю параметрів багатьох різних за природою фізичних процесів (горіння дуги, нагрів, електродинамічні зусилля, та ін.), що перебігають при комутації в електричних апаратах захисту, зокрема в автоматичних вимикачах, виникає необхідність у визначенні їх характеристик та запису даних. Ці процеси при відключенні аварійних струмів вельми короткочасні і мають тривалість від 1 до 10 мкс. Для вирішення поставленого завдання й усунення викладених вище недоліків пропонується у структурних схемах автоматичних вимикачів та у їх розчіплювачах використати високопродуктивний, малогабаритний, високонадійний, з низьким енергоспоживанням МК MSP430F [36]. Структурна схема розчіплювача на базі МК MSP430F показана на рис. Д 6.4. Вона містить у собі

– МК MSP430F, показаний на рис. Д 6.5;
– три датчики струму у вигляді трансформаторів струму ТА1–ТА3;
– джерела живлення у вигляді трансформатора TV;
– компаратори К1–К3;
– має зручний інтерфейс.

Усі електронні компоненти розчіплювача розташовані на одній печатній платі. Це дозволить швидко налагодити серійне виробництво, оскільки, не потрібно змінювати габаритні розміри вимикача. Крім того, вимикач з мікроконтролерним розчіплювачем має ряд нових захисних функцій. А саме:

– захист від струму витоку;
– два набори параметрів захисту і можливість перемикатися між ними вручну або за допомогою автоматики;
– подвійний захист від замикання на землю;
– подвійний захист від КЗ із селективністю за часом.

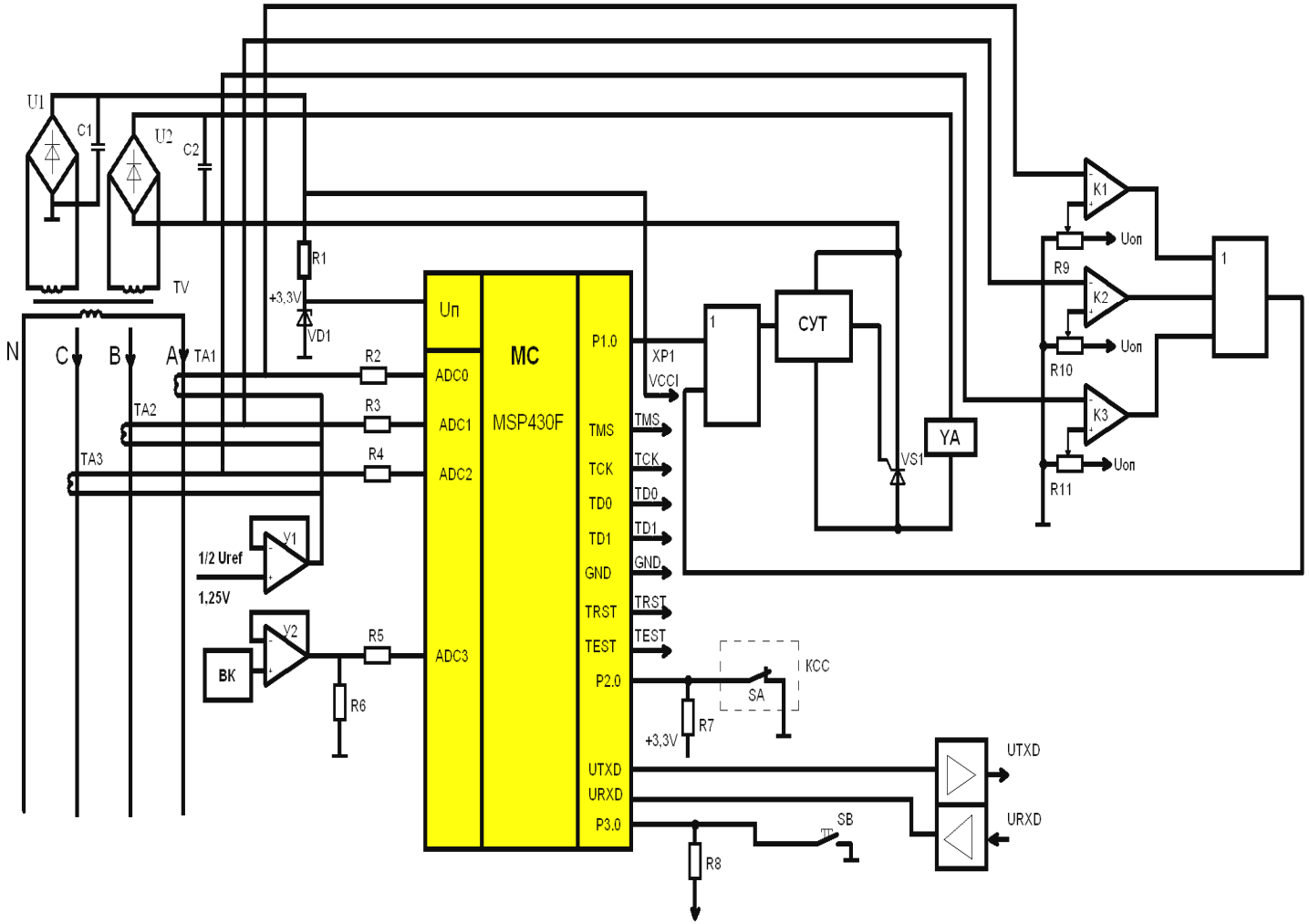


Рисунок Д 6.4 – Структурна схема мікроконтролерного розчеплювача на базі МК MSP430F

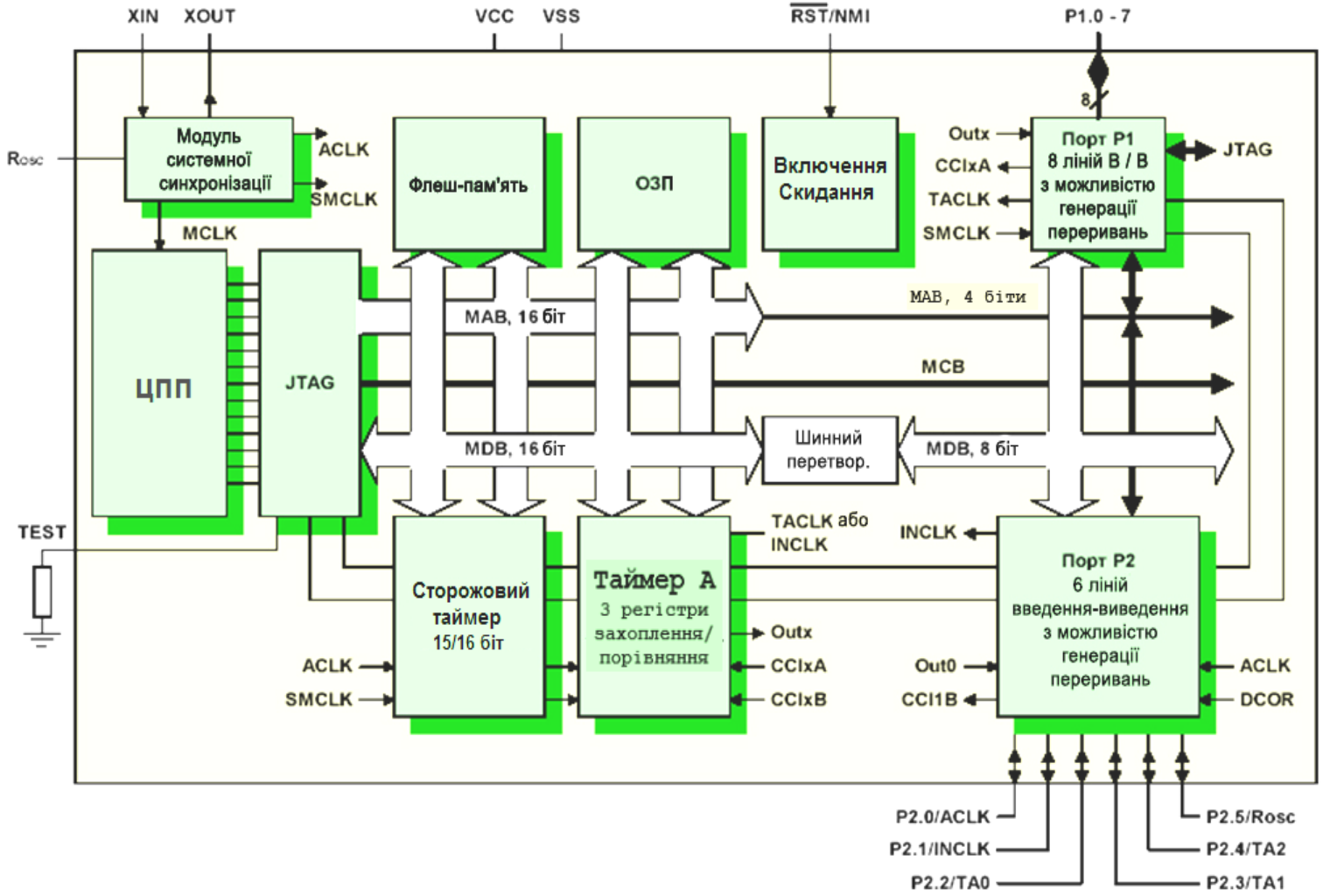


Рисунок Д 6.5 – Архітектура MSP430F

Первинна обмотка трансформатора живлення TV включена між однією з фаз і нулем. Одна вторинна обмотка підключена до випрямного мосту U1 на виході, до якого підключена ємність C1, а друга вторинна обмотка служить джерелом живлення для котушки електромагніта відключення YA. Напруга з ємності C1 подається на вхід живлення МК MSP430F $U_п$ і на мікросхему підсилювача потужності DA, вхід якого приєднаний до одного з розрядів порту, наприклад P1.0. Вихід підсилювача потужності DA приєднаний до схеми керування тиристором СУТ, що шляхом подачі напруги на керуючий електрод тиристора VS1 відкриває його й забезпечує протікання струму через котушку електромагніта відключення YA незалежного розчеплювача AB і Продовження додатка 6 замикає його після відключення. Коли спрацьовує захист, автоматичний вимикач розмикається за допомогою електромагніта відключення, при цьому змінюється стан контакту сигналізації спрацьовування КСС розчіплювача AB. Скидання сигналізації механічне і здійснюється переведенням, важеля керування у нижнє положення. Котушка розчіплювача не вимагає зовнішнього живлення, тому що вона живиться від трансформатора TV через випрямний міст U2 і ємність C2. МК розчіплювач може використовувати додаткове живлення від портативного блоку батарей, що дозволяє встановлювати параметри захисних функцій при відсутності живлення AB.

У разі виходу зі строю мікроконтролера з датчиків струму сигнал буде подаватися на компаратори $К_1$–$К_3$ і порівнюватись із заданим. Якщо він більший допустимого, то через елемент «АБО» він подається на схему керування тиристором. Потім шляхом подачі напруги на керуючий електрод тиристора VS1 відкриває його і забезпечує протікання струму через котушку електромагніта відключення YA незалежного розчіплювача AB і автоматичний вимикач відключається. Потім подається сигнал, що мікроконтролер вийшов зі строю.

Опис базового датчика струму

Важливим компонентом будь-якого електронного розчіплювача автоматичного вимикача є датчик струму. У низьковольтних вимикачах, обладнаних мікроконтролерними розчіплювачами, ці датчики застосовуються не тільки для вимірювання струму, але і забезпечують живлення електронних схем. Тому в розроблювальному вимикачі датчик повинен

виконувати ті ж функції. Оскільки в цей час найкращим способом передачі енергії розчіплювачу є трансформатор струму, зупиняємо свій вибір на гібридному датчику з контуром Роговського як базовому елементі. У доповнення до нього встановлений трансформатор струму, розрахований на окреме живлення мікро-контролера. Одна з переваг такого рішення – лінійна характеристика контуру Роговського у всьому діапазоні струмів, в якому працює вимикач. Іншою перевагою є те, що для нового трансформатора струму знижується кількість енергії, яка підводиться до розчіплювача при великих струмах.

Контур Роговського видає сигнал, пропорційний першій похідній струму, і цей сигнал необхідно інтегрувати. Інтегрування виконується у цифровому вигляді із застосуванням потужного мікроконтролера, який описаний вище і що є, фактично, «серцем» розчіплювача. Мікроконтролер також застосовується для виконання інших функцій, наприклад, зв'язку, що у попередніх розчіплювачах вимагало застосування спеціальних виконань. Відмова від цих виконань і спрощення вхідного каскаду розчіплювача дозволили розмістити всю електроніку пристрою на одній друкованій платі. Це значний прогрес у порівнянні з попередніми конструкціями, де компоненти, що виконують ті ж функції, розміщалися на чотирьох платах.

Такі серйозні зміни привели до важливих результатів:

– досягнуто скорочення кількості типорозмірів датчиків струму з 24 до 2 для всього ряду автоматичних вимикачів;

– виконано вимоги споживачів з підвищення точності вимірювання струму і реалізації додаткових функцій захисту.

Приклад Д 6.4. Алгоритм роботи мікроконтролерного розчіплювача

Алгоритм роботи мікроконтролерного розчіплювача зображений на рис. Д 6.6. На початку алгоритму показано приведення до вихідного стану зовнішніх схем (1). У головному циклі алгоритму (2–12) виконується включення АЦП (2), проводиться вибирання результатів вимірю вання (3). Далі порівнюється величина струму фази А і величина струму фази В (4). Якщо величина струму фази А більша ніж величина струму фази В, то відбувається порівняння величини струму фази А і величини струму фази С (5). Якщо величина струму фази А більша ніж величина

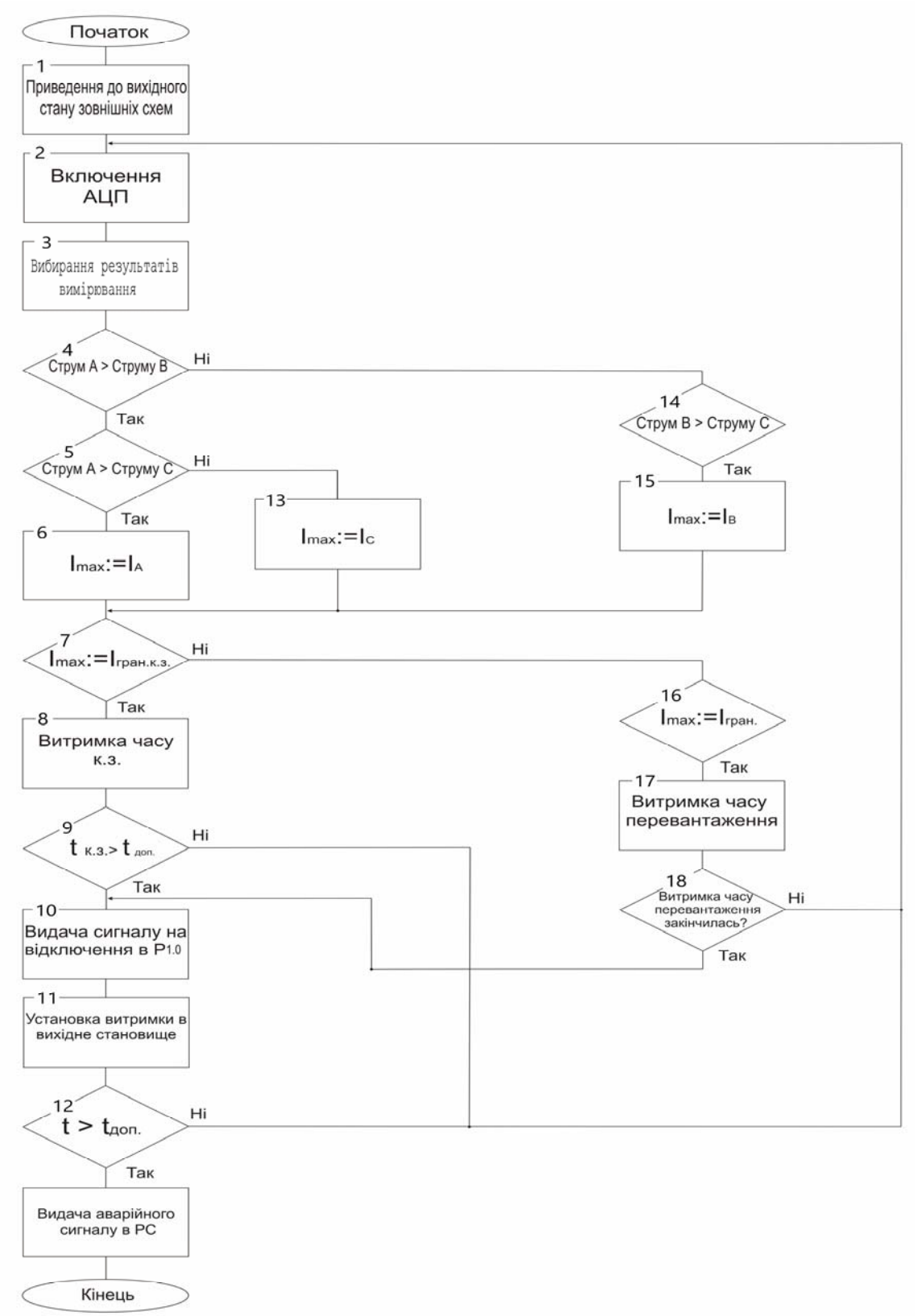


Рисунок Д 6.6 – Алгоритм роботи мікроконтролерного розчіплювача

струму фази С, то величина струму фази А визначається як максимальна (6). Величина максимального струму порівнюється з величиною граничного струму короткого замикання (7).

Якщо витримка часу $t_{к.з.}$ відповідає $t_{доп.}$ (8), то порівнюється час короткого замикання з часом витримки заданим програмно (9).

Якщо витримка часу короткого замикання закінчилась видається сигнал відключення у порт Р 1.0 (10).

Витримка встановлюється у вихідне положення (11) і знову час порівнюється із заданим (12). Якщо час більший ніж час заданий програмно, то відбувається відключення вимикача. У випадку коли величина струму фази А менша величини струму фази В (4), порівнюються величини струмів фаз В і С (14). Якщо величина струму фази В більша ніж величина струму фази С, то величина струму фази В визначається як максимальна (15). Далі повторюються дії (7–12). Якщо величина струму фази А менша ніж величина струму фази С, то максимальною визначається величина струму фази С (13) і далі повторюються дії (7–12). Якщо величина максимального струму не дорівнює величині граничного струму короткого замикання (7), то вона дорівнює величині граничного струму перевантаження (16) і відбувається витримка часу перевантаження (17) та порівняння її із заданою (18). Коли витримка часу перевантаження закінчується, то повторюються дії (10–12), а якщо витримка часу перевантаження не закінчилась, то повторюються дії (2–12).

Приклад Д 6.5. Програма керування мікроконтролерним розчіплювачем автоматичних вимикачів на базі МК MSP430 [44]

Опис програми

1. Загальна інформація

Для роботи програми не потрібно додаткового програмного забезпечення.

Програма написана мовою асемблер мікроконтролера MSP430.

2. Функціональне призначення

Програма призначена для функціонування в мікроконтролері MSP430F147, на основі якого виконаний мікроконтролерний розчіплювач для автоматичних вимикачів виробництва ВО «Промфактор». Програма виконує вимірювання струмів в електричному контурі, в якому автома-

тичний вимикач здійснює функції захисту від надструмів, струмообмеження при струмах короткого замикання і теплового захисту при струмах перевантаження.

Крім того, програма опитує кнопки керування і програмування мікроконтролерного розчіплювача, за допомогою яких набираються і задаються величини уставок для струму і поточного часу, а також їх запис в енергонезалежний запам'ятовуючий пристрій мікроконтролера. Також, за допомогою програми проводиться індикація на світлодіодах режимів роботи і значень запрограмованих уставок мікроконтролерного розчіплювача.

3. Логічна структура програми

Програма складається з наступних блоків:

– ініціалізації мікроконтролера, що здійснює налагодження його периферійних пристроїв і зчитування даних з енергонезалежного запам'ятовуючого пристрою;

– розрахунку уставок за струмом і часом спрацьовування захисту мікроконтролерного розчіплювача;

– вимірювання струмів в контрольованому електричному контурі;

– реалізації функцій захисту: порівняння виміряних струмів з уставками захисту, формування витримки часу та сигналу керування електронним розчіплювачем;

– опитування кнопок керування, індикації режиму роботи і запису уставок в енергонезалежний запам'ятовуючий пристрій.

Текст програми

```
#include  "msp430x14x.h" // Standard Equations
 #include "Define.h"
// #include       "Macros.h"
//-----------------------
     Rseg CODE
//-----------------------
Reset   Mov     #SpTag,Sp           ;init stack
//------ Налалагодження портів введеня-виведення
//------P2 налагоджений на приймання P2.0-P2.7=0
     BIC.B   #(BIT0+BIT1+BIT2+BIT3+BIT4+BIT5+BIT6+BIT7),&P2DIR
//------P1 налагоджений на приймання P1.0-P1.7=0
     BIC.B   #(BIT0+BIT1+BIT2+BIT3+BIT4+BIT5+BIT6+BIT7),&P1DIR
//------P5 налагоджений на виведення P5.0-P5.4=1
     BIS.B   #(BIT0+BIT1+BIT2+BIT3+BIT4),&P5DIR
```

```
//------P5 налагоджений на введення P5.5-P5.7=0
     BIC.B   #(BIT5+BIT6+BIT7),&P5DIR
//------P5 виведення P5.0-P5.4=1 світлодіоди викл
     BIS.B   #(BIT0+BIT1+BIT2+BIT3+BIT4),&P5OUT
//------P4.0 налагоджений на виведення P4.0=1
     BIS.B   #BIT0,&P4OUT
//------P4 налагоджений на введення P4.1-P4.7=0
     BIC.B   #(BIT1+BIT2+BIT3+BIT4+BIT5+BIT6+BIT7),&P4DIR
//------P3 налагоджений на введення P3.0-P3.7=0
     BIC.B   #(BIT0+BIT1+BIT2+BIT3+BIT4+BIT5+BIT6+BIT7),&P3DIR
//------P4.0 настроен на вывод P4.0=1
     BIS.B   #BIT0,&P4DIR
//-Налагодження базовых частот XT2=8МГц
        BiC    #OscOff,Sr
        BiC    #(ScG1+ScG0),Sr
        MOV.B  #0,&BCSCTL1
L1:  BiC.B   #OFIFG,&IFG1
     MOV     #0x00ff,R15
L2   Mov  #5a00h+WdTCntCl,WdTCtl        //Очищення WdT
        DEC     R15
     JNZ     L2
        Mov    #5a00h+WdTCntCl,WdTCtl        //Очищення WdT
     BIT.B   #OfIFg,&IFG1
     JNZ     L1
//-Налагодження базових частот XT2=8МГц, MCLK=8МГц
//-SMCLK=XT2=8MHZ FOR WDT и АЦП12 період WDT=4мсек
//- період WDT=4мсек
        nop
        Mov.B   #SELM_2+SELS,&BcsCtl2
     BIT.B   #OfIFg,&IFG1
     JNZ     L1
        Mov    #5a00h+WdTCntCl+WDTHOLD,WdTCtl //Очищення WDT
//-----Запис налагоджень із флеш-пам'яті в ОЗУ
Stand1: mov   #M0,R7  //Початкова адреса налагоджень у flash
WRT0:   mov   @R7+,RM0-M0-2(R7)  //Запис із флеш-пам'яті в ОЗУ
        CMP   #M15+2,R7  //Запис всіх налагоджень закінчений?
        JNZ   WRT0
        CLR   R11
        CLR   R12
        CLR   Shold
```

```
     CLR  Ihold
     CLR  Imax
     CLR  Imaxper
     CLR  Imaxper2
     CLR  Imaxpernew
     CLR  IAmeas
     CLR  IBmeas
     CLR  ICmeas
        CLR    IAtek
     CLR  IBtek
     CLR  ICtek
     CLR  Status
     CLR  I2TrHigh
     CLR  I2TrLow
        CLR     TimI2TrHigh
        CLR     TimI2TrLow
//--Налагодження порта P6, як  входів АЦП
     BiS.B     #0x1f,P6Sel
//---підготовка і розрахунок установок захистів--------------
------------
SetKg:  MOV     RM2,R7
        CMP     #8,R7
        JLO     WrKg
        MOV     #7,R7
WrKg:   DEC     R7
        RLA     R7
        ADD     #Kg1,R7
        Mov     @R7,Kg
//--------------------------------
SetKt:  MOV     RM3,R7
        CMP     #10,R7
        JLO     WrKt
        MOV     #9,R7
WrKt:   ADD     R7,Kg
//--------------------------------
SetTr:  MOV     RM4,R7
        CMP     #6,R7
        JLO     WrTr
        MOV     #5,R7
WrTr:   DEC     R7
        RLA     R7
        ADD     #Tr1,R7
        Mov     @R7,Tr
```

```
//------Trres=Tr*5sec---------------
        Mov     Tr,&MPYS
        Mov     #250,&OP2
        nop
        Mov    &RESLO,Trres
//--------------------------------
SetKsd: MOV    RM5,R7
        CMP     #9,R7
        JLO     WrKsd
        MOV     #8,R7
WrKsd:  DEC     R7
        RLA     R7
        ADD     #Ksd1,R7
        Mov     @R7,Ksd
//--------------------------------
SetTsd: MOV     RM6,R7
        CMP     #5,R7
        JLO     WrTsd
        MOV     #4,R7
WrTsd:  DEC     R7
        RLA     R7
        ADD     #Tsd1,R7
        Mov     @R7,Tsd
//--------------------------------
SetKi: MOV     RM7,R7
        CMP    #10,R7
        JLO    WrKi
        MOV    #9,R7
WrKi:   DEC    R7
        RLA    R7
        ADD    #Ki1,R7
        Mov    @R7,Ki
//--------------------------------
SetTi: MOV    RM8,R7
       CMP    #4,R7
       JLO    WrTi
       MOV    #3,R7
WrTi:  DEC     R7
       RLA    R7
       ADD    #Ti1,R7
       Mov    @R7,Ti
```

```
//--------Ir=In*Kg*328/256/128-------------------------------
-------------------------
        MOV    #328,&MPYS
     MOV    Kg,&OP2
        nop
        Mov    &RESLO,R7
        MOV    R7,&MPY
     MOV    Inorm,&OP2
        nop
        Mov    &RESLO,R7
        MOV    &RESHI,R15
        RLC    R7
        RLC    R15
        MOV    R15,Ir
//--------Ir2=In*Kg*301/256/128------------------------------
-------------------------
        MOV    #320,&MPYS
     MOV    Kg,&OP2
        nop
        Mov    &RESLO,R7
        MOV    R7,&MPY
     MOV    Inorm,&OP2
        nop
        Mov    &RESLO,R7
        MOV    &RESHI,R15
        RLC    R7
        RLC    R15
        MOV    R15,Ir2
//--------Irst=Ir*1,25=Ir*5/4--------------------------------
-------------------------
        MOV   #5,&MPYS
     MOV   R15,&OP2
        nop
        Mov   &RESLO,R15
        RRA   R15
        RRA   R15
        MOV   R15,Irst
//--------Ir85%=Ir*0,85=Ir*218/256---------------------------
-------------------------------
        MOV   #218,&MPYS
```

```
     MOV   Ir,&OP2
        nop
        Mov   &RESLO,R15
        Mov   &RESHI,Ir85
        SWPB  Ir85
        AND   #0xff00,Ir85
        SWPB  R15
        AND   #0x00ff,R15
        BIS    R15,Ir85
//--------Ir105%=Ir*1,05=Ir*267/256-------------------------
-------------------------------
        MOV   #269,&MPYS
     MOV   Ir,&OP2
        nop
        Mov   &RESLO,R15
        Mov   &RESHI,Ir105
        SWPB  Ir105
        AND   #0xff00,Ir105
        SWPB  R15
        AND   #0x00ff,R15
        BIS    R15,Ir105
//-----const=Ir*Ir*Tr*50/256-------------------------------
---------------
        Mov   Tr,R7
        RRA   R7
        MOV   R7,&MPYS
        MOV   #9,&OP2
        nop
        Mov   &RESLO,R7
        Mov   Ir,R15
        RRA   R15
        RRA   R15
        MOV   R7,&MPYS
        MOV   R15,&OP2
        nop
        Mov   &RESLO,R7
        MOV   R15,&MPYS
        MOV   #25,&OP2
        nop
        Mov   &RESLO,R15
        MOV   R7,&MPY
        MOV   R15,&OP2
        nop
        Mov   &RESLO,I2TrLow
        MOV   &RESHI,I2TrHigh
```

```
//----------Ii-----------------------------------------------
-------------
     MOV    Ki,&MPYS
     MOV    Ir2,&OP2
        nop
        Mov    &RESLO,Ii
        CLRC
        RRC    Ii
        CMP    #3250,Ii   //1000A
        JL     AnIsd
        CMP    #6500,Ii   //2000A
        JL     AnIsd1000
        MOV   Ii,&MPY
        MOV   #63306,&OP2 //#55704,&OP2   //0,85 Ii
        nop
        MOV   &RESHI,Ii
        JMP   AnIsd
AnIsd1000:  MOV   Ii,&MPY
        MOV   #60948,&OP2   //0,93 Ii
        nop
        MOV   &RESHI,Ii
//-------------Isd-----------------------------------------
-------------
AnIsd:    MOV    Ksd,&MPYS
     MOV    Ir2,&OP2
        nop
        Mov    &RESLO,Isd
        CLRC
        RRC    Isd
        CMP    #3250,Isd   //1000A
        JL     AnTim
        CMP    #6500,Isd   //2000A
        JL     AnTim1000
        MOV   Isd,&MPY
        MOV   #63306,&OP2   //0,966 Isd
        nop
        MOV   &RESHI,Isd
        JMP   AnTim
AnTim1000:  MOV   Isd,&MPY
        MOV   #60948,&OP2   //0,93 Isd
        nop
        MOV   &RESHI,Isd
//----- Старт таймера A в continious mode TAclk=SMCLK/8=1MHz,
```

```
//Доступні переривання в порівнянні із вмістом регістра,
//виведення на індикацію,мітки часу
AnTim:  Mov   #MC1+TASSEL1+ID1+ID0,&TACTL
     MOV  #CCIE,&CCTL0
        MOV    &TAR,CCR0Old
        Mov    #5a00h+WdTCntCl+WDTHOLD,WdTCtl //Очищення WDT
        ADD         #2000,CCR0Old
        MOV    CCR0Old,&CCR0
//Запуск АЦП перетворення
//        BiS  #CSTARTADD_0,&Adc12Ctl1
     BIC  #ENC,&Adc12Ctl0
        Mov.B       #SREF_1+INCH_0,&ADC12MCTL8
        Mov.B       #SREF_1+INCH_1,&ADC12MCTL9
     Mov.B     #EOS+SREF_1+INCH_2,&ADC12MCTL10
     Mov       #CON-
SEQ_1+ADC12SSEL_3+SHP+ISSH+SHS_0+CSTARTADD_8,&ADC12CTL1
        Mov        #ADC12ON+REFON+REF2_5V+MSc+SHT1_2,&ADC12CTL0
          BIC  #BIT1,&ADC12IFG
        CLR   R11
        CLR   R12
        EINT
//******цикл підготовки 105мс*********************************
PrepStart:
        CMP     #525,R11
        JLO     PrepStart
        CLR     R11
     CLR  IAmeas
     CLR  IBmeas
     CLR  ICmeas
//******робочий цикл*****************************************
WaitStart:
        CMP     #5,R11
        JLO     WaitStart
//--------- прохід 1 раз за мс----------------------
     Mov  IAmeas,IAtek
     Mov  IBmeas,IBtek
     Mov  ICmeas,ICtek
        CLR     R11
     CLR  IAmeas
     CLR  IBmeas
     CLR  ICmeas
        INC     R12
        INC     R13
```

```
//-------аналіз миттєвого значення струму для струмообмеження
(струмової відсічки)---
        CMP     IBtek,IAtek
        JGE     GrIA
        CMP     ICtek,IBtek
        JGE     GrIB
GrIC:     Mov  ICtek,Imax
Ilim:   CMP     Imax,Ii
        JGE      NoWorkIlim
//-------робота струмообмеження---
        CMP     #0,Ihold
        JNZ     NoWrite
        Mov     Imax,Ihold
          Mov     #2,Shold
NoWrite: BIT     #Istart,Status
        JNZ     ContIstart
        BIS     #Istart,Status
        BIC     #Norm,Status
        MOV     Ti,Tist
        MOV     #100,Tires
ContIstart: DEC Tist
        JGE     WorkIlim
SetItrim: BIS   #Itrim,Status
        BIC     #Norm+Istart,Status
        BIC.B   #BIT0,&P4OUT  //Swith OFF
        JMP     WorkIlim
GrIA:   CMP     ICtek,IAtek
        JL      GrIC
     Mov  IAtek,Imax
        JMP     Ilim
GrIB:   CMP     ICtek,IBtek
        JL      GrIC
     Mov  IBtek,Imax
        JMP     Ilim
NoWorkIlim: BIT #Istart,Status
        JZ     WorkIlim
        DEC     Tires
        JGE     WorkIlim
        BIC     #Istart,Status
        BIS     #Norm,Status
WorkIlim:  CMP Imaxper,Isd
        JGE      NoWorkSD
```

```
//-------робота струмової відсічки---
        CMP     #0,Ihold
        JNZ     NoWrite2
        Mov     Imaxper,Ihold
          Mov     #1,Shold
NoWrite2:  BIT     #SDstart,Status
        JNZ     ContSDstart
        BIS     #SDstart,Status
        BIC     #Norm,Status
        MOV     Tsd,Tsdst
        MOV     #400,Tsdres
ContSDstart: DEC Tsdst
        JGE     WorkSD
SetSDtrim: BIS   #SDtrim,Status
        BIC     #Norm+SDstart,Status
        BIC.B   #BIT0,&P4OUT  //Swith OFF
NoWorkSD: BIT #SDstart,Status
        JZ     WorkSD
        DEC     Tsdres
        JGE     WorkSD
        BIC     #SDstart,Status
        BIS     #Norm,Status
//----опитування перемикачів,введення на індикацію-----------
--------------
WorkSD:   Mov.B &P1IN,R7
          INV   R7
          RRA   R7
          RRA   R7
          RRA   R7
          AND   #0x001e,R7
          CMP   #0x0014,R7
          JZ    Outnimb1
          CMP   #0x0016,R7
          JZ    Outnimb2
          CMP   #0x0018,R7
          JZ    Outnimb3
```

СПИСОК ЛІТЕРАТУРИ

1. Краткий терминологический словарь по микропроцессорной технике. – Москва : Международный центр науч. и техн. информации, 1984. – 104 с.

2. Микропроцессорные БИС и микроЭВМ / А. А. Васенков, Н. М. Воробьев, В. Л. Дшхунян и др. / под ред. А. А. Васенкова. – Москва : Сов. радио, 1980. – 280 с.

3. Грищук Ю. С. Микропроцесорные устройства : учеб. пособ. / Ю. С. Грищук. – Харьков : НТУ «ХПИ», 2007. – 280 с.

4. Напрасник М. В. Микропроцессоры и микроЭВМ : учеб. пособ. / М. В. Напрасник. – Москва : Высш. шк., 1989. – 192 с.

5. Гилмор Ч. Введение в микропроцессорную технику / Ч. Гилмор ; пер. с англ. – Москва : Мир, 1984. – 334 с.

6. Акушский И. Я. Машинная арифметика в остаточных классах / И. Я. Акушский, Д. И. Юдицкий. – Москва : Сов. радио, 1986. – 440 с.

7. Лысиков Б. Г. Арифметические и логические основы автоматов : учебник для вузов / Б. Г. Лысиков. 2-е изд. – Минск : Высш. шк., 1980. – 336 с.

8. Микропроцессоры. – В 3 т. – Т. 1. Архитектура и проектирование микроЭВМ. Организация вычислительных процессов / П. В. Нестеров, В. Ф. Шаньеин, В. Л. Горбунов и др. / под ред. Л. Н. Преснухина. – Москва : Высш. шк., 1986. – 495 с.

9. Микропроцессоры. – В 3 т. – Т. 2. Средства сопряжения. Контролирующие и информационно управляющие системы / В. Д. Вернер, Н. В. Воробьев, А. В. Горячев и др. / под ред. Л. Н. Преснухина. – Москва : Высш. шк., 1986. – 383 с.

10. Алексенко А. Г. Проектирование радиоэлектронной аппаратуры на микропроцессорах / А. Г. Алексенко, А. А. Галицин, А. Д. Иванников. – Москва : Радио и связь, 1984. – 272 с.

11. Методичні вказівки до лабораторних робіт з курсу «Мікропроцесорні пристрої» для студентів спеціальності 092206 «Електричні машини та апарати» / Уклад.: Ю. С. Грищук, Т. П. Павленко. – Харків : ХДПУ, 1999. – 32 с.

12. Современный компьютер : сб. научно-популярных статей ; пер. с англ. / под ред. В. М. Курочкина. – Москва : Мир, 1986. – 212 с.

13. Балашов Е. П. Микропроцессоры и микропроцессорные системы / Е. П. Балашов, Д. В. Пузанков. – Москва : Радио и связь, 1981. – 328 с.

14. Горбунов Л. П. Микропроцессоры. Основы построения микроЭВМ / Л. П. Горбунов, Д. И. Панфилов, Д. Л. Преснухин. – Москва : Высш. шк., 1984. – 144 с.

15. Щербаков О. А. Физические основы записи информации в ПЗУ / О. А. Щербаков // Микропроцессорные средства и системы. – 1985. − № 3. – С. 72–75.

16. Лебедев О. Н. Микросхемы памяти и их применение / О. Н. Лебедев. – Москва : Радио и связь, 1990. – 234 с.

17. Бобков В. А. Расширенный микропроцессорный комплект БИС се-

рии К588 / В. А. Бобков, Б. Н. Чернуха, В. С. Свиридович, В. П. Ключников // Микропроцессорные средства и системы. – 1987. – № 1. – С. 6–10.

18. Гитис Э. И. Аналого-цифровые преобразователи / Э. И. Гитис, Е. А. Пискунов. – Москва : Энергоиздат, 1981. – 360 с.

19. Мікропроцесорна техніка : підручник / Ю. І. Якименко, Т. О. Терещенко, Є. І. Сокол та ін. / за ред. Т. О. Терещенко. – Київ : Політехнік, 2003. – 440 с.

20. Сташин В. В. Проектирование цифровых устройств на однокристальных микроконтроллерах / В. В. Сташин, А. В. Урусов, О. Ф. Мологонцева. – Москва : Энергоатомиздат, 1990. – 224 с.

21. Алексієв О. П. Мікроконтролери для транспортних і промислових застосувань.: архітектура та програмування : навч. посіб. / О. П. Алексієв, О. Б. Богаєвський, В. П. Волков. – Харків : ХНАДУ, 2004. – 156 с.

22. Встраиваемый микроконтроллер 8XC251SB : руководство пользователя. – Київ : «Квазар – Микро», 1995. – 379 с.

23. Ульрих В. А. Микроконтроллеры PIC16X7XX / В. А. Ульрих. – Изд. 2-е, перераб. и доп. / под ред. С. Л. Корякина-Черняка. – Санкт-Петербург : Наука и техника, 2002. – 320 с.

24. Костин Г. Ю. Микроконтроллеры фирмы Motorola / Ю. Г. Костин. – Киев : КТЦ–МК, 1995. – 37 с.

25. Методические указания к изучению курса «Микропроцессорные устройства» для студентов специальности 092206 «Электрические машины и аппараты» / сост. Ю. С. Грищук. – Харьков : НТУ «ХПІ», 2001. – 24 с.

26. Методичні вказівки до лабораторних робіт з курсу «Мікропроцесорні пристрої». – Ч. 2 : «Однокристальні мікро контролери» для студентів спеціальностей 092206 «Електричні машини та апарати» і 092205 «Електропобутова техніка» усіх форм навчання / уклад. Ю. С. Грищук. – Харків : НТУ «ХПІ», 2003. – 43 с.

27. Устименко Д. В. Применение микроконтроллеров в схемах электроподвижного состава / Д. В. Устименко // Вестник НТУ «ХПИ» : сб. науч. трудов. – Вып. 11. – Харьков : НТУ «ХПИ». – 2003. – С. 126–128.

28. Процессоры цифровой обработки сигналов фирмы «TEXAS INSTRUMENTS». – Москва : ЗАО СКАНТИ-РУС, 2001. – 35 с.

29. Грищук Ю. С. Автоматизированная система управления для коммутационных исследований и испытаний электрических аппаратов / Ю. С. Грищук, А. Н. Ржевский, С. Ю. Грищук // Вестник НТУ «ХПИ». сб. науч. трудов. – Вып. 17. – Харьков : НТУ «ХПИ», 2001. – С. 48–50.

30. Грищук Ю. С. Применение микроконтроллеров в схемах автоматизированного управления испытаниями электрических аппаратов / Ю. С. Грищук, А. И. Кузнецов, А. Н. Ржевский, С. Ю. Грищук // Вісник НТУ «ХПІ» : зб. наук. праць. – Харків : НТУ «ХПІ», 2005. – Вип. 35 – С. 63–68.

31. Комплектное распределительное устройство серии КУ–10Ц : Техническая информация НКАИ. 670049.003 и 670049.007. – Ровно : СП РЗВА, 1998. – 24 с.

32. Микропроцессорные гибкие системы релейной защиты / В. В. Михайлов, Е. В. Кириевский, Е. М. Ульяницкий и др. ; под ред. В. П. Морозкина. – Москва : Энергоатомиздат, 1988. – 240 с.

33. Дзербицкий С. Испытания электрических аппаратов / С. Дзербицкий. – Ленинград : Энергия, 1975. – 204 с.

34. Петинов О. В. Испытания электрических аппаратов / О. В. Петинов, Е.Ф. Щербаков. – Москва : Высш. шк., 1985. –215 с.

35. Евстифеев А. В. Микроконтроллеры AVR семейств Tiny и Mega фирмы «Atmel» / А. В. Евстифеев. – Москва : Издательский дом «Додэка XXI», 2004. – 560 с.

36. Семейство микроконтроллеров MSP430x1xx. Руководство пользователя : пер. с англ. – Москва : Серия «Библиотека Компэла». ЗАО «Компэл», 2004. – 368 с.

37. Грищук Ю. С. Автоматизація досліджень мультиварок / Ю. С. Грищук, М. Г. Пантелят, А. К. Єлоєв // Вісник НТУ «ХПІ». Сер. : Проблеми удосконалення електричних машин і апаратів. – 2018. – № 28 (1304). – С. 16–21.

38. Грищук Ю. С. Мікроконтролерні лабораторні стенди для дослідження електричних апаратів / Ю. С. Грищук, А. Є. Вишневський // Вісник НТУ «ХПІ». Серія : Проблеми удосконалення електричних машин і апаратів. – 2010. – № 29 (1204). – С. 18–24.

39. Грищук Ю. С. Застосування мікроконтролерів при дослідженнях електричних апаратів / Ю. С. Грищук // Вісник НТУ «ХПІ». Серія : Проблеми удосконалення електричних машин і апаратів. – 2016. – № 32 (1204). – С. 23–28.

40. Виговський В. С. Автоматизація керування живильними насосами енергоблоку потужністю 200 МВт / В. С. Виговський, Ю. С. Грищук // Вісник НТУ «ХПІ». Серія : Проблеми удосконалення електричних машин і апаратів. – 2015. – № 13 (1122). – С. 20–31.

41. Грищук Ю. С. Аналіз мікропроцесорного терміналу шафи керування обігрівом тунелів метрополітену / Ю. С. Грищук, С. Л. Зуєнко // Вісник НТУ «ХПІ». Серія : Проблеми удосконалення електричних машин і апаратів. – 2014. – № 20 (1063). – С. 8–13.

42. Зарвиро В. О. Автоматизація дослідження та випробовування мікрохвильових печей / В. О. Зарвиро, Ю. С. Грищук // Вісник НТУ «ХПІ». Серія : Проблеми удосконалення електричних машин і апаратів. – 2010. – № 32 (1204). – С. 23–28.

43. Грищук Ю. С. Мікроконтролерний розчіплювач для автоматичних вимикачів / Ю. С. Грищук, Т. В. Сухоставцева // Вісник НТУ «ХПІ» : зб. наук. праць. – Харків : НТУ «ХПІ», 2008. – № 25. – С. 29–35.

44. Электронный блок защит автоматического выключателя АВ3004/3Б. Руководство по эксплуатации ПФ.АВ3004.002.031 РЭ. – Харків : НТУ «ХПІ», 2010. – 11 с.

Навчальне видання

ГРИЩУК Юрій Степанович

МІКРОКОНТРОЛЕРИ: АРХІТЕКТУРА, ПРОГРАМУВАННЯ ТА ЗАСТОСУВАННЯ В ЕЛЕКТРОМЕХАНІЦІ

Навчальний посібник
для студентів електромеханічних спеціальностей

Роботу до видання рекомендував *проф. Борисенко А. М.*
Редактори *О. С. Самініна*, *Н. В. Верстюк*

План 2018 р., поз. 138

Підп. до друку 05.03.2019 р. Формат 60x84 $^{1}/_{16}$. Папір офсетний. Друк – ризографія. Гарнітура Times New Roman. Ум. друк. арк. 22,9 Наклад 300 прим. Зам. № Ціна договірна

Видавничий центр НТУ «ХПІ». 61002, Харків, вул. Кирпичова, 2
Свідоцтво про державну реєстрацію ДК № 5478 від 21. 08. 2017 р.

Самостійне електронне видання