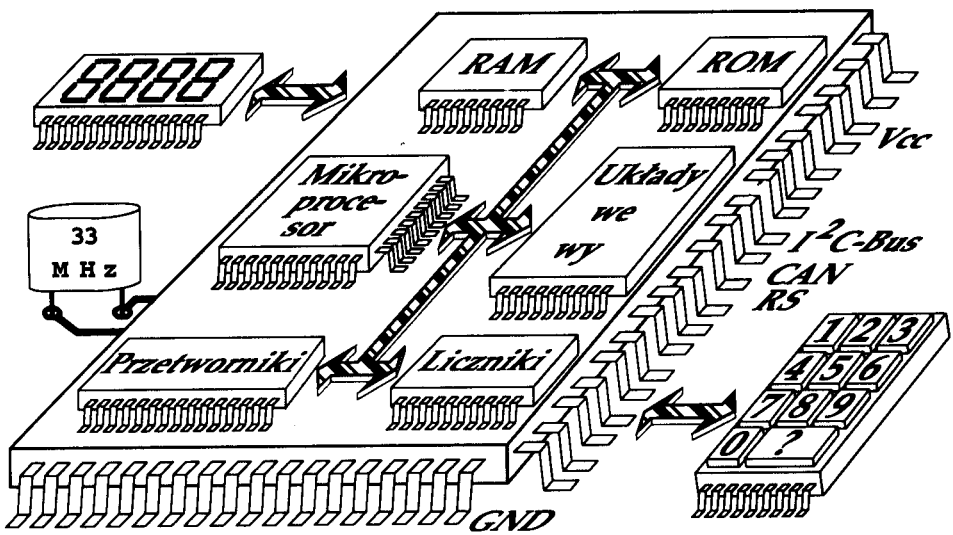
Systemy mikroprocesorowe



Mikrokontrolery

Janusz Janiczek
Andrzej Stępień

Spis treści:

1. Dlaczego mikrokontrolery '51 ?

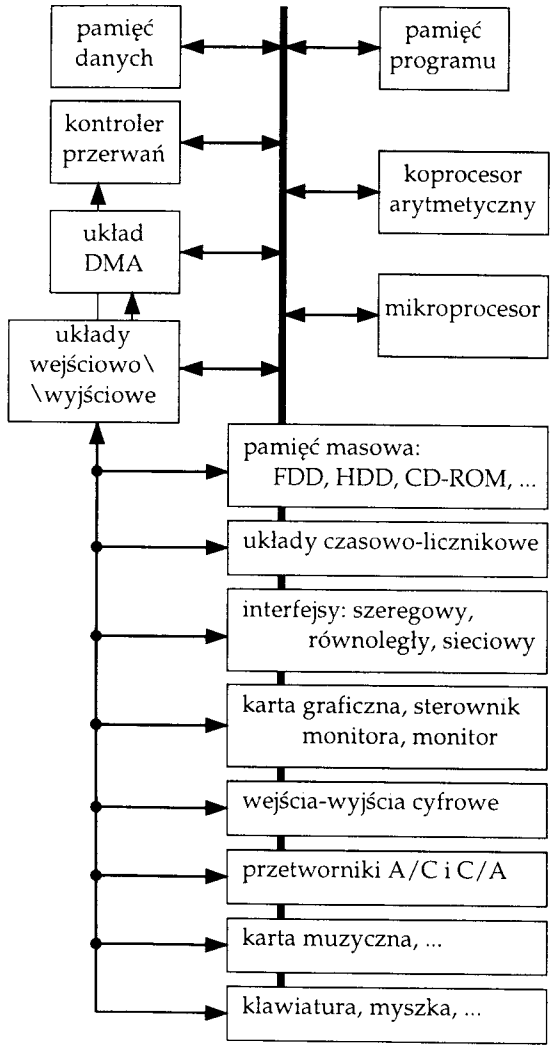
Gwałtowny rozwój elektroniki spowodował, że komputery nie wywołują już dreszczyka emocji. Stały się wręcz niezbędne i niezastąpione, a ich awarie wprowadzają chaos w życiu codziennym. Bardzo złożone i rozbudowane komputery, a tym samym bardzo drogie, przeznaczone są dla celów militarnych, obliczeń naukowych oraz symulacji zjawisk fizycznych, chemicznych, meteorologicznych itp. W tej grupie znajdują się również komputery tworzące tzw. efekty specjalne w kinematografii i telewizji, do kreowania sztucznego, wirtualnego świata. Na drugim biegunie znajdują się proste, niedrogie, powszechnie stosowane sterowniki mikroprocesorowe. Można znaleźć je na przykład w:

- sprzęcie powszechnego użytku, np. w odbiornikach radiowych i telewizyjnych, magnetowidach; służą do programowania funkcji, czasu włączenia i wyłączenia odbiornika, cyfrowego sterowania parametrami odbioru lub odtwarzania itd.
- technice motoryzacyjnej do kontroli i sterowania silników spalinowych z zapłonem iskrowym i samoczynnym, urządzeń klimatyzacyjnych, układów antypoślizgowych (np. ABS), testowania stopnia zużycia niektórych elementów, np. amortyzatorów,
- aparatach fotograficznych i kamerach video do określania optymalnych warunków naświetlania błony filmowej lub korekcji układów optycznych sprzężonych z fotoelementami, ustawienia ostrości,
- artykułach gospodarstwa domowego, np. w automatycznych pralkach do nadzoru temperatury i ilości wody, środków piorących, czasu trwania poszczególnych faz prania (wstępne, zasadnicze i płukanie).

Z pewnym uproszczeniem można założyć, że komputer klasy IBM-PC złożony jest z następujących elementów (rysunek 1-1):

- mikroprocesora realizującego wymagane programem obliczenia,
- pamięci programu i danych,
- różnego typu pamięci masowych takich jak: HDD, FDD, CD-ROM,
- układu bezpośredniego dostępu do pamięci (DMA) przy wymianie danych bez pośrednictwa mikrokontrolera,
- układu kontrolera przerwań wraz z kontrolerem priorytetów,
- układów czasowych i liczników taktowanych wewnętrznym generatorem komputera,
- układów wejściowo-wyjściowych do komunikacji komputera z otoczeniem, przyjmowania danych i wysyłania efektów ich przetwarzania, np. wejść-wyjść cyfrowych, interfejsu szeregowego (RS232), równoległego (Centronics), a także układów sieciowych (Novell),
- układów lub dodatkowych kart przetworników A/C i C/A,

- klawiatury, myszki itp.
- karty graficznej zawierającej sterownik monitora oraz monitor np. LCD,
- innych kart zgodnie z wymaganiami użytkownika, np. dźwiękowej.



Rys. 1-1 Upro. czony schemat blokowy komputera klasy IBM-PC.

Mikroprocesor jako programowalny układ scalony o dużym stopniu integracji zawiera jedynie jednostkę arytmetyczno-logiczną i część układu sterowania. Pozostałe elementy traktowane są jako układy zewnętrzne. W rozbudowanych komputerach taki podział ułatwia zadanie konstruktorom sprzętu i oprogramowania ze względu na budowę modułową. Taką koncepcję przyjęto

przy realizacji komputerów osobistych. To co jest zaletą w dużych systemach jest wadą w małych. Rozproszone, małe systemy kontrolno-pomiarowe oprócz samego mikroprocesora wraz z pamięcią programu i danych potrzebują najczęściej także:

- wejść i wyjść cyfrowych,
- układu czasowego lub licznika taktowanego zewnętrznym sygnałem,
- łącza szeregowego do komunikacji z nadrzędnym komputerem,
- prostego kontrolera przerwań,
- układu redukcji mocy przy zasilaniu bateryjno-akumulatorowym,
- generatora taktującego, stabilizowanego rezonatorem kwarcowym.

Producenci mikroprocesorów oferują również wszystkie, niezbędne układy zewnętrzne. W prostych systemach komputerowych ich możliwości nie są wykorzystywane. Należy również pamiętać, że każdy z układów zajmuje miejsce na płytce drukowanej, wymaga niezbędnych połączeń z innymi układami. Komplikuje to konstrukcję płytek drukowanych, zmniejsza niezawodność systemu oraz, co jest równie ważne, zwiększa pobór prądu z zasilacza w porównaniu z jednym układem, który pełniłby te same funkcje. Względy te oraz postęp w technologii wytwarzania układów scalonych spowodowały, że niektóre elementy peryferyjne wykonano łącznie z mikroprocesorem na jednej płytce krzemu. W ten sposób w 1977 roku firma Intel opracowała pierwszy 8-bitowy mikroprocesor jednoukładowy o symbolu 8048. Trzy lata później na rynku pojawił się także 8-bitowy mikroprocesor jednoukładowy 8051, który zapoczątkował powstanie całej rodziny mikroprocesorów oznaczonych symbolem '51. Ze względu na możliwości i pełnione funkcje mikroprocesory jednoukładowe nazwano mikrokontrolerami. Mówiąc o 8-bitowym mikrokontrolerze mówi się o wielkości, liczbie linii magistrali danych, sposobie przetwarzania i przesyle informacji.

Obecnie rodzina mikrokontrolerów '51 liczy kilkadziesiąt typów i znajduje się w ofercie dużych jak i mniejszych, światowych producentów mikroprocesorów takich jak: Intel, AMD, Philips, Signetics, Siemens, Oki, Matra/Harris, Dallas, Atmel itd.

Mikrokontroler 8051 w zamierzeniach jego twórców nie był strukturą zamkniętą i już od początku przygotowany był do rozbudowy. Pierwsza zmiana wprowadzona przez samą firmę Intel związana była z rozszerzeniem wewnętrznej pamięci RAM oraz pamięci programu i wprowadzenie trzeciego układu czasowego, licznika T2(8052). Kolejne zmiany były już tylko kwestią czasu.

Podstawowymi zaletami mikrokontrolera 8051 są:

- stała lista instrukcji, niezależnie od rozbudowy układów wewnętrznych,
- sprzętowe procedury mnożenia i dzielenia,

- wewnętrzna pamięć programu i danych,
- dwa programowalne układy czasowo-licznikowe,
- niezależny od jednostki arytmetyczno-logicznej układ transmisji szeregowej,
- duża liczba, jak na tamte czasy, wejść i wyjść cyfrowych,
- wewnętrzny generator sterujący działaniem mikrokontrolera,
- priorytetowy, 2-poziomowy układ kontroli przerwań, wystarczający przy niewielkiej liczbie wewnętrznych układów.

Wielu konstruktorom mikroprocesorowych sterowników brakowało możliwości szybkiej wymiany danych, współpracy z innymi zewnętrznymi układami na wspólnej szynie (np. tryb DMA-bezpośredniego dostępu do pamięci) oraz współpracy z wolniej (w stosunku do mikrokontrolera) działającymi układami zewnętrznymi lub pamięciami. Obie wady zostały częściowo usunięte. Tryb DMA wprowadziła firma Philips w mikrokontrolerach PCF 8xC592/598. Programowanie czasu trwania sygnałów $\overline{RD}$ (ReaD) i $\overline{WR}$ (WRite) sterujących zewnętrznymi pamięciami RAM możliwe jest w mikrokontrolerach firmy Dallas, np. DS8xC320 lub DS8xC5x0. Firma ta wprowadziła w mikrokontrolerze DS8xC530 dodatkowy wewnętrzny układ RTC (Real Time Clock) oraz rozbudowane i energooszczędne tryby pracy, co jest szczególnie ważne przy zasilaniu bateryjnym.

Nowe konstrukcje mikrokontrolerów 16-bitowych, np. 8xC251 firmy Intel lub 8xC51XA firmy Philips, są zgodne układowo i programowo z mikrokontrolerem 8051. To świadczy o zakresie zastosowań mikrokontrolerów rodziny '51, przyzwyczajeniach programistów, niezbyt wysokich kosztach oprogramowania wspomagającego testowanie i uruchamianie oraz bardzo bogatej biblioteki oprogramowania, kompilatorów języków wyższego poziomu itd.

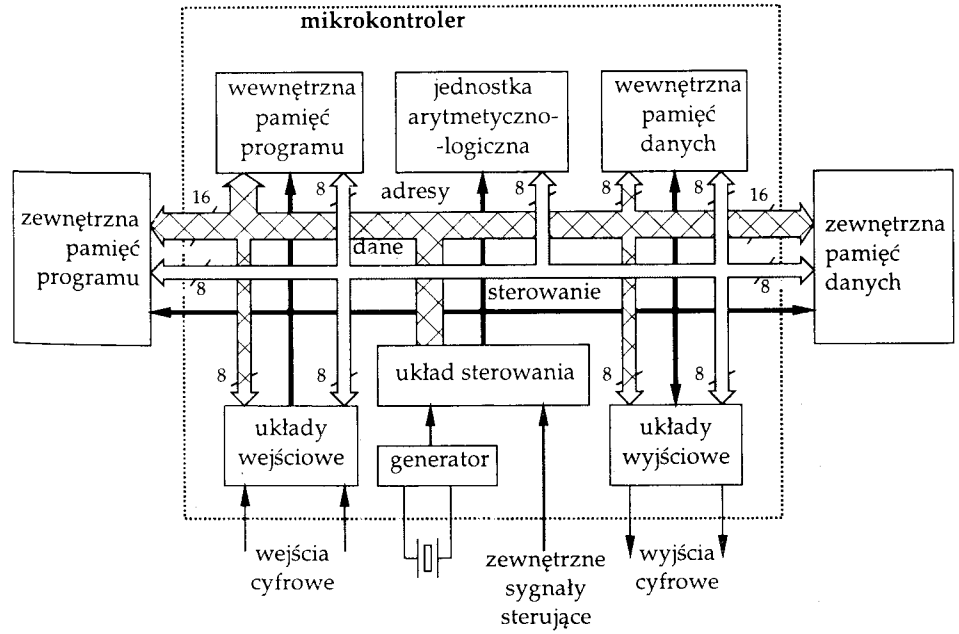
Mikrokontroler 8051 potęgą jest i basta.

2. Co jest w środku ?

Mikroprocesor jest układem o dużym stopniu scalenia (dużej skali integracji), którego działanie zależy od programu, ciągu rozkazów. Program może być dowolnie zmieniany przez użytkownika w ramach dostępnej listy rozkazów. Dla mikrokontrolerów rodziny '51 lista rozkazów jest stała, nie zależy od liczby wewnętrznych układów i liczy 111 instrukcji. Realizowany program i przetwarzane dane przechowywane są w:

- pamięci stałej programu, której zawartość nie jest zmieniana w trakcie wykonywania programu; oprócz samego programu w pamięci tej przechowywane są również niektóre stałe, tablice kodujące itp.,
- pamięci danych, której zawartość jest ciągle modyfikowana, zmieniana; pamięć ta zawiera dane do obliczeń i wyniki tych obliczeń.

Im bardziej złożone obliczenia lub sterowanie tym większa liczba danych musi być przechowywana w pamięci danych. Ze względu na niezbyt dużą wewnętrzną pamięć danych w mikrokontrolerach rodziny '51, konieczne jest dołączenie zewnętrznej pamięci danych. Ten sam problem występuje w przypadku pamięci programu. Mikrokontrolery rodziny '51 umożliwiają dołączenie zewnętrznych pamięci programu i danych, tak jak przedstawiono to na rysunku 2-1.



Rys. 2-1 Magistrale mikrokontrolera 8051.

Wewnętrzna lub zewnętrzna pamięć programu jest pamięcią typu ROM (Read Only Memory). Jej zawartość jest ustalana w trakcie produkcji mikrokontrolera. Takie rozwiązanie jest możliwe jedynie w przypadku długich, liczących co najmniej kilka tysięcy mikrokontrolerów z tym samym programem. Jeśli serie są krótsze, a w szczególności są to pojedyncze egzemplarze, prototypy, jako pamięci wykorzystuje się:

- pamięci EPROM, programowane elektrycznie ale kasowane optycznie przez naświetlanie promieniami UV (promieniowanie ultrafioletowe); wiele pamięci jednokrotnie programowanych (OTP) jest wykonywanych jako pamięci EPROM,
- pamięci flash ROM, programowane i kasowane elektrycznie, które są znacznie tańsze niż pamięci EPROM, a ich kasowanie trwa kilkanaście ms (kasowanie pamięci EPROM trwa zwykle kilka minut).

Oba typy pamięci mogą być pamięciami zarówno wewnętrznymi jak i zewnętrznymi.

Do przechowywania danych, które są zmieniane w trakcie wykonywania programu służą pamięci RAM (Random Access Memory). Wewnętrzna pamięć RAM w mikrokontrolerze 8051 liczy tylko 128 bajtów, a w mikrokontrolerze 8052 i nowszych zwykle 256 bajtów. Rozmiar zewnętrznej pamięci danych zależy tylko od potrzeb użytkownika ale nie może przekroczyć 64 KB(ajtów). Problem polega na tym, że do wymiany danych z zewnętrzną pamięcią RAM konstruktorzy mikrokontrolera przewidzieli tylko 2 rozkazy do zapisu danych i 2 rozkazy do ich odczytu. We wszystkich wymianach bierze udział wyłącznie jeden wyspecjalizowany wewnętrzny rejestr. Powoduje to znaczne spowolnienie wykonywanego programu. Dlatego jeśli czas wykonywania obliczeń ma być krótki, to do przechowywania danych należy wykorzystywać tylko wewnętrzną pamięć danych.

Magistrale (adresowa, danych i sterująca) występujące w mikrokontrolerach rodziny '51 mają różną wielkość, w zależności od tego, czy dotyczą wnętrza mikrokontrolera, czy komunikują się z zewnętrznymi pamięciami (rysunek 2-1):

- magistrala adresowa:
  → jeśli współpracuje z wewnętrzną lub zewnętrzną pamięcią programu ROM oraz z zewnętrzną pamięcią danych RAM liczy 16 linii, co powoduje, że maksymalna wielkość tych pamięci ograniczona jest do 2^{16}= 64 KB(ajtów); pełny adres zawierają 2 bajty,
  → jeśli współpracuje z wewnętrzną pamięcią danych RAM i wewnętrznymi układami peryferyjnymi (liczniki, łącze szeregowe itp.) liczy tylko 8 linii; z tego powodu wewnętrzna pamięć RAM ograniczona jest do 2^8=256 bajtów,
- wewnętrzną i zewnętrzną magistralę danych tworzy 8 linii; dane przesyłane są więc bajtowo,

- magistrala sterująca jest bardzo zróżnicowana. Liczba linii tworzących tę magistralę zależy od typu układu. Dla pamięci jest to 1 lub 2 linie, dla układu sterowania znacznie więcej, zależnie od typu mikrokontrolera (liczników, przerwań itp.).

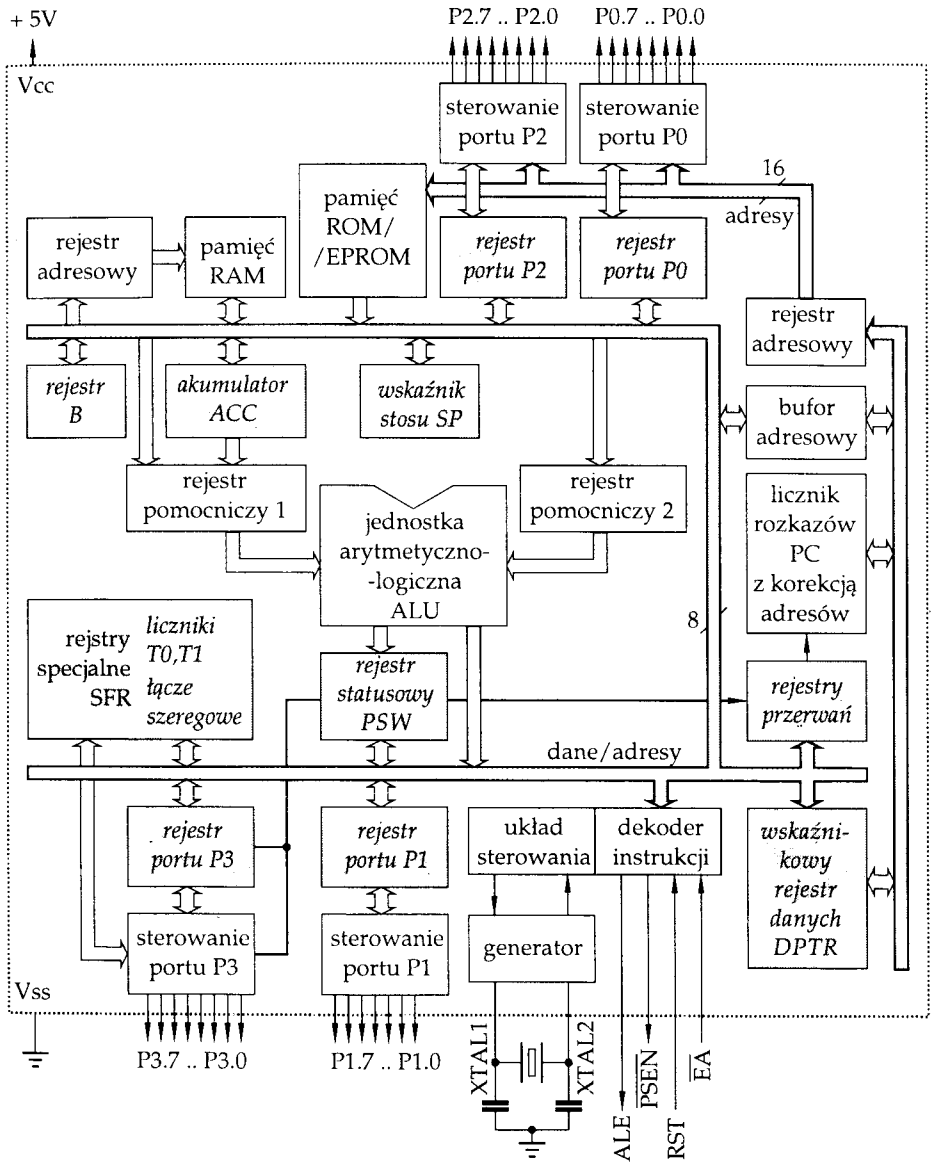
Struktura wewnętrzna mikrokontrolera 8051 jest bardziej złożona niż przedstawiono to na rysunku 2-1. Złożoność jest efektem większej liczby wewnętrznych elementów niż udostępnionych użytkownikowi, programiście. Są to przede wszystkim dodatkowe rejestry wykorzystywane przez jednostkę arytmetyczno-logiczną w trakcie wykonywania niektórych wewnętrznych przesłań, bufory portów P0 .. P3, układy korekcji stanu licznika rozkazów, rejestry dekodujące realizowane instrukcje itd. Schemat blokowy mikrokontrolera 8051 przedstawia rysunek 2-2.

Mikrokontroler 8051 oferowany jest przez wiele firm w obudowie z 40 wyprowadzeniami. Jeśli policzyć wszystkie elementy struktury wewnętrznej mikrokontrolera, które mikrokontroler wykorzystuje do komunikowania się z układami zewnętrznymi, tzn.:

- 8-bitową magistralę danych,
- 16-bitową magistralę adresową,
- linie sterujące dostępem do zewnętrznej pamięci programu ($\overline{EA}$, $\overline{PSEN}$) i zewnętrznej pamięci danych ($\overline{RD}$, $\overline{WR}$),
- cztery 8-bitowe porty pełniące rolę wejść lub wyjść cyfrowych (P0, .. , P3),
- wejście i wyjście układu transmisji szeregowej,
- po dwie linie sterujące pracą każdego z dwóch wewnętrznych liczników,
- dwa wejścia zewnętrznych przerwań,
- dwie linie do dołączenia rezonatora kwarcowego, jedna linia sprzętowego zerowania mikrokontrolera oraz dwa wyprowadzenia do dołączenia zasilania mikrontrolera,

to okaże się, że liczba wyprowadzeń mikrokontrolera jest znacznie mniejsza niż wymienione potrzeby. Takie rozwiązanie jest możliwe wskutek pełnienia wielu różnych funkcji przez jedno i to samo wyprowadzenie:

- porty P0 i P2 służą do przesyłania adresów do zewnętrznej pamięci programu i danych,
- przez port P0 przesyłane są dodatkowo wszystkie dane z/do zewnętrznej pamięci programu i danych,
- port P3 pełni rolę wejść wewnętrznych liczników, łącza szeregowego, wejść kontrolera przerwań oraz steruje kierunkiem przesyłem danych z/do zewnętrznej pamięci danych (linie $\overline{RD}$, $\overline{WR}$).



Rys. 2-2 Schemat blokowy mikrokontrolera 8051.

Takie przyporządkowanie wymaga multipleksowania stanów linii portów. Oznacza to, że w określonych momentach czasowych stany tych linii zależą od aktualnie wykonywanej przez mikrokontroler funkcji. Wprowadza to pewne ograniczenia. Nie można korzystać z portów P0 i P2 jeśli konstruktor dołączy do mikrokontrolera zewnętrzną pamięć programu lub danych. Także wykorzystanie linii portu P3 jest ograniczone jeśli planuje się uaktywnienie łącza szeregowego itd.

Generator umożliwia poprzez dołączenie zewnętrznego rezonatora kwarcowego lub ceramicznego ustalenie żądanej częstotliwości pracy mikrokontrolera. Sygnał z generatora synchronizuje pracę całego procesora oraz służy do określenia momentów czasu, w których pojawiają się sygnały sterujące wymianą danych mikrokontrolera z otoczeniem, np. sygnały $\overline{RD}$ (odczytu) i $\overline{WR}$ (zapisu) z/do zewnętrznej pamięci danych.

Do adresowania pamięci programu służy 16-bitowy **licznik rozkazów PC** (Program Counter). Szesnastobitowy adres umożliwia zaadresowanie 2^{16}= 65 536 komórek pamięci, tzn. 64 KB. Należy pamiętać, że 1 KB = 2^{10} = 1024 bajty. Z zaadresowanej komórki pamięci programu procesor pobiera rozkaz, który złożony jest z jednego, dwóch lub trzech bajtów. Po pobraniu każdego bajtu rozkazu następuje zwiększenie licznika rozkazów PC o jeden (licznik rozkazów jest inkrementowany). W ten sposób licznik rozkazów PC wskazuje na adres pierwszego bajtu następnego do wykonania rozkazu. Przy pobieraniu rozkazów z zewnętrznej pamięci programu jej adres przesyłany jest za pośrednictwem portu P0 i P2, a bajty rozkazów przesyłane są poprzez port P0. Zawartość licznika rozkazów PC może być zmieniana odpowiednimi rozkazami, co umożliwia wykonywanie skoków do różnych fragmentów programu.

Jednostka arytmetyczno-logiczna wykonuje podstawowe działania arytmetyczne i logiczne na danych. Należą do nich dodawanie arytmetyczne (binarne i dziesiętne), odejmowanie, mnożenie, dzielenie, porównanie, przesunięcie o jeden bit w lewo lub w prawo, suma i iloczyn logiczny, różnica symetryczna oraz negacja. Wynik operacji wykonywanych w jednostce arytmetyczno-logicznej może mieć wpływ na kolejność wykonywanych rozkazów, działania programu.

Z jednostką arytmetyczno-logiczną są funkcjonalnie powiązane trzy rejestry o ściśle sprecyzowanym przeznaczeniu:

- akumulator **A** lub **ACC** (Accumulator) służy przede wszystkim do przechowywania jednego z argumentów (danej) w operacjach arytmetycznych i logicznych, a po ich wykonaniu do przechowywania wyniku operacji. Jednak część operacji logicznych wykonywana jest poza akumulatorem. Tylko w akumulatorze można przesuwać daną o jeden bit w lewo lub w prawo. Akumulator, co jest jego zasadniczą wadą, uczestniczy przy przesyłaniu wszystkich danych z/do zewnętrznej pamięci danych oraz pobierania argumentów z pamięci programu.

- pewne cechy wyników operacji wykonywanych w jednostce arytmetyczno-logicznej są zapamiętywane w **rejestrze statusowym PSW** (Program Status Register). Wybrane bity tego rejestru informują o wyniku wykonywanych działań:
  → przekroczenia zakresu liczb całkowitych ze znakiem i bez znaku w operacjach dodawania i odejmowania,
  → korekcji sumy liczb przedstawionych w kodzie BCD,

→ próbie dzielenia przez zero,
→ parzystej liczbie jedynek w akumulatora,
→ wyborze grupy dostępnych rejestrów wewnętrznych.

- w operacjach mnożenia i dzielenia jeden z argumentów przechowywany jest w akumulatorze, a drugi w **rejestrze B**. Podobnie wynik wykonanej operacji. Jeśli nie przewiduje się użycia tych rozkazów, to rejestr B może być wykorzystany do innych celów.

W czasie wykonywania programu pobierane są dane lub stałe zapisane w pamięci programu (ROM). Przechowywanie dużej liczby zmiennych, np. wyników obliczeń, pomiarów lub transmitowanych danych, wymaga dołączenia do mikrokontrolera zewnętrznej pamięci danych (RAM), jeśli wewnętrzna pamięć jest za mała. Do adresowania obu typów pamięci wykorzystywany jest 16-bitowy **wskaźnikowy rejestr danych DPTR** (Data Pointer Register).

Jak w każdym komputerze także i w mikrokontrolerach rodziny '51 do wywoływania podprogramów, przechowywania argumentów przenoszonych z jednego podprogramu przy przejściu do wykonywania innego podprogramu tworzony jest stos. Stosem jest wydzielony fragment wewnętrznej pamięci RAM adresowany **wskaźnikiem stosu SP** (Stack Pointer). Kolejność odczytu bajtów ze stosu jest odwrotna w stosunku do kolejności ich zapisywania, zgodnie z regułą pamięci LIFO (Last In First Out). Bezpośrednio dostępne są tylko bajty adresowane przez wskaźnik stosu SP znajdujące się na wierzchołku stosu, a nie w jego wnętrzu. Sytuację tę można porównać do stery książek ułożonych jedna na drugiej. Dokładając kolejną książkę umieszcza się ją na szczycie stery. Zabierając książkę ze sterty zabiera się książkę leżącą na jej szczycie. Książki leżące we wnętrzu sterty nie są dostępne.

Wszystkie rejestry procesora , poza licznikiem rozkazów PC i wskaźnikowym rejestrem danych DPTR, są rejestrami 8-bitowymi. Umieszczone są w wewnętrznej przestrzeni adresowej co znacznie upraszcza przesyłanie danych, zarówno wewnątrz mikrokontrolera jak i na zewnątrz oraz ułatwia programowanie takich elementów jak liczniki, układ transmisji szeregowej itp.

Rejestry specjalne SFR (Special Function Register) mają szczególne znaczenie (na rysunku 2-2 zostały zacieniowane). Każdy element struktury wewnętrznej ma swoją reprezentację w postaci 1-bajtowego lub 2-bajtowego rejestru. Wpisanie informacji do tych rejestrów inicjuje działanie poszczególnych bloków funkcjonalnych mikrokontrolera. Dla przykładu wpisanie dowolnej wartości do rejestru SBUF rozpoczyna transmisję szeregową, a wpisanie jedynki logicznej na pozycję czwartego bitu w rejestrze TCON uruchamia licznik T0 itd. Rejestry specjalne są pomostem między programem, a wszystkimi urządzeniami peryferyjnymi mikrokontrolera (licznikami, portami równoległymi i szeregowym, strukturą przerwań itp.).

Mikrokontroler 8051 zawiera następujące wewnętrzne układy peryferyjne mające swoje odzwierciedlenie w rejestrach specjalnych:

- dwa 16-bitowe liczniki T0 i T1,
- układ portu szeregowego z programowaną szybkością transmisji, synchroniczną lub asynchroniczną,
- kontroler przerwań reagujący na dwa sygnały zewnętrzne i trzy sygnały wewnętrzne wytwarzane przy przepełnieniu licznika T0 i T1, zakończenia nadawania lub odbioru danych łączem szeregowym,
- układ redukcji mocy pobieranej przez mikrokontroler, co jest szczególnie istotne przy konstrukcji urządzeń zasilanych bateryjnie.

? Pytania i problemy

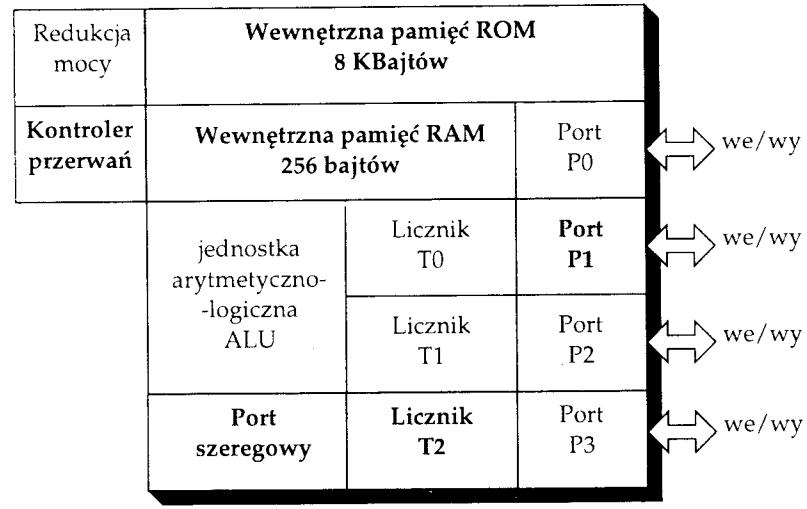
1. 1 KB to ile bajtów ?
2. Z ilu linii złożona jest magistrala danych i adresowa w mikrokontrolerze 8051 ?
3. Czym różni się pamięć EPROM i flash ROM ?
4. Czy do mikrokontrolera 8051 można dołączyć 128 Kbajtową zewnętrzną pamięć programu lub danych ?
5. Patrząc na rysunek 2-2 wymienić przeznaczenie portu P0, P2 i P3.
6. Jaką rolę pełni licznik rozkazów PC ?
7. Do czego służy jednostka arytmetyczno-logiczna ?
8. Jak określany jest status mikrokontrolera ?
9. Czy stos można umieścić w zewnętrznej pamięci RAM ?
10. Po co w mikrokontrolerze rejestry specjalne ?
11. Które rejestry mikrokontrolera 8051 są 16-bitowe ?
12. Czy można bezpośrednio do wyprowadzeń mikrokontrolera 8051 dołączyć rezonator kwarcowy lub ceramiczny ?

3. Od mikrokontrolera 8051 do 80515/535

Mikrokontroler 8051 doczekał się wielu następców. Pierwszym w kolejności był opracowany również przez firmę Intel procesor 8052/32. Pierwsze oznaczenie dotyczy mikrokontrolera z wewnętrzną pamięcią programu ROM, a drugie - mikrokontrolera bez wewnętrznej pamięci ROM. Liczba wyprowadzeń nie uległa zmianie. Zmieniła się częściowo struktura wewnętrzna. Największym prezentem dla programistów i użytkowników było:

- podwojenie wewnętrznej pamięci danych RAM ze 128 do 256 bajtów,
- dodanie trzeciego, wewnętrznego 16-bitowego licznika oznaczonego kolejnym symbolem T2; licznik ten znacznie rozszerzał możliwości podstawowych liczników T0 i T1.

Wprowadzenie trzeciego licznika T2 zmieniło także sposób sterowania łączem szeregowym. Szybkości nadawanych i odbieranych danych mogły być zmieniane niezależnie. Porównanie obu struktur wewnętrznych przedstawiono na rysunku 3-1.



Rys. 3-1 Struktura wewnętrzna mikrokontrolera 8052 w porównaniu z mikrokontrolerem 8051.

W przedstawionej na rysunku 3-1 strukturze wewnętrznej mikrokontrolera 8052 zacieniowanymi polami zaznaczono układy nowe lub o zmienionych funkcjach w stosunku do pełnionych w mikrokontrolerze 8051.

W kontrolerze przerwań, wskutek wprowadzenia trzeciego licznika T2, zwiększyła się liczba obsługiwanych przerwań wewnętrznych. Liczba prze-

rwań zewnętrznych również zwiększyła się o jedno. Mikrokontroler reagował na dwa przerwania zewnętrzne i jedno wspólne z licznikiem T2.

Rolę wejścia i sterowania licznika T2 pełnią dwie linie portu P1, stąd zmienione (zaciemnione pole) funkcje tego portu.

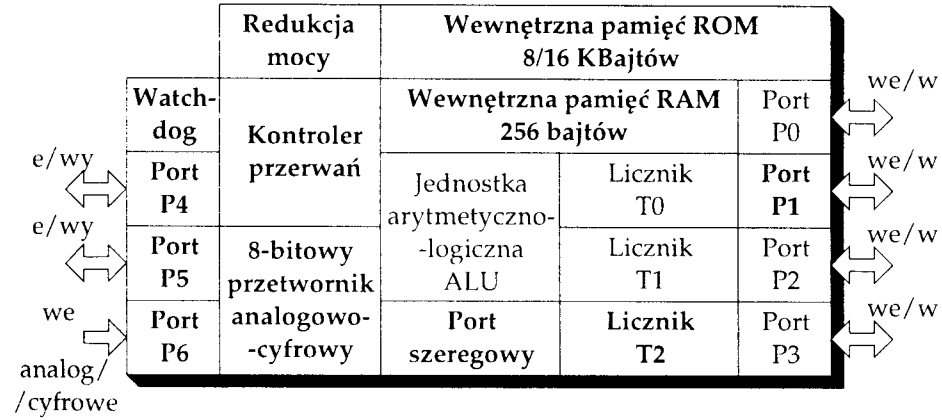
Najważniejszą cechą z punktu wiedzenia programistów jest to, że lista rozkazów nie zmieniła się. Zamierzeniem konstruktorów mikrokontrolera 8051 było wprowadzenie rejestrów specjalnych SFR, w obrębie których realizowane będą wszystkie zmiany struktury wewnętrznej mikrokontrolera. Sposób zapisu i odczytu tych rejestrów nie zmieniał się.

Europejscy konstruktorzy mikrokontrolerów 8-bitowych, przede wszystkim firma Philips i Siemens, skupili się na mikrokontrolerach z przetwornikami pomiarowymi, wielofunkcyjnymi łączami szeregowymi oraz rozbudowanymi układami czasowo-licznikowymi. Przetworniki napięcie-kod rozszerzyły zakres zastosowań mikrokontrolerów. Bezpośrednie dołączenie czujników z wyjściem napięciowym, rezystancyjnym i prądowym przestało być problemem. Zmiany w konstrukcji łącz szeregowych spowodowane zostały wprowadzeniem nowych interfejsów, przede wszystkim magistrali I^2C-Bus opatentowanej przez firmę Philips. Magistrala ta stała się wkrótce standardem w sprzęcie audio-video (RTV).Dostępne funkcje liczników T0 i T!, a nawet wprowadzonego w mikrokontrolerze 8051 licznika T2, są w wielu przypadkach niewystarczające. W szczególności przy sterowaniu np. silników elektrycznych. Wprowadzenie nowych technologii w produkcji półprzewodników spowodowało obniżenie poboru prądu, prawie 8-krotną redukcję mocy, a w niektórych przypadkach jeszcze większą. Nowe elementy wewnętrzne to także nowe źródła przerwań, które musi obsłużyć kontroler przerwań. W najnowszych konstrukcjach mikrokontrolerów kontroler przerwań obsługuje 17 różnych, wewnętrznych i zewnętrznych przerwań. Za najbardziej zaawansowany technologicznie mikrokontroler rodziny '51 uważany jest mikrokontroler 8xC517A firmy Siemens, który zawiera także szybki koprocesor arytmetyczny.

Pierwszym mikrokontrolerem 8-bitowym rodziny '51, który zawierał przetwornik analogowo-cyfrowy był mikrokontroler 80515/535. Producentem tego układu jest wspomniana wcześniej firma Siemens oraz amerykańska firma AMD (Advanced Micro Devices). Pierwotne podwójne oznaczenie 80515/535 dotyczyło wersji mikrokontrolera z wewnętrzną pamięcią programu (80515) lub mikrokontrolera bez tej pamięci (80535). Nowe oznaczenia tych mikrokontrolerów są nieco zmienione (patrz uwagi zamieszczone na końcu książki). W mikrokontrolerach 80515/535 w porównaniu z procesorem 8051 rozszerzono funkcje wielu wewnętrznych układów (rysunek 3-2):

- wewnętrzna pamięć ROM liczy:
  → 8 KB w SAB80(C)515/535,
  → 16 KB w SAB 83515-4 (mikrokontroler z pamięcią ROM programowaną maską, rozszerzoną w stosunku do 80(C)515/535),

- programowanie szybkości transmisji szeregowej może odbywać się bez pomocy licznika T1,
- podwyższono częstotliwość dołączanego do procesora rezonatora kwarcowego,
- zwiększyła się liczba wejść kontrolera przerwań; mikrokontroler reaguje na 5 przerwań wewnętrznych i 7 przerwań zewnętrznych w ramach 4-poziomowego systemu przerwań,
- rozszerzono tryby redukcji mocy pobieranej przez mikrokontroler.



Rys. 3-2. Struktura wewnętrzna mikrokontrolera 80515/535 w porównaniu z mikrokontrolerem 8051.

Zaciemnione pola na rysunku 3-2 są wynikiem zmienionych funkcji układów istniejących w mikrokontrolerze 8051 lub zupełnie nowych, charakterystycznych dla mikrokontrolera 80515/535. Nowymi wewnętrznymi układami są:

- dwa uniwersalne porty wejściowo-wyjściowe P4 i P5,
- licznik T2 z modulacją szerokości impulsu (PWM - Pulse With Modulation) i synchroniczną zmianą wartości czterech najmniej znaczących linii portu P1,
- 8-kanałowy, 8-bitowy przetwornik analogowo-cyfrowy z wewnętrznym układem próbkująco-pamiętającym i programowanymi podzakresami pomiarowymi,
- port P6 w mikrokontrolerach wykonanych w technologii ACMOS,
- Watchdog - układ nadzorujący wykonywanie programu,
- wyprowadzony na jedną z linii portu P1 wewnętrzny sygnał taktujący mikrokontroler.

Podobnie jak w mikrokontrolerze 8052 także w mikrokontrolerach 80515/535 lista rozkazów nie uległa zmianie. Wszystkie nowości dotyczą jedynie rejestrów specjalnych SFR. Ważne jest to, że wskutek zwiększenia liczby przerwań i zastosowania 4-poziomowego kontrolera przerwań zmieniły się nazwy dwóch rejestrów specjalnych.

Liczniki T0 i T1 oraz zasada ich programowania nie uległy zmianie. Licznik T2 w mikrokontrolerach 80515/535 jest zupełnie innym licznikiem w porównaniu z procesorem 8052. Zasadniczym powodem zmiany struktury wewnętrznej licznika T2 jest możliwość wprowadzenie licznika w tryb porównania, wykorzystywany do synchronicznego generowania impulsów o programowanym współczynniku wypełnienia, automatycznego wpisu wartości początkowej i zapamiętania wartości chwilowej. Aby uzyskać te właściwości z licznikiem T2 związano 11 dodatkowych rejestrów specjalnych.

Wzorem innych układów również łącze szeregowe uległo rozbudowie. Wprowadzono dodatkowy generator o stałej szybkości transmisji wynoszącej 4800 lub 9600 bodów (bitów na sekundę).

Zupełnie nowym elementem jest przetwornik analogowo-cyfrowy o rozdzielczości 8 bitów z 8-wejściowym, analogowym multiplekserem. Oznacza to możliwość pomiaru napięcia w 8 różnych kanałach. Czas przetwarzania jest krótki i wynosi 13..15 cykli maszynowych, zależnie od wykonania mikrokontrolera.

Watchdog jest także nowym układem, którego nie było w mikrokontrolerach 8051 czy też 8052. Jest to układ nadzorujący poprawne działanie mikrokontrolera jako sprzętowe zabezpieczenie programu przed zakłóceniami, które mogą spowodować niezamierzoną zmianę realizowanego programu. Głównym elementem watchdoga jest 16-bitowy licznik, który raz uruchomiony nie może zostać zatrzymany. Przepełnienie tego licznika powoduje wewnętrzne zerowanie mikrokontrolera. Aby nie dopuścić do takiej sytuacji, licznik musi być cyklicznie, programowo zerowany.

Rozbudowane zostały możliwości redukcji prądu pobieranego prze mikrokontroler. W najbardziej oszczędnym trybie redukcji mocy prąd pobierany z zasilacza o napięciu $V_{CC} = 5V$ zredukowany został do wartości mniejszej niż $I_{CC} = 50\ \mu A$.

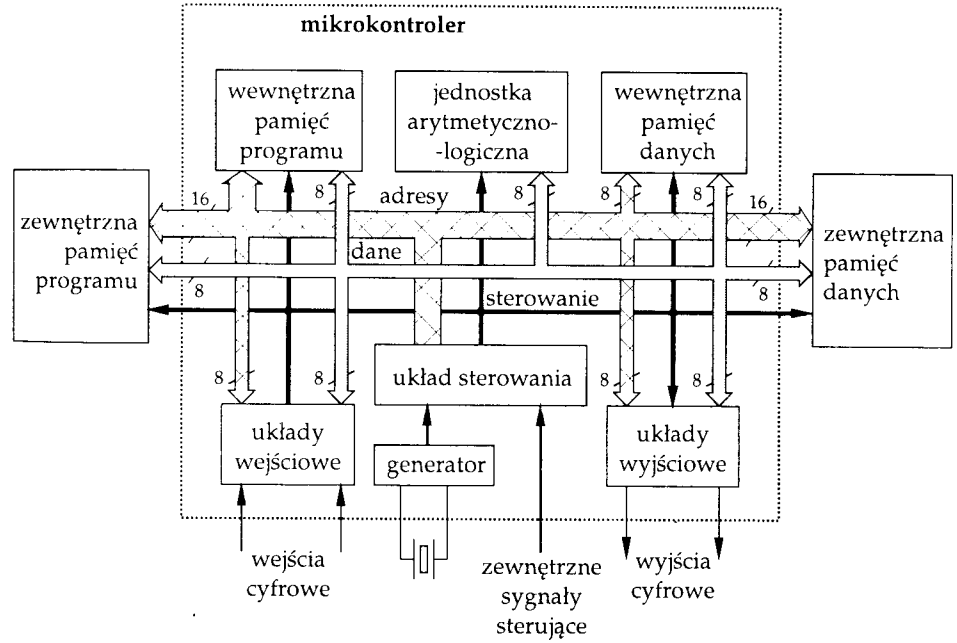
Pytania i problemy

1. Jakie zmiany wprowadzili konstruktorzy w mikrokontrolerze 8052 w stosunku do mikrokontrolera 8051 ?
2. Czy rozbudowa wewnętrznych układów, np. liczników, łącza szeregowego itp. zmienia listę rozkazów mikrokontrolera i dlaczego ?
3. Jak duża jest wewnętrzna pamięć RAM w mikrokontrolerach: 8051, 8052 i 80515/535 ?
4. Ile portów, równoległych układów wejściowo-wyjściowych, jest w mikrokontrolerze 80515/535 ?
5. Wyjaśnij sposób modulacji szerokości impulsu (metoda PWM).
6. Jakimi parametrami charakteryzuje się przetwornik analogowo-cyfrowy w mikrokontrolerze 80515/535 ?
7. Jak definiowana jest szybkość przesyłu danych łączem szeregowym ?
8. Co to jest watchdog i do czego służy ?
9. Narysuj samodzielnie strukturę wewnętrzną mikrokontrolera 8051, 8052 i 80515/535. Wyjaśnij przeznaczenie poszczególnych bloków.

4. Jak odczytywać i zapisywać dane z pamięci programu ROM i zewnętrznej pamięci danych RAM

Jeśli mikroprocesor jest układem o dużym stopniu scalenia (dużej skali integracji), którego działanie zależy od programu, ciągu rozkazów, to w jaki sposób przygotować program ? Jakie rozkazy można zastosować ? Co to jest rozkaz ? Gdzie przechowywane są dane i jak sterować przepływem danych ?

Przed odpowiedzią na postawione pytania należy przypomnieć sobie co jest w środku mikrokontrolera. Pomocny będzie rysunek z rozdziału 2 przedstawiony jeszcze raz poniżej.



Rys. 4-1 Schemat blokowy mikrokontrolerów rodziny 8051.

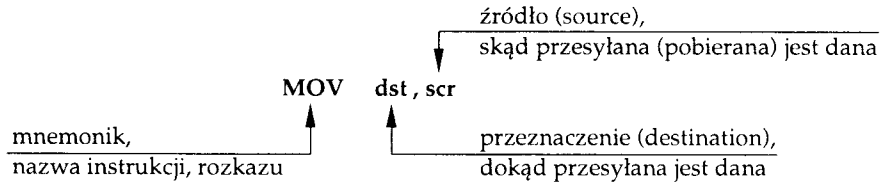
Program działania czyli sposób działania mikrokontrolera zapisany jest w pamięci programu ROM, wewnętrznej lub zewnętrznej. Rodzaj pamięci nie jest istotny. Pewne stałe współczynniki, wartości znane w czasie pisania programu i takie które nie zmienią się w trakcie wykonywania programu mogą być również wpisane wraz z programem do pamięci ROM. Ale wyniki obliczeń czy też pomiarów mogą być wpisywane tylko do pamięci RAM. Są to wartości zmieniające się w trakcie wykonywania programu, wartości, które nie są znane przed uruchomieniem, wykonaniem programu. Jeśli mikrokontroler ma

zmierzyć temperaturę, to dopóki nie zostanie wykonany pomiar, dopóty nie wiadomo jaka jest jej wartość. Należy pamiętać, że:

- pamięć ROM jest pamięcią tylko do odczytu, mikrokontroler nie może do niej nic wpisać,
- pamięć RAM jest przystosowana do zapisu i odczytu, mikrokontroler może odczytywać i zapisywać dane.

Realizowany przez mikrokontroler program jest ciągiem poleceń, instrukcji, rozkazów, które są zrozumiałe przez mikrokontroler. Ponieważ konstruktorem mikrokontrolera 8051 jest amerykańska firma Intel, dlatego nazwy rozkazów pochodzą z języka angielskiego, podobnie jak wszystkie inne pojęcia wiążące się z mikrokontrolerem. W instrukcjach nie są stosowane pełne nazwy poleceń, a jedynie ich skróty, zwane mnemonikami. W zależności od typu instrukcji w skład instrukcji poza mnemonikiem wchodzą argumenty (od jednego do trzech).

Najpowszechniejszym rozkazem są rozkazy, instrukcje przesłań, których składnia jest następująca:



Skrót instrukcji MOV pochodzi od słowa move czyli przesunięcia, ruszenia czegoś z miejsca. W rozkazie tym użyto tylko dwóch argumentów. Pierwszy (dst) do określenia komórki pamięci lub rejestru specjalnego SFR, do których ma być przesunięta dana. Drugi (scr) także do określenia adresu komórki pamięci lub rejestru specjalnego, z których ma być pobrana dana. Kolejność argumentów jest bardzo ważna. W instrukcjach 2-argumentowych jako pierwszy występuje zawsze adres docelowy, a jako drugi adres źródłowy. W innych instrukcjach liczba argumentów zmienia się od jednego do trzech, w zależności od realizowanej funkcji.

Patrząc na rysunek 4-1 nietrudno zauważyć, że do adresowania wewnętrznej pamięci RAM wystarcza 8 bitów, czyli jeden bajt, ponieważ wewnętrzna magistrala adresowa jest 8-bitowa.

Dlatego przesłanie danych między dwiema komórkami wewnętrznej pamięci RAM można wykonać poprzez następujący rozkaz:

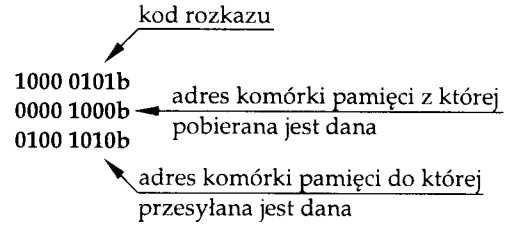
```
MOV     4Ah , 8
```


Wykonując ten rozkaz, mikrokontroler pobiera zawartość komórki o adresie 8 i przesyła ją do komórki o adresie 4Ah. Adresy argumentów podane zostały w różnych kodach. Pierwszy (4Ah) w kodzie szesnastkowym, a drugi (8) w kodzie dziesiętnym. Adresy można także podawać w kodzie binarnym i ósemkowym, np:

4Ah = 74 = 0100 1010b = 112o

Przedstawiony sposób adresowania obu komórek pamięci RAM nazywany jest adresowaniem bezpośrednim, ponieważ oba adresy są częścią rozkazu i w sposób jawny (bezpośredni) wskazują na miejsce w pamięci RAM.

To co jest zrozumiałe dla człowieka wcale nie musi być zrozumiałe dla mikroprocesora. Program zawierający mnemoniki wykonywanych instrukcji jest czytelny dla człowieka. Jeśli te same instrukcje mają być zrozumiałe dla mikrokontrolera, to muszą zostać przetłumaczone na ciąg zero-jedynkowy. Proces tłumaczenia i kodowania nazw instrukcji i ich argumentów nazywany jest asemblacją. Wynika z tego (w dużym skrócie), że program będący ciągiem poleceń tłumaczony jest na kod wynikowy za pośrednictwem programu zwanego asemblerem. Przytoczona powyżej instrukcja MOV 4Ah,8 zamieniona zostaje na trzy bajty w postaci:



W celu łatwiejszej interpretacji z ciągu zero-jedynkowego wyodrębnione zostały 3 bajty, a w każdym bajcie rozróżniono 4 mniej i 4 bardziej znaczące bity. Taka postać rozkazów jest przyjmowana i wykonywana przez mikrokontroler. Nie jest ona za bardzo czytelna dla człowieka. Jeśli program liczy kilkaset lub kilka tysięcy takich ciągów zero-jedynkowych to nietrudno o pomyłkę. Dlatego na przyszłość należy pozostać przy mnemonikach instrukcji.

Pamięć programu ROM oraz zewnętrzna pamięć danych RAM adresowane są za pośrednictwem 16-bitowej magistrali adresowej. Dlatego do ich adresowania należy użyć 16-bitowego rejestru. Takim rejestrem jest licznik rozkazów PC oraz wskaźnikowy rejestr danych DPTR. W przypadku zewnętrznej pamięci RAM możliwy jest jeszcze jeden sposób adresowania poprzez połączenie 8-bitowego rejestru R0 lub R1 i portu P2. Jednym z argumentów w tych rozkazach jest akumulator A, do którego przesyłane są lub z którego pobierane są dane (rysunek 4-2).

4.1 Odczyt danych z pamięci programu ROM

Wewnętrzna lub zewnętrzna pamięć programu zawiera oprócz kodu programu także stałe wykorzystywane w programie. Ze względu na swój charakter, pamięć ROM (EPROM, EEPROM, Flash ROM), możliwy jest jedynie odczyt danych.

Wyboru typu pamięci, wewnętrznej lub zewnętrznej, mikrokontroler dokonuje testując linię $\overline{EA}$ (External Access enable) w trakcie zerowania:

- $\overline{EA} = 0$, zewnętrzna pamięć programu,
- $\overline{EA} = 1$, wewnętrzna pamięć programu

Stan linii $\overline{EA}$ nie ma znaczenia jeśli licznik rozkazów PC zawiera adres większy niż adres wewnętrznej pamięci programu. Przykładowo jeśli mikrokontroler 8051 zawiera 4KB-ową wewnętrzną pamięć programu to dla stanu licznika rozkazów PC:

- PC = 0000h .. 0FFFh i $\overline{EA} = 0$ – mikrokontroler wykonuje program z zewnętrznej pamięci programu,
- PC = 0000h .. 0FFFh i $\overline{EA} = 1$ – mikrokontroler wykonuje program z wewnętrznej pamięci programu,
- PC = 1000h .. 0FFFFh – mikrokontroler wykonuje program z zewnętrznej pamięci programu, niezależnie od stanu linii $\overline{EA}$

Pamięć program (wewnętrzna lub zewnętrzna) adresowana jest na trzy różne sposoby:

1a. przez drugi lub przez trzeci bajt wykonywanej instrukcji. Sytuacja taka występuje przy pobieraniu 8-bitowych stałych z pamięci programu, np. przy inicjalizacji wartości początkowych rejestrów, komórek pamięci, np:

instrukcje:	zapis binarny instrukcji:	wykonywane operacje
MOV A,#3Ch	0111 0100b 0011 1100b	A ← 3Ch
MOV R0,#6	0111 1000b 0000 0110b	R0 ← 6
MOV 35,#4Bh	0111 0101b 0010 0011b 0100 1011b	(35) ← 4Bh

Dwie pierwsze instrukcje są instrukcjami 2-bajtowymi. Pierwszy bajt zawiera kod wykonywanej instrukcji i informację, adres docelowego rejestru. Drugi bajt zawiera wartość stałej, która wpisywana jest do akumulatora A (MOV A,#3Ch) i rejestru R0 (MOV R0,#6).

W trzeciej instrukcji (MOV 35,#4Bh) pierwszy bajt zawiera kod wykonywanej instrukcji, drugi adres komórki wewnętrznej pamięci RAM (35) zamienianej na liczbę szesnastkową (23h), a trzeci wartość stałej pobieranej z pamięci programu (4Bh).

W powyższej tabeli przedstawiono również zapis binarny wykonywanych instrukcji (zrozumiały dla mikrokontrolera) oraz wykonywane operacje. Strzałka ← oznacza kierunek przepływu danych, a wartość ujęta w okrągłe nawiasy (35) jest adresem komórki pamięci, w tym przypadku wewnętrznej pamięci RAM. Należy zwrócić uwagę na sposób zapisu stałych, które poprzedzone są zawsze znakiem #. Pominięcie tego znaku powoduje, że odwołujemy się do komórki pamięci RAM lub rejestrów specjalnych SFR. Ten sposób adresowania nazywany jest **adresowaniem natychmiastowym**.

1b. przez drugi i trzeci bajt wykonywanej instrukcji. Sytuacja taka występuje przy inicjalizacji wartości początkowej 16-bitowego wskaźnikowego rejestru danych DPTR:

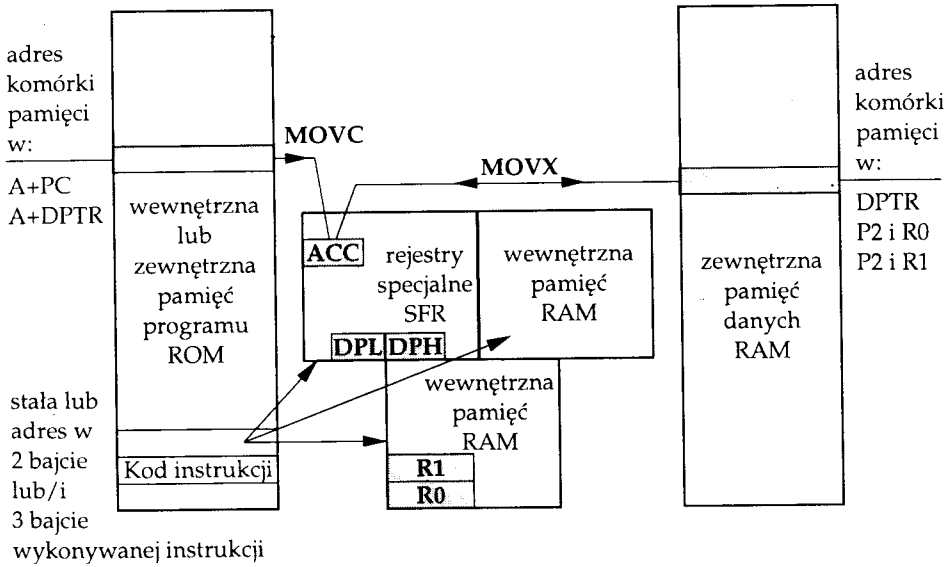
instrukcja:	zapis binarny instrukcji:	wykonywana operacja
MOV DPTR,#2E59h	1001 0000b 0010 1110b 0101 1001b	DPTR ← 2E59h

Przedstawiona 3-bajtowa instrukcja jest jedyną instrukcją, w której wpisywana jest do rejestru DPTR stała 16-bitowa. Pierwszy bajt zawiera kod operacji, bajt drugi i trzeci wpisywaną stałą. Ten sposób adresowania także nazywany jest **adresowaniem natychmiastowym**.

2. sumą zawartości 8-bitowego akumulatora A i 16-bitowego wskaźnikowego rejestru danych DPTR. Liczba zawarta w akumulatorze A traktowana jest jako liczba całkowita bez znaku, co oznacza, że jej zakres zmienności ograniczony jest do przedziału 0 .. 255.

instrukcja:	zapis binarny instrukcji:	wykonywana operacja
MOVC A,@A+DPTR	1001 0011b	A ← (A+DPTR)$_{CODE}$

Aby wykonać dodawanie zawartości 8-bitowego akumulatora A i 16-bitowego wskaźnikowego rejestru danych DPTR, zawartość akumulatora A rozszerzana jest znakowo z 8 bitów do 16 bitów o bajt równy 00. Adresowanie pamięci programu określane jest jako **adresowanie za pomocą rejestrów bazowych** (w tym przypadku za pośrednictwem wskaźnikowego rejestru danych DPTR) lub **indeksowo-rejestrowo pośrednie**.



Rys. 4-2 Pobieranie danych z pamięci programu i wymiana danych z/do zewnętrznej pamięci RAM.

3. sumą zawartości 8-bitowego akumulatora A i 16-bitowego licznika rozkazów PC. Podobnie jak poprzednio liczba zawarta w akumulatorze A traktowana jest jako liczba całkowita bez znaku:

instrukcja:	zapis binarny instrukcji:	wykonywana operacja
MOVC A,@A+PC	1000 0011b	A ← (A+PC)$_{CODE}$

Zawartość akumulatora A jest rozszerzana znakowo z 1 do 2 bajtów. Różnica między tym, a poprzednim rozkazem polega na tym, że po pobraniu kodu instrukcji licznik rozkazów PC zwiększany jest o 1. W ten sposób licznik rozkazów PC wskazuje na pierwszy bajt następnej instrukcji znajdującej się po wykonywanym rozkazie MOVC A,@A+PC. W podanej instrukcji pamięć programu **adresowana jest za pomocą rejestru bazowego** jakim jest licznik rozkazów PC (**adresowanie indeksowo-rejestrowo pośrednie**).

W dwóch ostatnich rozkazach wskaźnikowy rejestr danych DPTR i bieżąca wartość licznika rozkazów PC traktowane są jako wskaźniki (adresy początkowe) do tablic umieszczonych w pamięci programu ROM. Zawartość akumulatora A traktowana jest jako indeks tablicy. Wielkość tych tablic ograniczona jest do 256 bajtów, ze względu na 8-bitową zawartość akumulatora A. Oba rozkazy stosowane są również bardzo często do pobierania z pamięci programu kolejnych znaków komunikatów, np: wyświetlanych na polu odczytowym modułu dydaktycznego, przyrządu pomiarowego, sterowanego urządzenia itp.

4.2 Odczyt/zapis danych do zewnętrznej pamięci RAM

Zewnętrzna pamięć danych RAM, jeśli istnieje potrzeba jej dołączenia do mikrokontrolera wskutek zbyt małej wewnętrznej pamięci RAM, adresowana jest w dwojaki sposób:

1. wskaźnikowym rejestrem danych DPTR:

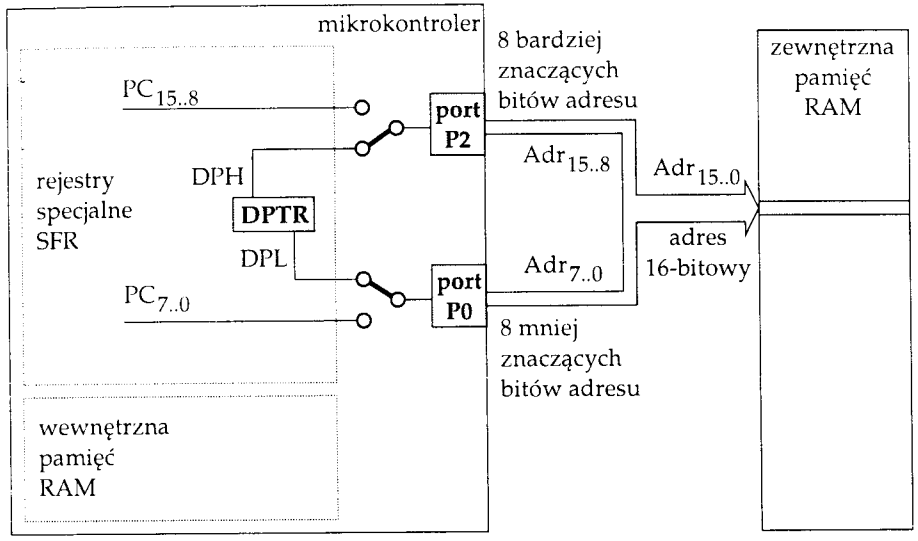
instrukcja:	zapis binarny instrukcji:	wykonywana operacja
MOVX @DPTR,A	1111 0000b	(DPTR)$_{XDATA}$ ← A
MOVX A,@DPTR	1110 0000b	A ← (DPTR)$_{XDATA}$

Jest to standardowy sposób odwoływania się do pamięci. 16-bitowy wskaźnikowy rejestr danych DPTR zawiera pełny 16-bitowy adres komórek pamięci. Ponieważ rejestr ten tworzą dwa połączone rejestry:

- DPH jako 8-bitowa, bardziej znacząca część rejestru DPTR,
- DPL jako 8-bitowa, mniej znacząca część rejestru DPTR

W trakcie adresowania komórek pamięci stan obu rejestrów pojawia się na liniach portu P0 (stan rejestru DPL) i P2 (stan rejestru DPH), tak jak przedstawiono na rysunku 4-3. Multipleksowanie stanu obu

portów realizowane jest dynamicznie, tylko na czas wykonywanej instrukcji.



Rys. 4-3 Adresowanie zewnętrznej pamięci RAM za pośrednictwem wskaźnikowego rejestru danych DPTR.

2. zawartością portu P2 i rejestru R0 lub R1:

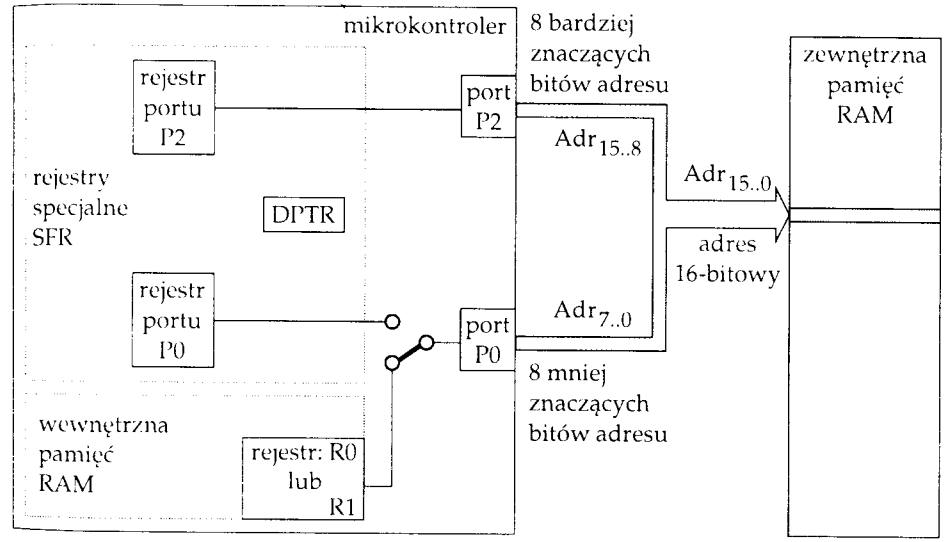
instrukcja:	zapis binarny instrukcji:	wykonywana operacja
MOVX @R0,A	1111 0010b	$(256*P2+R0)_{XDATA} \leftarrow A$
MOVX A,@R0	1110 0010b	$A \leftarrow (256*P2+R0)_{XDATA}$
MOVX @R1,A	1111 0011b	$(256*P2+R1)_{XDATA} \leftarrow A$
MOVX A,@R1	1110 0011b	$A \leftarrow (256*P2+R1)_{XDATA}$

Składnia instrukcji jest trochę myląca. Wynika z niej bezpośrednio, że do adresowania zewnętrznej pamięci RAM używany jest tylko jeden z rejestrów mikrokontrolera, R0 lub R1. Oba rejestry są rejestrami 8-bitowymi, a do zaadresowania pamięci wymagany jest adres 16-bitowy. W takcie wykonywania jednej z czterech przedstawionych instrukcji na liniach portu P0 pojawia się zawartość rejestru R0 lub R1 (8 mniej znaczących bitów adresu) ale stan portu P2 nie ulega zmianie (rysunek 4-4). Oznacza to, że przed wykonaniem instrukcji należy wpisać do portu P2 8-bardziej znaczących bitów adresowanej komórki zewnętrznej pamięci RAM.

Przykładowo do zapisu akumulatora A do komórki pamięci o adresie 3C58h należy wykonać poniższe instrukcje:

```
MOV   R0,#58h      ;R0 ← 58h
MOV   P2,#3Ch      ;P2 ← 3Ch
MOVX  @R0,A        ;(256*P2+R0) ← A
```

W każdej linii tego krótkiego programu pojawił się po znaku średnika (;) komentarz, który ma wyjaśnić, pomóc w zrozumieniu działania programu. W trakcie asemblacji wszystkie komentarze (od znaku średnika do końca wiersza) są pomijane.



Rys. 4-4 Adresowanie zewnętrznej pamięci RAM za pośrednictwem rejestru R0 lub R1 i portu P2.

Oba przedstawione sposoby adresowania zewnętrznej pamięci RAM określane są mianem **adresowania pośredniego**. We wszystkich trybach takiego adresowania, za pośrednictwem jakiegoś rejestru, przed nazwą rejestru pojawia się znak @. Znak ten świadczy, że adres komórki pamięci podany jest w bezpośrednio po znaku występującym rejestrze, np:

- w instrukcji MOVX A,@DPTR adres komórki podany jest w rejestrze DPTR
- w rozkazie MOVX @R1,A rejestr R1 zawiera część adresu komórki zewnętrznej pamięci RAM, 8-mniej znaczących bitów adresu. Pozostałą część adresu, 8-bardziej znaczących bitów, zawiera port P2.

Odczytywanie danych zapisanych w pamięci programu ROM oraz zapisywanie i odczytywanie danych z zewnętrznej pamięci danych RAM wiąże się z pewnymi ograniczeniami:

- zawsze jednym z argumentów wykonywanych instrukcji jest akumulator A, co powoduje, że akumulator A staje się jednym z najbardziej zapracowanych rejestrów mikrokontrolera,
- do odczytu danych z pamięci programu ROM mamy tylko 2 instrukcje; w ich mnemoniku występuje zawsze litera C (MOVC), będąca pierwszą literą symbolu Code oznaczającego pamięć programu (Code memory),
- do zapisu i odczytu danych do/z zewnętrznej pamięci danych RAM dostępne są po dwie instrukcje, których mnemoniki zawierają zawsze literę X (MOVX); x pochodzi od symbolicznej nazwy tej pamięci XData (eXternal Data memory),
- nie należy zapominać, że przy adresowaniu zewnętrznej pamięci danych RAM za pośrednictwem rejestru R0 lub R1 brakującą część adresu należy wpisać do portu P2 (8 bardziej znaczących bitów adresu).

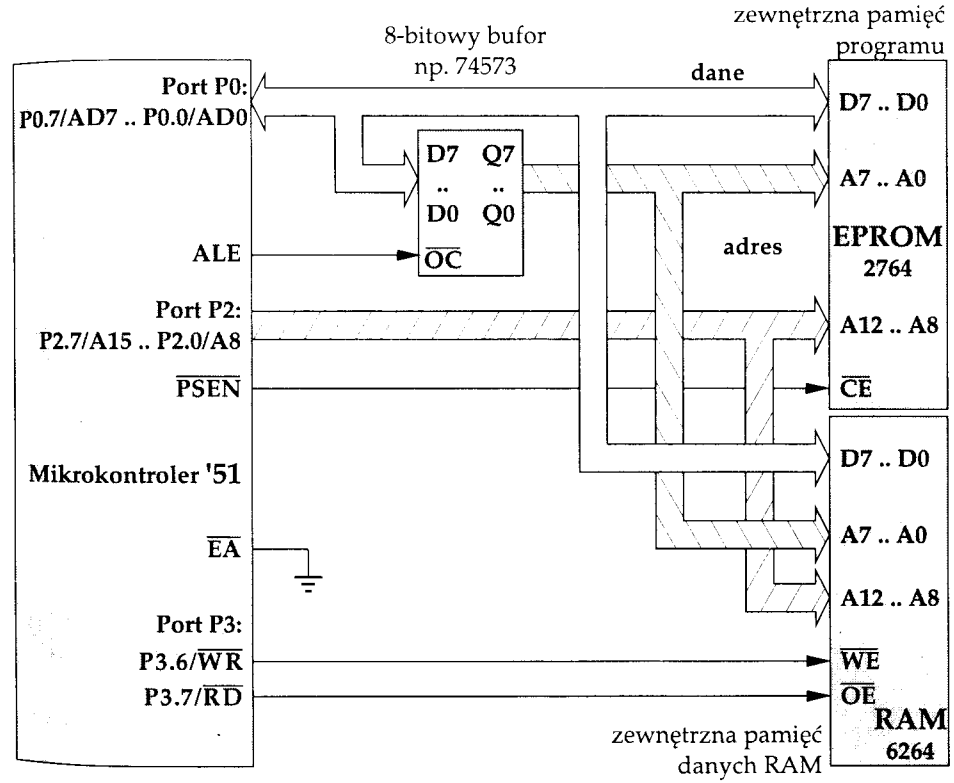
Dążąc do uzyskania dużej szybkości działania mikrokontrolerów rodziny '51 należy unikać przechowywania często używanych danych w zewnętrznej pamięci RAM mikrokontrolera. Do tego powinna wystarczyć wewnętrzna pamięć RAM.

4.3 Dołączanie zewnętrznych pamięci do mikrokontrolera

Jeśli mikrokontroler rodziny '51 ma współpracować z zewnętrznymi pamięciami, np. kodu programu (ROM, EPROM lub EEPROM, np. 2764) i danych (RAM, np. 6264), to oprócz samych pamięci konieczny jest jeszcze 8-bitowy bufor (np. 74HCT573). Zadaniem tego bufora jest zapamiętanie 8-mniej znaczących bitów adresu, co jest wynikiem multipleksowania magistrali danych i magistrali adresowej (ośmiu mniej znaczących linii tej magistrali). Rolę tej magistrali pełni port P0. Pozostałe bity adresu, 8-bardziej znaczących bitów, przesyłane są za pośrednictwem portu P2. Bardziej szczegółowo przedstawiono ten problem w rozdziale 5 i 7.

Sposób dołączenia zewnętrznych pamięci programu i danych do mikrokontrolera przedstawiono na rysunku 4-5. Mikrokontroler generuje następujące sygnały sterujące przepływem adresów lub danych w zależności od wykonywanych instrukcji:

- ALE (Address Latch Enable) - przepisanie 8-mniej znaczących bitów adresu z portu P0 do pomocniczego 8-bitowego bufora,
- $\overline{\text{PSEN}}$ (Program Store ENable) - uaktywnienie zewnętrznej pamięci programu, pobranie kodu instrukcji,
- $\overline{\text{RD}}$ (ReaD) - uaktywnienie zewnętrznej pamięci danych RAM, odczyt danej z pamięci,
- $\overline{\text{WR}}$ (WRite) - uaktywnienie zewnętrznej pamięci danych RAM, wpis danej do pamięci.



Rys. 4-5 Połączenie mikrokontrolera z zewnętrznymi pamięciami.

Przy zewnętrznej pamięci programu linia $\overline{\text{EA}}$ musi mieć poziom zera logicznego, aby po sprzętowym zerowaniu procesora, kody rozkazów były pobierane z pamięci zewnętrznej, a nie wewnętrznej.

Pytania i problemy

1. Jaki cel pełni linia $\overline{EA}$?
2. Czy zmiana stanu linii $\overline{EA}$ w trakcie wykonywania programu zmienia sposób pobierania przez mikrokontroler kodów instrukcji ?
3. Do czego służy pamięć programu, a do czego pamięć danych ?
4. Ilo bitowa jest magistrala adresowa adresująca pamięć programu, wewnętrzną i zewnętrzną pamięć danych RAM ?
5. Z jaki elementów składają się instrukcje przesłań ? Jak podawane są argumenty ?
6. Co to jest asembler i do czego służy ?
7. W jaki sposób można odczytać stałe , dane zapisane w pamięci programu mikrokontrolera ?
8. Jak obliczany jest adres pamięci programu w rozkazach:

```
MOVC    A,@A+DPTR
MOVC    A,@A+PC
```

7. W jaki sposób adresowane są komórki zewnętrznej pamięci danych RAM ?
8. Czy wykonanie podanych poniżej dwóch programów prowadzi do odczytu tej samej komórki zewnętrznej pamięci danych RAM ?

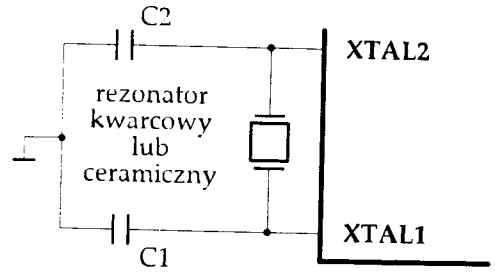
a)
```
MOV     DPTR,#4A06h
MOVX    A,@DPTR
```

b)
```
MOV     R1,#4Ah
MOV     P2,#6
MOVX    A,@R1
```

9. Dlaczego przy odczycie danych z pamięci programu ROM w mnemoniku instrukcji znajduje się litera C, a przy dostępie do zewnętrznej pamięci danych RAM litera X ?
9. Na czym polega adresowanie bezpośrednie, natychmiastowe, indeksowo-rejestrowo pośrednie i pośrednie ? Podaj przykłady instrukcji.

5. Działanie mikrokontrolera

Mikroprocesory są układami sekwencyjnymi, synchronicznymi, tzn. wszystkie operacje wykonywane przez układy procesora odbywają się w określonej kolejności i w ściśle ustalonych momentach czasowych. Dlatego muszą współpracować z generatorami impulsów zegarowych. Układy mikroprocesorowe korzystają najczęściej z generatorów zewnętrznych. Natomiast mikrokontrolery mają wewnętrzny generator, do którego dołącza się z zewnątrz albo tylko sam rezonator kwarcowy lub ceramiczny oraz dodatkowo dwa kondensatory, ułatwiające wzbudzenie się generatora na pożądanej częstotliwości, rysunek 5-1. W mikrokontrolerach rodziny '51 jeżeli jest stosowany rezonator kwarcowy, to pojemności mają wartości 22 ÷ 39 pF, a przy rezonatorze ceramicznym - 30 ÷ 51 pF.



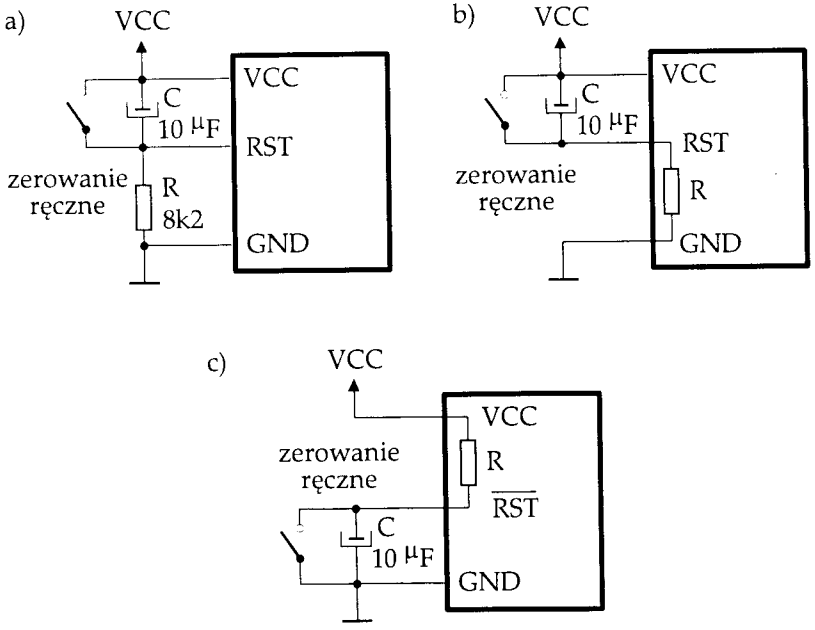
Rys. 5-1 Podłączenie rezonatora kwarcowego lub ceramicznego do mikrokontrolera.

Dobierając częstotliwość generatora należy pamiętać, że musi się ona mieścić w zakresie określonym przez producenta. Przy częstotliwości za dużej lub za małej mikrokontroler przestanie poprawnie pracować. Stosowanie wyższych częstotliwości generatora przyspiesza wykonywanie przez mikrokontroler operacji, ale odbywa się to kosztem zwiększenia prądu zasilania, zwiększenia poziomu zakłóceń wytwarzanych przez procesor, a ponadto wymaga stosowania szybszych, a więc i droższych elementów współpracujących z mikrokontrolerem, np. pamięci zewnętrznych. Ponadto przy wyższych częstotliwościach generatora mikrokontroler, zwłaszcza pobierający rozkazy z pamięci zewnętrznej programu, jest bardziej podatny na zakłócenia zewnętrznych pól elektromagnetycznych. Dlatego częstotliwość generatora powinna być najniższa ale zapewniająca wykonanie zadania w założonym czasie.

W większości mikrokontrolerów praca generatora może być zatrzymana odpowiednim rozkazem, co powoduje zatrzymanie działania wszystkich elementów mikrokontrolera. Ma to na celu zredukowanie do minimum prądu zasilania, np. w momencie awarii sieci zasilającej.

Układ każdego generatora wymaga odpowiedniego czasu od momentu włączenia zasilania do momentu osiągnięcia założonej częstotliwości. Czas ten wynosi na ogół około 10 ms.

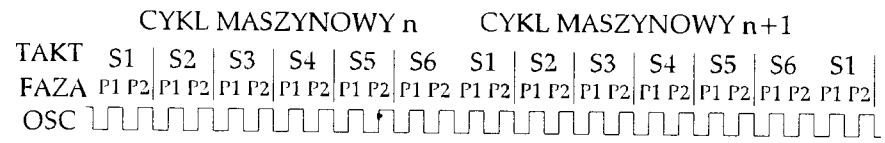
Ponieważ po włączeniu zasilania układy wewnętrzne mikrokontrolera przyjmują stan dowolny, dlatego musi istnieć możliwość wymuszenia na mikroprocesorze powtarzalnych stanów początkowych wszystkich jego rejestrów. W komputerach służy do tego przycisk RESET. Mikroprocesory posiadają wejście (RST), na które jeżeli poda się odpowiedni sygnał, to nastąpi wymuszenie stanów początkowych rejestrów wewnętrznych. Ponieważ stan początkowy procesora musi być ustalony po włączeniu zasilania, to do tego wejścia dołącza się układ czasowy RC, rysunek 5-2 a, który wymusza, na wymagany odcinek czasu, sygnał zerujący. W większości mikrokontrolerów, np. w mikrokontrolerach rodziny '51 wykonanych w technologii CMOS rezystor jest umieszczony wewnątrz układu, rysunek 5-2 b Sygnałem tym, w zależności od typu mikrokontrolera może być poziom zera logicznego (masa układu), jak na przykład w mikrokontrolerach 80C515, rysunek 5-2 c lub poziom jedynki logicznej (zasilanie) jak w mikrokontrolerach 8051, rysunek 5-2 a i b.



Rys. 5-2 Układy zerujące mikrokontrolery.

W większości układów mikroprocesorowych stosuje się przełączniki, dołączone do linii RST, do ręcznego zerowania procesorów. Daje to możliwość uruchomienia od początku programu procesora, np. w przypadku testowania lub zawieszenia się programu procesora.

Podstawową jednostką określającą czas wykonywania instrukcji jest cykl maszynowy. Dla rodziny mikrokontrolerów '51 składa się on z sześciu stanów, rysunek 5-3, oznaczonych od S1 do S6, z których każdy dzieli się na dwie fazy P1 i P2. Czas trwania jednej fazy jest równy okresowi oscylatora. Wynika stąd, że czas trwania cyklu maszynowego jest 12 razy dłuższy od okresu oscylatora.



Rys. 5-3 Cykle maszynowe mikrokontrolerów.

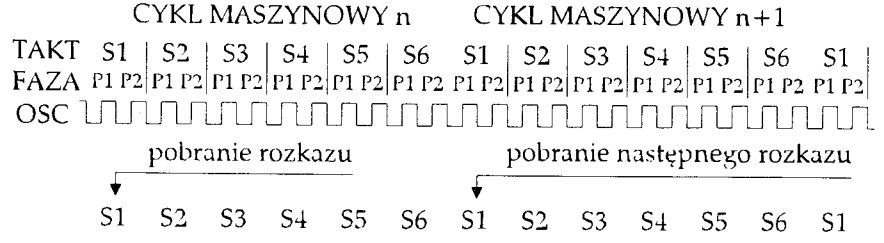
W każdym stanie cyklu maszynowego są realizowane pewne podstawowe procedury związane z wykonywaniem rozkazów.

Rozkazy są pobierane z pamięci wewnętrznej lub zewnętrznej mikrokontrolera spod adresu wskazywanego przez licznik rozkazów (PC).

Rozkazy mogą być jedno lub wielobajtowe, a czas ich wykonania może trwać jeden lub kilka cykli maszynowych. W rodzinie '51 występują rozkazy jedno, dwu i trójbajtowe, które są wykonywane w jednym, dwóch lub czterech cyklach maszynowych.

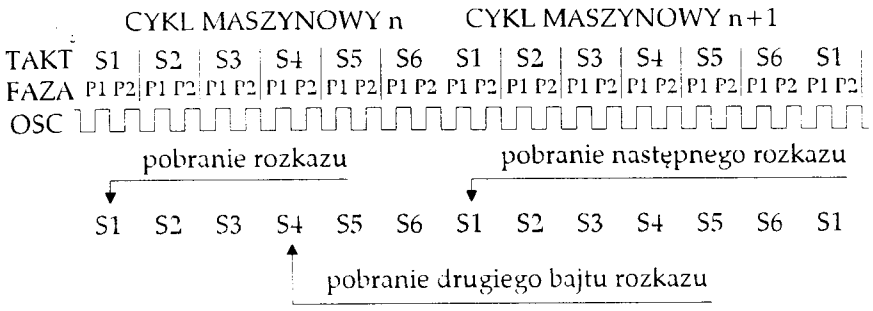
Pobranie pierwszego bajtu rozkazu odbywa się zawsze w takcie S1.

Jeżeli rozkaz jest jednobajtowy, to jest on wykonywany w jednym cyklu maszynowym, a bajt następnego rozkazu jest pobierany w takcie S1 kolejnego cyklu maszynowego, rysunek 5-4.



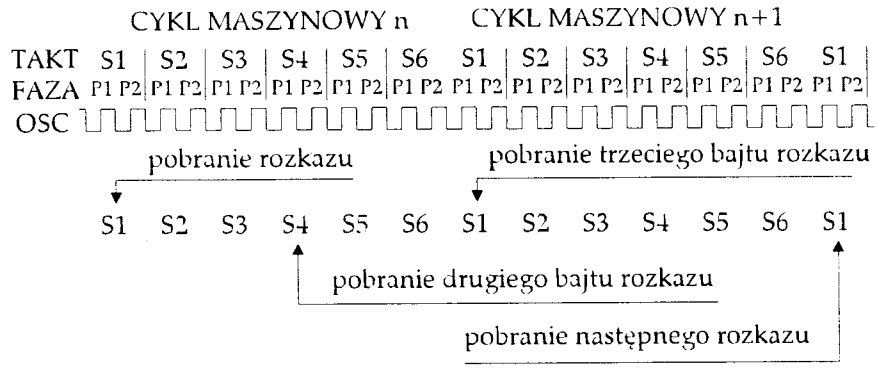
Rys. 5-4 Wykonanie rozkazu jednobajtowego.

Większość rozkazów dwubajtowych jest wykonywana w jednym cyklu maszynowym, z tym że drugi bajt rozkazu jest pobierany w takcie S4 cyklu maszynowego, rysunek 5-5.



Rys. 5-5 Wykonanie rozkazu dwubajtowego.

Rozkazy trójbajtowe są wykonywane w dwóch cyklach maszynowych. W jednym cyklu maszynowym, w taktach S1 i S4, są pobierane dwa bajty rozkazu, a w takcie S1 drugiego cyklu maszynowego jest pobierany trzeci bajt rozkazu. Bajt nowego rozkazu jest pobierany w takcie S1 trzeciego cyklu maszynowego, rysunek 5-6.



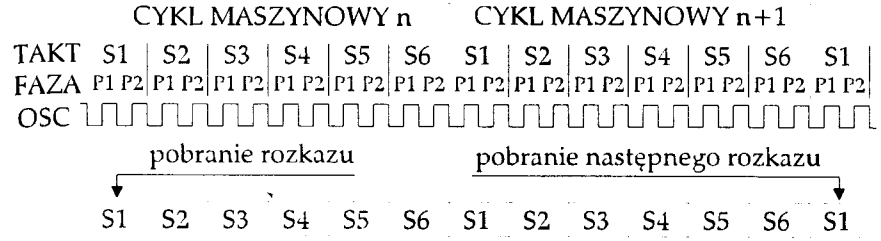
Rys. 5-6 Wykonanie rozkazu trójbajtowego.

W rodzinie '51 występują również rozkazy jednobajtowe i dwubajtowe, które są wykonywane w dwóch lub czterech cyklach maszynowych. W czterech cyklach maszynowych są wykonywane tylko rozkazy mnożenia (MUL AB) i dzielenia (DIV AB).

Na rysunku 5-7 jest podany przykład rozkazu jednobajtowego wykonywanego w dwóch cyklach maszynowych.

Po pobraniu każdego bajtu rozkazu następuje automatyczne zwiększenie o jeden licznika rozkazów.

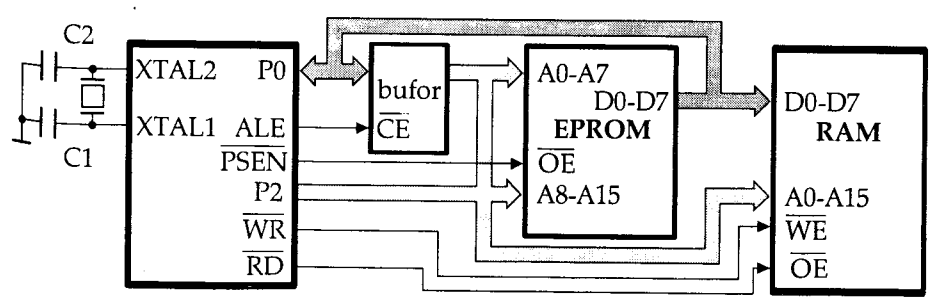
Znajomość liczby cykli maszynowych potrzebnych na wykonanie poszczególnych rozkazów jest konieczna w przypadku generowania przez mikrokontroler odcinków czasów o dużej dokładności. Sygnały o krótkich czasach uzyskuje się najczęściej przez wykonanie programu wymagającego odpowiedniej liczby cykli maszynowych.



Rys. 5-7 Rozkaz jednobajtowy wykonywany w dwóch cyklach maszynowych.

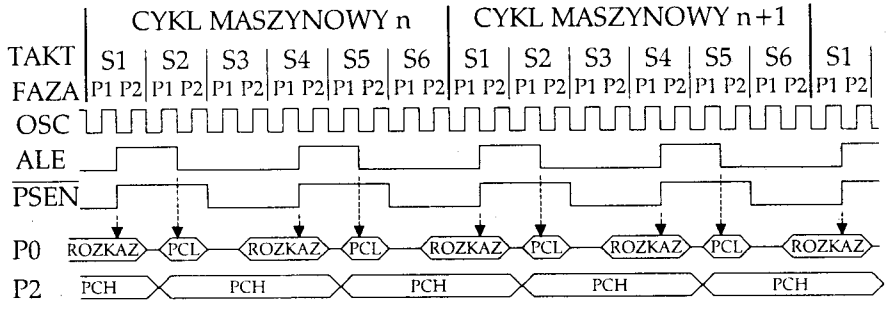
Jeżeli mikrokontroler pobiera rozkazy z pamięci zewnętrznej, to musi istnieć możliwość wysłania na zewnątrz adresu komórki pamięci, z której jest pobierany rozkaz oraz wejście poprzez które rozkaz jest przesyłany do struktury wewnętrznej mikrokontrolera. W mikrokontrolerach rodziny '51 można również dołączyć zewnętrzną pamięć RAM służącą do chwilowego przechowywania danych. Do dostępu zarówno do pamięci programu jak i danych wykorzystuje się dwa ośmiobitowe porty P0 i P2. Żeby zmniejszyć liczbę wyprowadzeń obsługujących pamięci zewnętrzne przez port P0 jest wysyłany mniej znaczący bajt 16-bitowego adresu oraz są przesyłane bajty rozkazów oraz dane do lub z pamięci RAM. Dla rozdzielenia bajtu adresu od rozkazów i danych jest stosowany zatrzask, w którym jest zapamiętywany adres. Rozdzielenie dostępu do pamięci danych i programu odbywa się sygnałami $\overline{PSEN}$ i $\overline{RD}$,rysunek 5-8.

$\overline{PSEN}$ odblokowuje pamięć programu na czas pobierania z niej rozkazów,
$\overline{WR}$ wpisuje dane do zewnętrznej pamięci RAM,
$\overline{RD}$ odczytuje dane z pamięci zewnętrznej RAM



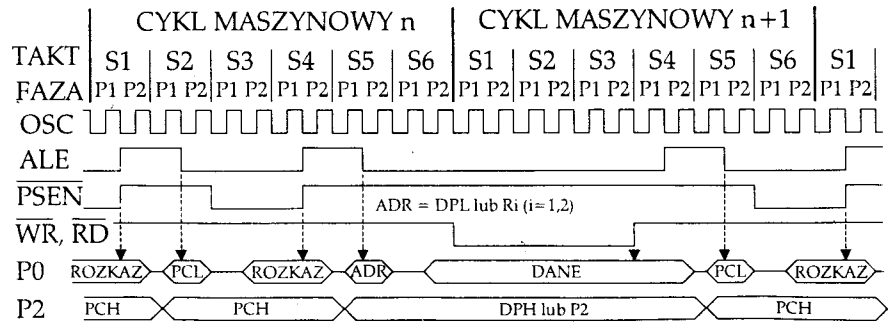
Rys. 5-8 Połączenie mikrokontrolera z pamięciami zewnętrznymi.

Mniej znaczący bajt adresu jest wpisywany do zatrzasku opadającym zboczem sygnału ALE i jest w nim pamiętany aż do ponownego wpisu. Pamięć programu jest odblokowywana poziomem logicznego zera sygnału $\overline{PSEN}$. Narastające zbocze tego sygnału określa moment, w którym rozkaz jest przyjmowany przez procesor. Na czas wysyłania przez port P0 adresu i pobierania rozkazu, przez port P2 jest wysyłany bardziej znaczący bajt adresu, rysunek 5-9. Strzałki na rysunkach wskazują momenty czasowe, w których następuje zapamiętanie adresu, pobranie rozkazu lub przesłanie danej. PCL oznacza mniej znaczący bajt licznika rozkazu, a PCH - bardziej znaczący bajt licznika rozkazu.



Rys. 5-9 Przebiegi czasowe podczas pobierania rozkazów z pamięci zewnętrznej.

Gdy ma nastąpić przesyłanie danych do lub z zewnętrznej pamieci RAM, to w zatrzasku jest zapamiętywany mniej znaczący bajt adresu komórki pamięci RAM. Sygnał ALE przyjmuje poziom 0 logicznego a sygnał $\overline{PSEN}$ 1 logicznej. Jeżeli dana jest zapisywana do pamięci, to sygnał $\overline{WR}$ przyjmuje, na czas wpisywania danej, poziom 0 logicznego. Natomiast gdy dana jest odczytywana z pamięci, to poziom 0 przyjmuje sygnał $\overline{RD}$, rysunek 5-10.



Rys. 5-10. Przebiegi czasowe przy dostępie do zewnętrznej pamięci RAM.

Zewnętrzna pamięć RAM może być adresowana w całym obszarze 64 KB - wtedy na liniach portu P2 pojawia się bardziej znaczący bajt adresu. Przy tym sposobie adresowania adres jest pobierany z 16-bitowego rejestru DPTR, który składa się z dwóch rejestrów 8-bitowych: DPL, zawierający mniej znaczący bajt adresu i DPH, zawierający bardziej znaczący bajt adresu.

Zewnętrzna pamięć RAM może być również adresowana poprzez rejestry R0 lub R1. Ponieważ rejestry te są rejestrami ośmiobitowymi, to można adresować obszar do 256 bajtów. Linie portu P2 mogą służyć wtedy jako normalne linie wejścia - wyjścia, lub jeżeli są połączone z wejściami adresowymi pamięci RAM, to programując odpowiednio port P2 można wybierać określone obszary pamięci RAM o objętości 256 bajtów. Jest to tzw. stronicowanie pamięci.

Dostęp do pamięci RAM kończy się wraz z taktem S3. W ten sposób procesor może w takcie S5 wysłać adres rozkazu pobieranego w następnym cyklu maszynowym.

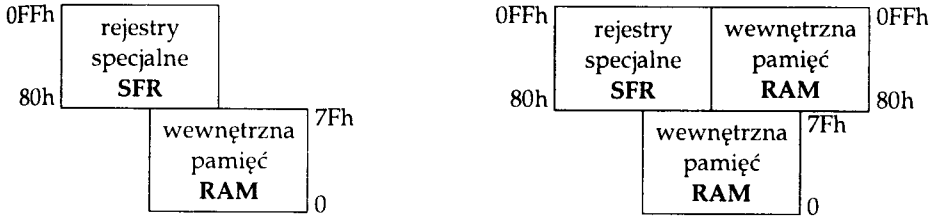
Dzięki wydzieleniu sygnałów $\overline{PSEN}$ oraz sygnałów $\overline{RD}$ i $\overline{WR}$, pamięć programu i pamięć danych są rozdzielone, mimo że znajdują się w tej samej przestrzeni adresowej.

? Pytania i problemy

1. W jakim układzie pracuje oscylator mikrokontrolera rodziny '51 ?
2. Jaki jest czas zerowania mikrokontrolerów rodziny '51 i co w tym czasie wykonuje mikrokontroler ?
3. Jakie są układy zerowania mikrokontrolerów rodziny '51 ?
4. Co to jest cykl maszynowy i z czego się składa ?
5. Jak wygląda pobieranie rozkazów w zależności od liczby bajtów rozkazów i liczby cykli maszynowych potrzebnych na ich wykonanie ?
6. Jakie elementy są niezbędne do uruchomienia mikrokontrolera z zewnętrzną pamięcią programu ?
7. Do czego służą sygnały ALE, $\overline{PSEN}$, $\overline{RD}$, $\overline{WR}$?
8. Czym różni się odczyt danych z zewnętrznej pamięci RAM od pobrania rozkazu z zewnętrznej pamięci programu ?

6. RAM czy SFR ?

Wewnętrzna pamięć RAM w mikrokontrolerze 8051 liczy tylko 128 bajtów. Wystarcza więc 8-bitowa magistrala adresowa. W następnym mikrokontrolerze 8052 konstruktorzy podwoili pamięć do 256 bajtów. Magistrala adresowa nie uległa zmianie. Rejestry specjalne SFR są odzwierciedleniem wszystkich wewnętrznych układów mikrokontrolera: liczników, łącza szeregowego, przetwornika analogowo-cyfrowego itd. Wzajemne położenie obu bloków, wewnętrznej pamięci RAM i rejestrów specjalnych SFR pokazano na rysunku 6-1.



Rys. 6-1 Wewnętrzna pamięć RAM i rejestry specjalne SFR w mikrokontrolerach rodziny '51.

O ile w mikrokontrolerze 8051 wewnętrzna pamięć RAM i rejestry specjalne SFR wzajemnie się uzupełniają, o tyle już w mikrokontrolerze 8052 połowa wewnętrznej pamięci RAM pokrywa się z rejestrami specjalnymi SFR (mają te same adresy: 80h .. 0FFh). **Rozróżnienie obu obszarów możliwe jest przez właściwe adresowanie:**

- **bezpośrednie przy odwołaniach do rejestrów specjalnych SFR**, np. do rejestru B przez podanie jego symbolu (B) lub adresu (0F0h):

  ```
  MOV   B,#4Eh          ;B ← 4Eh
  ```

  lub

  ```
  MOV   0F0h,#4Eh       ;(0F0h) ← 4Eh
  ```

  Wszystkie rejestry specjalne mają swoje nazwy. W trakcie asemblacji nazwom rejestrów przyporządkowane zostają właściwe adresy.

- **pośrednie dla wewnętrznej pamięci RAM o adresach 80H .. 0FFh**; pośrednio tzn. za pośrednictwem rejestru R0 lub R1 (tak jak przy odczycie i zapisie danych z/do zewnętrznej pamięci RAM). Dla przykładu wpisanie zawartości akumulatora A do komórki wewnętrznej pamięci RAM o adresie 80h wymaga wykonania:

```
MOV     R0,#80h         ;R0 ← 80h
MOV     @R0,A           ;(R0) ← A
```

Przy adresowaniu pośrednim nie należy zapominać o znaku @ przed nazwą właściwego rejestru. Tym rejestrem może być tylko rejestr R0 lub R1.

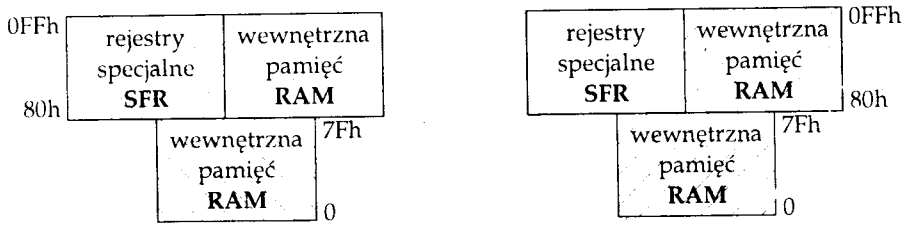
- **bezpośrednie i pośrednie w przypadku wewnętrznej pamięci RAM o adresach 0 .. 7Fh**. Dla tej części pamięci RAM dozwolone są oba tryby adresowania, ponieważ adres jednoznacznie wskazuje na komórkę pamięci. Przesłanie zawartości komórki wewnętrznej pamięci RAM o adresie 4Dh do akumulatora można zrealizować dwojako:

```
MOV     A,4Dh           ;A ← (4Dh)
```

lub

```
MOV     R1,#4Dh         ;R1 ← 4Dh
MOV     A,@R1           ;A ← (R1)
```

Graficznie oba sposoby, tryby adresowania wewnętrznej pamięci RAM i rejestrów specjalnych SFR przedstawiono na rysunku 6-2.



Rys. 6-2 Adresowanie wewnętrznej pamięci RAM i rejestrów specjalnych SFR.

Kolizja adresów wewnętrznej pamięci RAM i rejestrów specjalnych w mikrokontrolerach 8052 i nowszych (np. 80515/535) została usunięta przez właściwy dla danego obszaru **sposób adresowania**:

- **bezpośrednio i pośrednio wewnętrznej pamięci RAM o adresach 0 .. 7Fh**; w adresowaniu pośrednim nie zapominając o znaku @,
- **tylko pośrednio wewnętrznej pamięci RAM o adresach 80h .. 0FFh**,
- **tylko bezpośrednio rejestrów specjalnych SFR**.

Podane sposoby adresowania dotyczą wszystkich mikrokontrolerów rodziny '51.

6.1 Gdzie są rejestry R0 .. R7 ?

W 128-bajtowym segmencie wewnętrznej pamięci RAM o adresach od 0 do 7Fh rozmieszczone są rejestry R0 i R1, które wykorzystywane są przy adresowaniu pośrednim. Oprócz nich konstruktorzy mikrokontrolera dali użytkownikom do dyspozycji 6 następnych rejestrów oznaczonych kolejnymi symbolami cyfrowymi: R2, R3, R4, R5, R6 i R7. W ten sposób mamy w mikrokontrolerze 8 rejestrów. Osiem rejestrów (R0, .., R7) tworzy bank rejestrów i takich banków mamy 4 oznaczonych symbolami RB0 (bank numer 0), .. , RB3 (bank numer 3).

adres szesnastkowo:				adres dziesiętnie:
	7Fh	Obszar ogólnie dostępny		127
	.....			.....
	30h			48
	2Fh	Obszar o specjalnym przeznaczeniu		47
	.....			.....
	20h			32
	1Fh	Rejestr **R7**	Bank rejestrów **RB3**	31
	.....			.....
	18h	Rejestr **R0**		24
	17h	Rejestr **R7**	Bank rejestrów **RB2**	23
po zerowaniu	.....			.....
procesora	10h	Rejestr **R0**		16
linią RST	0Fh	Rejestr **R7**	Bank rejestrów **RB1**	15
(ReSeT)	.....			.....
wskaźnik stosu	8	Rejestr **R0**		8
SP=7	7	Rejestr **R7**	Bank rejestrów **RB0**	7
	6	Rejestr **R6**		6
	5	Rejestr **R5**		5
	4	Rejestr **R4**		4
	3	Rejestr **R3**		3
	2	Rejestr **R2**		2
	1	Rejestr **R1**		1
	0	Rejestr **R0**		0

Rys. 6-3 Rozmieszczenie rejestrów R0, .. , R7 w wewnętrznej pamięci RAM mikrokontrolera.

Z rysunku 6-3 wynika, że wszystkie banki rejestrów rozmieszczone są w początkowym obszarze wewnętrznej pamięci RAM mikrokontrolera. Oznacza to, że do każdego z rejestrów można odwołać się przez:

- podanie jego symbolu i numeru banku rejestrów, np. rejestr R7 w banku RB1,
- podanie adresu komórki wewnętrznej pamięci RAM, np. adres 0Fh dla tego samego rejestru.

Pomimo, że mamy do dyspozycji 4 banki rejestrów, każdy po 8 rejestrów (łącznie 32 rejestry), to **programowy dostęp przez podanie symbolu rejestru możliwy jest tylko do jednego, wybranego banku**. Po sprzętowym zerowaniu procesora (linią RST) wybrany jest jako domyślny bank RB0, tzn. rejestry R0, .. R7 o adresach od 0 do 7. **Adresując bezpośrednio wewnętrzna pamięć RAM w zakresie 0 .. 1Fh dostępne są wszystkie 32 rejestry** (traktowane jako komórki pamięci).

Poza rejestrami R0, .. , R7 wszystkie inne rejestry, np. akumulator A, rejestr B, rejestr słowa statusowego PSW itd., znajdują się w bloku, segmencie rejestrów specjalnych SFR (patrz rysunek 6-1).

Jeśli w mikrokontrolerach rodziny '51 przewidziano 4 banki rejestrów (RB0, .. RB3) to oznacza, że wystarczą 2 bity do określenia numeru banków. Te dwa bity oznaczone symbolami RS0 i RS1 (Register Select) znajdują się w rejestrze słowa statusowego PSW (Program Status Word):

rejestr PSW								adres
	CY	AC	F0	**RS1**	**RS0**	OV	F1	P	0D0h

Kodowanie numeru banku rejestrów za pomocą bitów RS1 i RS0 jest binarne, tzn.:

- RS1,RS0=00b oznacza zerowy bank rejestrów RB0,
- RS1,RS0=11b oznacza trzeci bank rejestrów RB3.

W liście rozkazów przewidziano następujące instrukcje dotyczące rejestrów R0 .. R7 i wymiany danych z wewnętrzną pamięcią RAM oraz rejestrami specjalnymi SFR:

```
MOV   Rn,#dana      ;Rn ← dana
MOV   Rn,A          ;Rn ← A
MOV   A,Rn          ;A ← Rn
MOV   Rn,adr        ;Rn ← (adr)
MOV   adr,Rn        ;(adr) ← Rn
XCH   A,Rn          ;A ↔ Rn, wzajemna wymiana danych
```

Instrukcje związane tylko z rejestrami R0 i R1 (rejestry te wykorzystywane są wyłącznie do adresowania pośredniego pamięci RAM) są następujące:

```
MOV   @Ri,#dana     ;(Ri) ← dana
```



```
MOV   @Ri,A         ;(Ri) ← A
MOV   A,@Ri         ;A ← (Ri)
MOV   @Ri,adr       ;(Ri) ← (adr)
MOV   adr,@Ri       ;(adr) ← (Ri)
XCH   A,@Ri         ;A ↔ (Ri), wzajemna wymiana danych
XCHD  A,@Ri         ;A3..0 ↔ (Ri)3..0, wzajemna wymiana tylko
                    ;                  4 mniej znaczących bitów
```

Z akumulatorem A związano instrukcję przestawienia czterech mniej i czterech bardziej znaczących bitów, w działaniu zbliżonym do instrukcji XCHD:

```
SWAP  A             ; A7..4 ↔ A3..0,
                    ;wzajemna wymiana 4 mniej znaczących
                    ;bitów z 4 bardziej znaczącymi bitami
                    ;akumulatora A
```

Rejestry R0 i R1 stosowane są również do adresowania zewnętrznej pamięci RAM, co przedstawiono w poprzednich rozdziałach:

```
MOVX  @Ri,A         ;(Ri)XDATA ← A
MOVX  A,@Ri         ;A ← (Ri)XDATA
```

Użyte w przykładach skróty mają następujące znaczenie:

Rn - rejestr R0 .. R7; n=0 .. 7,

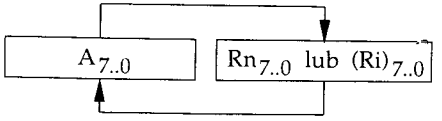
Ri - rejestr R0 lub R1; i=0 lub 1,

adr - adres pierwszych 128 bajtów wewnętrznej pamięci RAM (adr=0 .. 7Fh) lub rejestrów specjalnych SFR (adr=80h .. 0FFh),

dana - 8-bitowa zmienna wpisywana do wybranego rejestru Rn,

( ) - zawartość komórki pamięci danych RAM lub rejestrów specjalnych SFR o adresie podanym w nawiasie.

Poza typowymi przesłaniami (instrukcje MOV) w liście rozkazów znajdują się rozkazy, które przyspieszają wzajemną wymianę (eXCHange) zawartości akumulatora A i wybranego rejestru Rn lub komórki wewnętrznej pamięci RAM. Poglądowo przedstawia to rysunek:



Rozkazy XCH mogą być zastąpione innymi rozkazami, np:

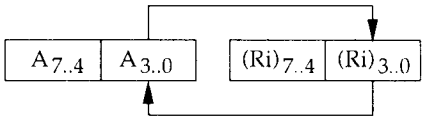
- wymiana typu XCH A,Rn zastąpiona trzema równoważnymi rozkazami ale z użyciem dodatkowego rejestru do przechowywania zmiennej, np. rejestru B:

```
MOV   B,A      ;B ← A, ochrona akumulatora A
MOV   A,Rn     ;A ← Rn
MOV   Rn,B     ;Rn ← B, czyli przesłanie akumulatora A
```

- rozkaz XCH A,@Ri można również zastąpić kombinacją trzech innych instrukcji, także z użyciem dodatkowego rejestru do przechowywania zmiennej, np. rejestru B

```
MOV   B,A      ;B ← A, ochrona akumulatora A
MOV   A,@Ri    ;A ← (Ri)
MOV   @Ri,B    ;(Ri) ← B, czyli przesłanie akumulatora A
```

Podobna w działaniu do instrukcji XCH A,@Ri jest instrukcja XCHD A,@Ri ale wzajemna wymiana dotyczy jedynie 4 mniej znaczących bitów (eXCHange Digit) akumulatora A i komórki wewnętrznej pamięci RAM adresowanej rejestrem Ri (rejestr R0 lub R1). Instrukcja ta stosowana jest w operacjach z liczbami przedstawionymi w kodzie BCD. Graficznie działanie tej instrukcji ilustruje poniższy rysunek:



Przykłady zastąpienia instrukcji XCH i XCHD innymi instrukcjami dobitnie świadczą, że niezbędny jest pomocniczy wewnętrzny rejestr. Rejestr taki rzeczywiście istnieje w mikrokontrolerze ale nie jest dostępny dla programisty. Na rysunku 2-2 przedstawiającym schemat blokowy mikrokontrolera zaznaczone są dwa rejestry pomocnicze.

Cztery banki rejestrów RB0 .. RB3 używane są zamiennie w różnych procedurach. Najczęściej roboczym bankiem jest bank RB0. Inne używane są w zależności od potrzeb, np: przy transmisji szeregowej bank RB1, wyświetlaniu danych na polu odczytowym - bank RB2, a obsługą przetwornika zajmują się rejestry w banku RB3.

Zmiana banku rejestrów jest najszybszym sposobem ochrony zawartości rejestrów przy wywołaniach podprogramów, a w szczególności w procedurach obsługi przerwań.

6.2 Co zawierają rejestry specjalne SFR ?

Jak już wcześniej stwierdzono, rejestry specjalne SFR są odbiciem wszystkich wewnętrznych układów mikrokontrolera. Konstruktorzy mikrokontrolera 8051 przewidzieli miejsce na 128 rejestrów specjalnych. Rejestrów adresowanych tylko bezpośrednio, o adresach od 80h do 0FFh. W miarę upływu czasu, rozwoju nowych technologii i potrzeb użytkowników pojawiły się nowe mikrokontrolery ze zmienionymi lub całkiem nowymi układami wewnętrznymi. Każdy nowy element zmieniał tylko znaczenie kolejnych komórek rejestrów specjalnych. Z 21 rejestrów specjalnych w mikrokontrolerze 8051 do 42 w 80515/535. Lista rozkazów, sposoby adresowania pozostały niezmienione.

Adresy i symbole wszystkich rejestrów specjalnych w mikrokontrolerach 8051, 8052 i 80515/535 pokazano na rysunku 6-4. Rejestry, których funkcje są zmienione lub całkowicie nowe rejestry zostały zacieniowane zgodnie z oznaczeniami pod rysunkiem.

Z rysunku 6-4 widać, że nie wszystkie rejestry są zagospodarowane, np. o adresach 0E1h..0EFh. Wolne rejestry nie mogą być jednak wykorzystane w programach jako dodatkowe komórki wewnętrznej pamięci RAM. Tylko niektóre z rejestrów mogą pełnić taką funkcję. O tym decyduje producent mikrokontrolera oraz programista. Analizując program wiadomo jakie jest wykorzystanie układów peryferyjnych, które mogą zmieniać w trakcie swojej pracy zawartość tych rejestrów. Dla przykładu rejestr SBUF związany z transmisją szeregową reprezentuje dwa niezależne rejestry. Jeden do zapisu (rejestr nadawanego znaku), a drugi do odczytu (rejestr odebranego znaku). Odczyt zawartości rejestru SBUF daje inną wartość niż wpisana.

Poniżej omówiono, łącznie z adresami, wybrane rejestry specjalne SFR mikrokontrolerów rodziny '51. Oznaczenie **rw-00h** sygnalizuje:

→ możliwość odczytu zawartości rejestru (read),
→ możliwość wpisu (write) innych wartości zgodnie z wymaganiami programu,
→ zawartość rejestru po sprzętowym zerowaniu mikrokontrolera, **00h**.

Akumulator (ACC - Accumulator) - adres 0E0h (rw-00h).

Jest jednym z ważniejszych rejestrów procesora, ponieważ większość rozkazów wykorzystuje ten rejestr. Oznaczany jest w mnemonikach instrukcji jako ACC (symboliczny adres komórki rejestru specjalnego) lub skróconym symbolem A (nazwa rejestru).

0F8h	**P5**							
0F0h	B							
0E8h	**P4**							
0E0h	ACC							
0D8h	**ADCON**	**ADDAT**	**DAPR**	**P6**				
0D0h	PSW							
0C8h	**T2CON**		***RCAP2L*** / **CRCL**	***RCAP2H*** / **CRCH**	**TL2**	**TH2**		
0C0h	**IRCON**	**CCEN**	**CCL1**	**CCH1**	**CCL2**	**CCH2**	**CCL3**	**CCH3**
0B8h	**IEN1** / **IP**	**IP1**						
0B0h	P3							
0A8h	**IEN0** / **IE**	**IP0**						
0A0h	P2							
98h	SCON	SBUF						
90h	P1							
88h	TCON	TMOD	TL0	TL1	TH0	TH1		
80h	P0	SP	DPL	DPH				PCON

rejestry adresowane bitowo i bajtowo

ACC — w 8051, 8052 i 80515/535

IE — zmieniona funkcja w 8052: IE, IP

T2CON — tylko w 8052 i 80515/535: T2CON, TL2, TH2

RCAP2L — tylko w 8052: RCAP2L, RCAP2H

ADDAT — tylko w 80515/535

Rys. 6-4 Mapa rejestrów specjalnych SFR w mikrokontrolerach 8051, 8052 i 80515/535.

Rejestr B - adres 0F0h (rw-00h).

Rejestr specjalnego przeznaczenia w operacjach mnożenia i dzielenia, zawierający jeden z argumentów oraz bardziej znaczącą część wyniku mnożenia lub resztę w operacji dzielenia. W innych sytuacjach może być używany jako rejestr ogólnego przeznaczenia.

Rejestr słowa statusowego PSW (Program Status Word) - adres 0D0h (rw-00h).

Rejestr zawiera bitowe informacje o wykonanej operacji, najczęściej o stanie akumulatora A oraz numer wybranego banku rejestrów.

Znaczenie poszczególnych bitów jest następujące:

rejestr PSW — adres 0D0h, rw-00h

CY	AC	F0	RS1	RS0	OV	F1	P

CY (PSW.7) znacznik przeniesienia (Carry Flag) z pozycji najbardziej znaczącego bitu akumulatora A_7; przekroczenie zakresu liczb całkowitych bez znaku,

AC (PSW.6) znacznik przeniesienia połówkowego (Auxiliary Carry Flag) między bitami akumulatora A_3 i A_4,

F0 (PSW.5) znacznik F0 ogólnego przeznaczenia,

RS1 (PSW.4) bit 1 wyboru banku rejestrów (Register Bank Select 1),

RS0 (PSW.3) bit 0 wyboru banku rejestrów (Register Bank Select 0),

OV (PSW.2) znacznik nadmiaru (Overflow Flag) dla dodawania i odejmowania liczb całkowitych ze znakiem w kodzie uzupełnienia do 2; przekroczenie zakresu liczb całkowitych bez znaku,

F1 (PSW.1) znacznik F1 ogólnego przeznaczenia, brak w mikrokontrolerze 8051/31,

P (PSW.0) znacznik parzystości (Parity Flag) będący dopełnieniem do parzystej liczby jedynek w akumulatorze A.

Znaczniki **F0** i **F1** nie mają swojego specjalnego przeznaczenia i mogą być używane jako 1-bitowe komórki pamięci lub 1-bitowy rejestr. Znaczniki te są testowane w rozkazach skoków warunkowych.

Bity **RS1** i **RS0** wyboru banku rejestrów umożliwiają uaktywnienie jednego z czterech banków:

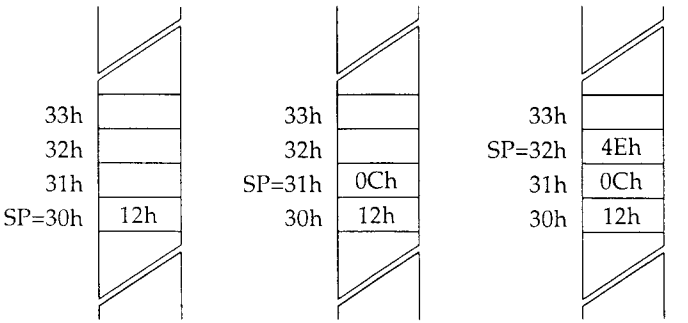
RS1	RS0	Numer wybranego banku rejestrów	Adres wybranego banku rejestrów
0	0	0 (RB0)	00h .. 07h
0	1	1 (RB1)	08h .. 0Fh
1	0	2 (RB2)	10h .. 17h
1	1	3 (RB3)	18h .. 1Fh

Wskaźnik stosu SP (Stack Pointer) - adres 81h (rw-00h).
Jest rejestrem, który adresuje stos - wydzielony logicznie fragment wewnętrznej pamięci RAM. Stos używany jest do zapamiętywania adresów powrotu z podprogramów wywoływanych programowo lub sprzętowo (przez przerwania), przenoszenia zmiennych między programami, czasowej ochrony rejestrów specjalnych i komórek wewnętrznej pamięci RAM jeśli są one zmieniane w trakcie wykonywania podprogramów.

Informacja wpisywana do stosu powoduje automatyczne zwiększenie zawartości wskaźnika stosu o liczbę wpisywanych bajtów. Jeśli w trakcie wykonanej instrukcji do stosu wpisywany jest jeden bajt to wskaźnik stosu zwiększany jest o jeden, jeśli dwa bajty to o dwa. Odczyt, pobranie jednego lub dwóch bajtów ze stosu powoduje automatyczne zmniejszenie zawartości wskaźnika stosu o jeden lub o dwa. 8-bitowy wskaźnik stosu SP wskazuje ostatnią zajętą komórkę stosu. Zrealizowanie poniższego programu powoduje:

```
MOV   SP,#2Fh      ;SP ← 2Fh,     wpisanie wartości początkowej
PUSH  10h          ;SP ← SP + 1,
                   ;(SP) ← (10h)
      ;wpisanie do stosu zawartości komórki wewnętrznej
      ; pamięci RAM o adresie 10h, np. (10h)=12h
PUSH  ACC          ;SP ← SP + 1,
                   ;(SP) ←ACC
      ;wpisanie do stosu zawartości akumulatora, np. A=0Ch
PUSH  B            ;SP ←SP + 1,
                   ;(SP) ←B
      ;wpisanie do stosu zawartości rejestru B, np. B=4Eh
```

powoduje wpisywanie do stosu 3 bajtów zgodnie z kolejnością wprowadzania. Sytuację tę przedstawia rysunek 6-5.



Rys. 6-5 Rozmieszczenie bajtów na stosie mikrokontrolera.

Przy odczytywaniu bajtów ze stosu instrukcją POP należy zachować właściwą kolejność. Zmiana kolejności, jeśli jest nieświadoma, może prowadzić do niezamierzonych skutków, ponieważ odczytywane bajty pobierane są zawsze ze szczytu stosu:

```
POP   10h          ;(10h) ← (SP),     (10h)=4Eh
                   ;SP ← SP -1
POP   ACC          ;ACC ←(SP),        ACC=0Ch
                   ;SP ← SP -1
POP   B            ;B ←(SP),          B=12h
                   ;SP ← SP -1
```

Jeśli rejestry i wybrana komórka pamięci RAM mają mieć przywrócone poprzednie wartości to po wykonaniu programu jedynie akumulator ma stan początkowy.

Wskaźnik stosu SP jest inkrementowany, zwiększana jest jego zawartość, **przed wykonaniem instrukcji PUSH** (o jeden) **i CALL** (o dwa) **oraz dekrementowany**, zmniejszana jest jego zawartość, **po pobraniu danych ze stosu w trakcie wykonywanych instrukcji POP** (o jeden) **i RET lub RETI** (o dwa). **Zerowanie procesora linią RST ustala wartość początkową wskaźnika stosu SP=7.** Oznacza to, że przy przesłaniu pierwszej danej do stosu następuje wpisanie jej do komórki o adresie 8, tzn. do rejestru R0 pierwszego banku rejestrów RB1 (rysunek 6-3). Zawartość wskaźnika stosu może być programowo zmieniana.

Ze względu na 8-bitową długość rejestru wskaźnika stosu SP i deklarację w wewnętrznej pamięci RAM należy kontrolować rozmiar stosu. Przy niekontrolowanym przesyłaniu danych do obszaru stosu możliwe jest zniszczenie danych przechowywanych w wyższych obszarach adresu pamięci RAM oraz w kolejnych bankach rejestrów rozpoczynając od banku RB0.

W arytmetyce 8-bitowej dodawanie lub odejmowanie wykonywane jest tylko dla 8-bitów, co oznacza, że:

0FFh + 1= 0,
0 - 1 = 0FFh

Wskaźnikowy rejestr danych DPTR (Data Pointer Register) - adresy: 82h, 83h (rw-00h każdy).
16-bitowy wskaźnikowy rejestr danych DPTR złożony jest z dwóch 8-bitowych rejestrów, z części bardziej znaczącej DPH (adres 83h) i części mniej znaczącej DPL (adres 82h).

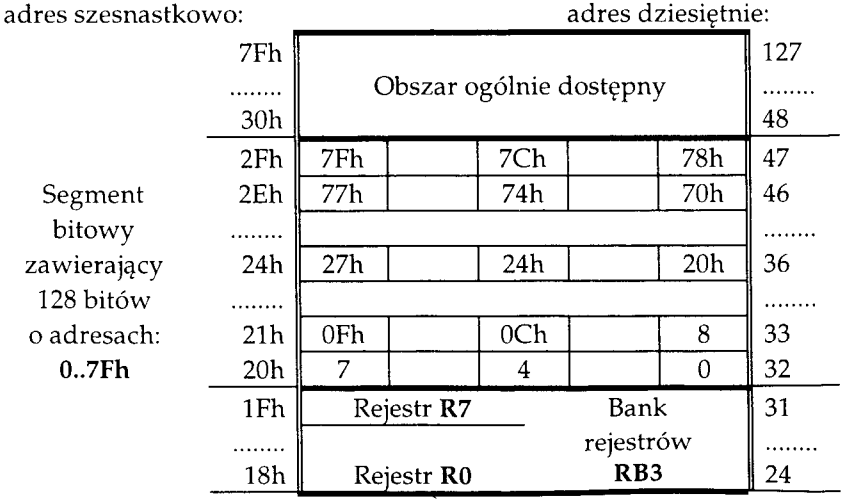
Rejestr ten stosowany jest do adresowania zewnętrznej pamięci danych RAM lub pamięci programu ROM w trybie indeksowo-rejestrowo pośrednim, np:

```
MOVX   A,@DPTR        ;A ← (DPTR)XDATA
MOVC   A,@A+DPTR      ;A ← (A+DPTR)CODE
```

6.3 Bit, bajt, słowo

W układach sterowania do zapamiętywania stanu obiektu wystarcza informacja dwu-stanowa. Silnik jest włączony lub wyłączony, suwnica jest w obszarze roboczym lub przesunęła się do końca - do ogranicznika, parametry wprowadzone przez użytkownika są poprawne lub błędna itd. Do pamiętania takich informacji wygodniej użyć jednego bitu niż całego bajtu. Przewidując takie sytuacje, konstruktorzy mikrokontrolera 8051 zaproponowali użytkownikom możliwość pamiętania danych bajtowo i bitowo. Dostępnych jest 256 indywidualnie adresowanych bitów w dwóch segmentach:

- w wewnętrznej pamięci RAM o adresach od 20h do 2Fh. 16 bajtów każdy po 8 bitów daje łącznie 128 bitów tak jak przedstawiono to na rysunku 6-6. Wszystkie bity adresowane są przez:
  - → przez podanie adresu bitu z przedziału: 0H .. 7Fh,
  - → przez odwołanie bajtowe i wskazanie numeru bitu w bajcie, np. bity od 0 do 7 adresowane są jako 20h.0 .. 20h.7, od 8 do 0Fh jako 21h.0 .. 21h.7 itd.



Rys. 6-6 Fragment wewnętrznej pamięci RAM z segmentem bitowym.

- w tych wszystkich rejestrach specjalnych SFR, których adresy podzielne są przez 8 bez reszty, tzn. o adresie 80h, 88h, .. , 0F0h i 0F8h, tak jak zaznaczono to na rysunku 6-4. Bity te mają adresy od 80h do 0FFh. Taki sposób dostępu do wybranych rejestrów specjalnych SFR jest bardzo użyteczny, ponieważ przeważnie każdy bit takiego rejestru odpowiada za inną funkcję pełnioną przez wewnętrzne układy. Ułatwia programowanie i czyni program czytelniejszym.

Łącznie w mikrokontrolerach rodziny '51 dostępnych jest 256 bitów o adresach od 0 do 0FFh.

Operacje bitowe znajdują swoje odzwierciedlenie w liście instrukcji mikrokontrolera. Tak jak w przesłaniach bajtowych uprzywilejowaną rolę odgrywa akumulator A, taką samą rolę w operacjach bitowych pełni znacznik przeniesienia C:

```
MOV   C,bit      ;C ← (bit)
MOV   bit,C      ;(bit) ← C

SETB  C          ;C ← 1,      ustawienie znacznika C
SETB  bit        ;(bit) ← 1,  ustawienie bitu

CLR   C          ;C ← 0,      zerowanie znacznika C
CLR   bit        ;(bit) ← 0,  zerowanie bitu
```

Wybór pierwszego banku rejestrów RB1 wymaga ustawienia bitu RS0 i zerowania bitu RS1 w rejestrze słowa statusowego PSW. Operację te można wykonać następująco:

```
→ bitowo:   SETB  RS0       ;RS0 ← 1,        symbol bitu
            CLR   RS1       ;RS1 ← 0
  lub       SETB  PSW.3     ;PSW.3 ← 1,      numer bitu i
            CLR   PSW.4     ;PSW.4 ← 0,      symbol rejestru
  lub       SETB  0D0h.3    ;(0D0h.3) ← 1    numer bitu i
            CLR   0D0h.4    ;(0D0h.4) ← 0    adres rejestru
  lub       SETB  0D3h      ;(0D3h) ← 1      adres bitu
            CLR   0D4h      ;(0D4h) ← 0

→ bajtowo:  MOV   PSW,#18h  ;PSW ← 18h
```

Obszar **wewnętrznej pamięci RAM o adresach 30h .. 7Fh i cały segment pamięci o adresach 80h .. 0FFh** przewidziany jest do **zastosowań ogólnych**, np. przechowywania danych, wyników obliczeń, wydzielenia jako stos itp. Dostęp do tego obszaru możliwy jest jedynie bajtowo.

Z instrukcjami bitowymi związane są również instrukcje umożliwiające rozgałęzienia programu. Wynika to ze szczególnej roli jaką pełnią bity rejestru statusowego PSW i bity w ogóle. Rozgałęzienia programu są efektem zaistnienia pewnych warunków, dlatego instrukcje realizujące te funkcje nazwane zostały skokami warunkowymi. W skład instrukcji skoków warunkowych wchodzą następujące rozkazy:

rozkazy 2-bajtowe:

```
JZ      rel     ;jeśli A=0 to skok do adresu = PC + 2 + rel
JNZ     rel     ;jeśli A≠0 to skok do adresu = PC + 2 + rel

JC      rel     ;jeśli C=1 to skok do adresu = PC + 2 + rel
JNC     rel     ;jeśli C=0 to skok do adresu = PC + 2 + rel
```

rozkazy 3-bajtowe:

```
JB      bit,rel ;jeśli bit=1 to skok do adresu = PC + 3 + rel
JNB     bit,rel ;jeśli bit=0 to skok do adresu = PC + 3 + rel

JBC     bit,rel ;jeśli bit=1 to bit ← 0 (zerowanie bitu) oraz
                ;              skok do adresu = PC + 3 + rel
```

We wszystkich podanych rozkazach licznik rozkazów PC zawiera adres wykonywanej instrukcji. Ponieważ każdy z rozkazów jest rozkazem 2 lub 3-bajtowym, dlatego do wartości licznika rozkazów PC dodawane jest 2 lub 3.

Wszystkie rozkazy testujące bity (z wyjątkiem rozkazu JBC) nie zmieniają wartości testowanego bitu. Instrukcja JBC bit,rel zeruje wskazany bit jeśli bit przed wykonaniem rozkazu miał wartość 1 i wykonuje skok do podanego adresu.

Konstruktorzy mikrokontrolera 8051 przewidzieli bitowy i bajtowy przepływ danych.

Jedyną operacją wykonywana na słowie, zmiennej 2-bajtowej, jest 3-bajtowy rozkaz:

```
MOV     DPTR,#dana_16
```

gdzie dana_16 jest zmienną 2-bajtową.

Nawet 16-bitowe rejestry wewnętrznych układów, takich jak rejestry liczników i rejestry sterujące pracą liczników, nie są dostępne słowowo. Dlatego wpisywanie do nich 16-bitowych wartości musi odbywać się dwu-etapowo. Najpierw do części mniej znaczącej, a później do części bardziej znaczącej (lub odwrotnie). Przykładowo wpis wartości początkowej 4AC6h do licznika T1 wymaga wykonania:


```
MOV     TL1,#0C6h       ;TL1 ← 0C6h
MOV     TH1,#4Ah        ;TH1 ← 4Ah
```

lub wykorzystując właściwości asemblera (polecenia asemblera):

```
MOV     TL1,#Low 4AC6h  ;TL1 ← 0C6h
MOV     TH1,#High 4AC6h ;TH1 ← 4Ah
```

Polecenia High i Low mają za zadanie wyodrębnić ze zmiennej 2-bajtowej 4AC6h:

- Low - bajt mniej znaczący czyli wartość 0C6h,
- High - bajt bardziej znaczący czyli wartość 4Ah.

Pytania i problemy

1. Jakie są sposoby adresowania wewnętrznej pamięci RAM i rejestrów specjalnych SFR ?

2. Czy rejestry specjalne SFR trzeba dołączać do mikrokontrolera jak zewnętrzną pamięć danych ?

3. Jakie adresy przyporządkowane są rejestrom specjalnym SFR ?

4. Ile rejestrów posiada mikrokontroler 8051, a ile 8052 ?

5. Co to są banki rejestrów, jak je zmieniać i adresować ?

6. Podaj przykłady instrukcji, w których do adresowania drugiego argumentu posłużono się adresowaniem:
   a) rejestrowym b) pośrednim c) bezpośrednim
   d) natychmiastowym e) indeksowym
   Podaj ograniczenia wynikające z sposobu adresowania.

7. Wyjaśnij jaka jest różnica między instrukcjami XCH, XCHD oraz SWAP.

8. Do czego służą rejestry:

a) A b) B c) PSW

d) SP e) DPTR

9. Co to jest stos i do czego służy ?

10. Jak zmienia się wskaźnik stosu SP w instrukcjach: PUSH, POP, LCALL i RET ?

11. Co to jest bit, bajt i słowo ? Podaj przykłady.

12. Czy pamięć programu ROM, wewnętrzną i zewnętrzną pamięć RAM oraz rejestry specjalne SFR można adresować bitowo i bajtowo ?

13. Ile bitów i o jakich adresach dostępnych jest w mikrokontrolerach rodziny '51 ? Czy liczba dostępnych bitów zależy od typu mikrokontrolera ?

14. Podaj przykłady instrukcji bitowych ?

15. Jakie jest przeznaczenia znacznika przeniesienia C w rozkazach bitowych ?

16. Czy w liście instrukcji mikrokontrolera 8051 występują rozkazy przesłań 2-bajtowych ? Jakich sytuacji dotyczą ?

7. Porty

Porty umożliwiają dołączenie do mikrokontrolera takich urządzeń zewnętrznych jak klawiatura, pole odczytowe, przekaźniki itp. Najczęściej porty posiadają osiem linii, którymi dane mogą być przesyłane w dwóch kierunkach, tzn. do lub z mikrokontrolera. Porty mogą być cyfrowe - przesyłające dane logiczne, lub analogowe, np. wejścia przetwornika A/C.

Mikrokontroler 8051 posiada cztery dwukierunkowe porty cyfrowe. Każdy port zawiera osiem linii, z których każda może pracować jako niezależna linia wejścia lub wyjścia. Inne mikrokontrolery rodziny '51 zawierają na ogół większą liczbę portów. Porty mogą być wielofunkcyjne, tzn. mogą być wykorzystywane do różnych celów.

Jak to już zostało opisane w rozdziale 5, port P0 przy współpracy mikrokontrolera z zewnętrzną pamięcią programu jest wykorzystywany do wysyłania mniej znaczącego bajtu adresu rozkazu oraz do przyjmowania bajtu rozkazu. Przy współpracy z zewnętrzną pamięcią danych RAM, lub pamięcią programu, przez port P0 jest również przesyłany mniej znaczący bajt adresu oraz są przesyłane dane do i z zewnętrznej pamięci RAM, lub z pamięci programu, a port P2 - do wysyłania bardziej znaczącego adresu dla zewnętrznej pamięci programu lub danych.

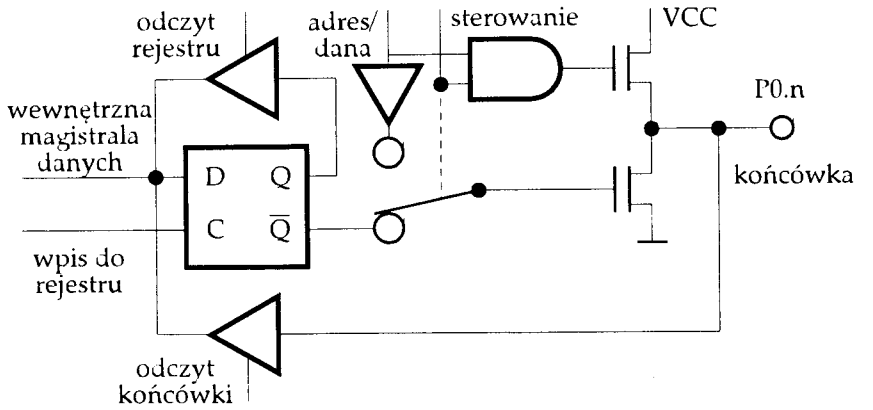
W mikrokontrolerze 8052 wyprowadzenia P1.0 i P1.1 portu P1 są również wielofunkcyjne, przy czym wyprowadzenie P1.0 może być wejściem taktującym licznik T2, a wyprowadzenie P1.1 - jako wejście T2EX służące do sterowania pracą licznika T2.

W mikrokontrolerze 80C515 linie P1.0 - P1.3 portu P1 mogą być wejściami przerwań zewnętrznych, wejściami przepisującymi zawartość licznika T2 do odpowiednich rejestrów lub wyjściami komparatorów współpracującymi z licznikiem T2. Również linie P1.5 i P1.7 są związane z licznikiem T2.

W porcie P3 każde wyprowadzenie ma swoją alternatywną funkcję, która występuje prawie we wszystkich mikrokontrolerach rodziny '51:

- P3.0/RXD - wejście portu szeregowego
- P3.1/TXD - wyjście portu szeregowego
- P3.2/$\overline{\text{INT0}}$ - wejście przerwania zewnętrznego lub bramkowania licznika T0
- P3.3/$\overline{\text{INT1}}$ - wejście przerwania zewnętrznego lub bramkowania licznika T1
- P3.4/T0 - wejście taktujące licznik T0
- P3.5/T1 - wejście taktujące licznik T1
- P3.6/$\overline{\text{WR}}$ - wyjście sygnału zapisu do zewnętrznej pamięci RAM
- P3.7/$\overline{\text{RD}}$ - wyjście sygnału odczytu z zewnętrznej pamięci RAM

Porty w mikrokontrolerach mogą mieć różną budowę wynikającą z przeznaczenia. Na rysunkach 7-1 ÷ 7-4 są przedstawione schematy portów mikrokontrolerów rodziny '51. Wszystkie one zawierają przerzutnik typu D, który jest elementem rejestru danego portu umieszczonego w obszarze SFR. Dana wysyłana na wyjście portu jest wpisywana właśnie do tego rejestru, np. rozkazem **MOV P1,A**, który powoduje przepisanie zawartości akumulatora do rejestru portu P1. Odczytywanie danej z portu odbywa się poprzez bufory, przy czym, w zależności od rozkazu, dana może być odczytana albo z rejestru portu, albo bezpośrednio z końcówki portu. Pozostałe elementy portów zależą od dodatkowego przeznaczenia danego portu. I tak w porcie P0, rysunek 7-1, znajduje się przełącznik przełączający wejście tranzystora wyjściowego do magistrali adresowej i bezpośrednio do magistrali danych, gdy mikrokontroler pobiera rozkazy z zewnętrznej pamięci programu lub komunikuje się z zewnętrzną pamięcią danych. Rejestr portu jest odłączony od końcówki i jego zawartość nie ma wpływu na sygnał wyjściowy portu. W tym trybie pracy jest również aktywny tranzystor dołączony do zasilania, dzięki czemu można uzyskać większy prąd wyjściowy do sterowania wejść pamięci zewnętrznych. Gdy port P0 pracuje jako normalny port, to tranzystor ten pracuje jak źródło prądowe.



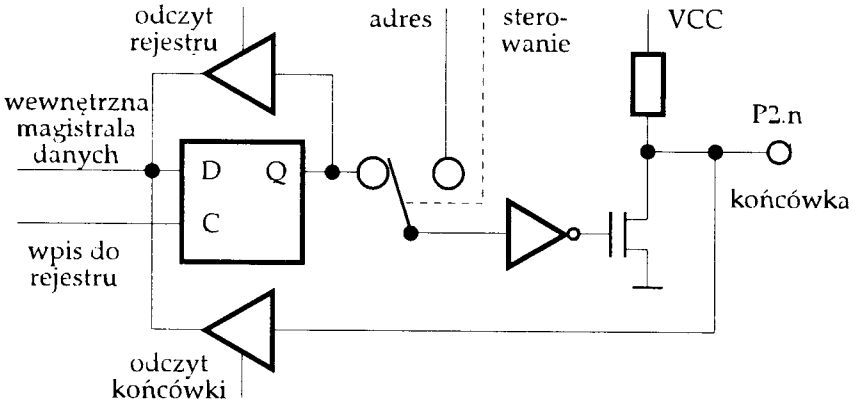
Rys. 7-1. Schemat portu P0.

Ponieważ przez port P2 dodatkowo może być przesyłany tylko adres (bardziej znaczący bajt), dlatego jego układ jest prostszy niż portu P0, rysunek 7-2. Zawiera on również przełącznik, dzięki któremu adres może być podany na wyjście portu niezależnie od zawartości rejestru portu. Obciążeniem tranzystora wyjściowego, podobnie jak w pozostałych portach jest rezystor, a w rzeczywistości źródło prądowe.

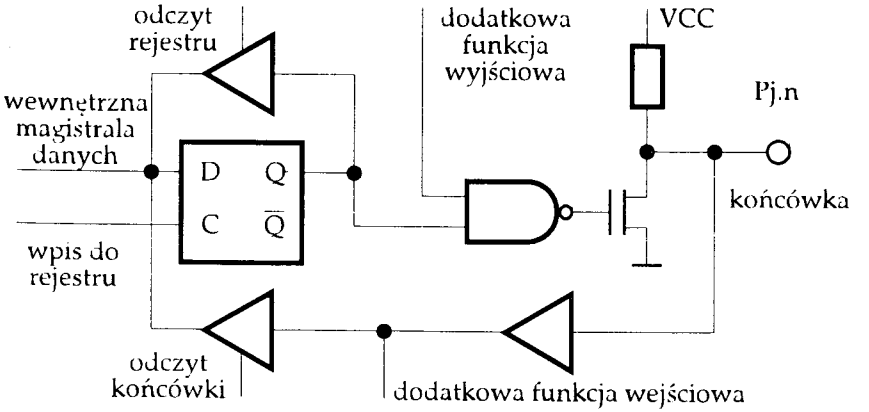
Konstrukcja pozostałych portów wielofunkcyjnych, wspólna dla całej rodziny mikrokontrolerów '51 jest pokazana na rysunku 7-3. W tych portach nie ma przełącznika dla dodatkowej funkcji wyjściowej, a sygnał funkcji jest wyprowadzony poprzez bramkę NAND, wspólnie z sygnałem wyjściowym rejestru

portu. Dlatego, aby funkcja wyjściowa mogła być aktywna, do rejestru portu musi być wpisana jedynka.

Sygnał dodatkowej funkcji wejściowej z końcówki portu, poprzez dodatkowy bufor, jest doprowadzony do odpowiednich układów, na przykład do liczników, portu szeregowego, układu przerwań zewnętrznych, itp. Również w przypadku dodatkowej funkcji zewnętrznej do rejestru portu musi być wpisana jedynka, gdyż inaczej tranzystor wyjściowy zostanie wprowadzony w stan nasycenia, zwierając końcówkę portu do masy. Z tego też powodu, by nie blokować działania dodatkowych funkcji, po zerowaniu mikrokontrolera rejestry wszystkich portów są ustawiane w stan 1.



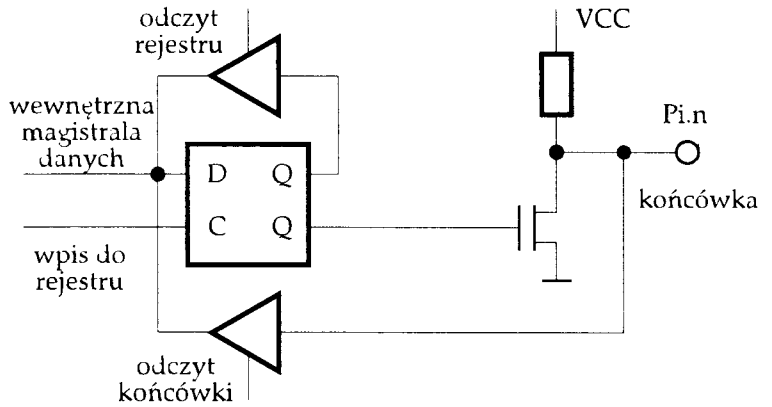
Rys. 7-2. Schemat portu P2.



Rys. 7-3. Schemat portu wielofunkcyjnego.

Schemat portów jednofunkcyjnych jest przedstawiony na rysunku 7-4. W układzie tym sygnał z wyjścia rejestru portu steruje bezpośrednio tranzystorem

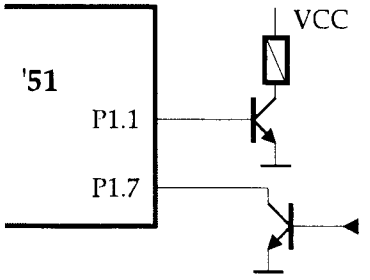
wyjściowym portu. Informacja wpisywana do rejestru jest przekazywana bezpośrednio na końcówkę portu.



Rys. 7-4. Schemat portu jednofunkcyjnego.

Zastosowanie jako obciążenia tranzystora wyjściowego portu źródła prądowego daje to, że zwarcie wyjść portów do masy nie powoduje ich zniszczenia. Natomiast zwarcie końcówki do zasilania przy stanie zera logicznego końcówki może zniszczyć tranzystor wyjściowy.

Do wejść portów można dołączać układy typu „otwarty kolektor", rysunek 7-5 - linia P1.7, gdyż obciążeniem dla nich jest właśnie źródło prądowe portu, albo bezpośrednio sterować bazę tranzystora ponieważ odpowiedni prąd bazy jest dostarczany ze źródła prądowego portu, rysunek 7-5 linia P1.1.



Rys. 7-5. Bezpośrednie połączenie tranzystorów z portami.

W wykonaniach standardowych mikrokontrolerów rodziny '51 wyjścia portów, poza portem P0, mogą sterować czterema wejściami TTL LS, natomiast mogą być sterowane układami TTL lub NMOS z otwartym kolektorem lub drenem. Wyjścia portu P0 mogą sterować ośmioma bramkami TTL LS. Jakie są rzeczywiste dopuszczalne obciążenia portów należy sprawdzać w danych katalogowych producenta stosowanego mikrokontrolera. Przykładowo, niektóre mi-

krokontrolery rodziny '51 firmy ATMEL dopuszczają wartość prądu płynącego przez tranzystor wyjściowy portu do 20 mA, co umożliwia bezpośrednie dołączenie do wyjścia portu diody LED lub wskaźnika 7-segmentowego.

Z rysunków 7-1 ÷ 7-4 wynika, że na wewnętrzną magistralę danych można przesyłać stan wyjścia rejestru portu lub jego końcówki. Może to wprowadzać niejednoznaczność przy pobieraniu informacji z portu. Na przykład, jeżeli sterujemy tranzystor, jak na rysunku 7-5, to w celu włączenia tranzystora do komórki rejestru portu (P1.1) wpisujemy stan jedynki logicznej. Ponieważ napięcie nasycenia złącza baza - emiter tranzystora wynosi 0,6 ÷ 0,7 V, to odczytując stan końcówki tego portu otrzymamy wartość zera logicznego.

Rozkazy odczytujące zawartość rejestru portu są to rozkazy, których wykonanie równocześnie modyfikuje ich zawartość (Read-Modify-Write Instructions), natomiast rozkazy czytające stan końcówki portu są to rozkazy, które nie wpisują wyniku operacji do rejestru portu.

Zestawy tych rozkazów są przedstawione w tabeli 7-1.

Tabela 7-1 Wykaz rozkazów operujących na rejestrach lub końcówkach portów.

Rozkazy czytające rejestry portów				Rozkazy czytające końcówki portów			
ANL	Pn,r	(ANL	P1,A)	ANL	r,Pn	(ANL	DPL,P1)
ORL	Pn,r	(ORL	P3,B)	ORL	r,Pn	(ORL	B,P3)
XRL	Pn,r	(XRL	P1,2Fh)	XRL	r,Pn	(XRL	23h,P1)
JBC	Pn.y,E_TA	(JBC	P1.3,PETLA)	JB	Pn.y,E_TA	(JB	P1.7,SKOK)
CPL	Pn.y	(CPL	P3.4)	JNB	Pn.y,E_TA	(JNB	P1.2,TAB1)
INC	Pn	(INC	P1)	CJNE	A,Pn,E-TA	(CJNE	A,P3,Z_1)
DEC	Pn	(DEC	P3)	MOV	r,Pn	(MOV	A,P1)
DJNZ	Pn,E-TA1	(DJNZ	P1,PETLA2)	ADD	A,Pn	(ADD	A,P2)
MOV	Pn.y,C	(MOV	P3.4,C)	ADDC	A,Pn	(ADDC	A,P3)
CLR	Pn.y	(CLR	P1.5)	SUBB	A,Pn	(SUBB	A,P1)
SET	Pn.y	(SET	P1.1)	XCH	A,Pn	(XCH	A,P3)
				PUSH	Pn	(PUSH	P1)

gdzie: n - numer portu y - numer bitu portu
r - rejestr SFR lub komórka pamięci E_TA - etykieta

Rejestry portów P0 ÷ P3 znajdują się w obszarze SFR pod tymi samymi adresami we wszystkich mikrokontrolerach rodziny '51. Wszystkie bity tych czterech portów mają tę właściwość, że posiadają własny indywidualny adres i mogą być ustawiane niezależnie odpowiednimi rozkazami. Najmniej znaczący bit ma adres odpowiadający adresowi całego rejestru.

Operowanie adresami poszczególnych bitów przy pisaniu programów jest bardzo uciążliwe, dlatego wszystkie narzędzia związane z programowaniem mikrokontrolerów dopuszczają adresowanie bitów poprzez podanie nazwy portu i numeru bitu. Na przykład bit 84h (port P0) może być opisany jako P0.4, a bit 0B0 jako P3.0.

P0	87h	86h	85h	84h	83h	82h	81h	80h	adres 80h rw=1
P1	97h	96h	95h	94h	93h	92h	91h	90h	adres 90h rw=1
P2	0A7h	0A6h	0A5h	0A4h	0A3h	0A2h	0A1h	0A0h	adres 0A0h rw=1
P3	0B7h	0B6h	0B5h	0B4h	0B3h	0B2h	0B1h	0B0h	adres 0B0h rw=1

We wszystkich rejestrach umieszczonych w obszarze SFR, których adresy są podzielne przez 8, każdy bit tych rejestrów może być ustawiany niezależnie.

Stan wyjść linii portów można ustawić rozkazami typu MOV, ale ma to tę niedogodność, że rozkazem tym ustawia się jednocześnie wszystkie linie danego portu. Wygodniej jest użyć rozkazów logicznych operujących na bajtach lub pojedynczych bitach.

W rodzinie mikrokontrolerów '51 są wykonywane następujące operacje logiczne na bajtach:

- CPL - negacja logiczna,
- CLR - zerowanie bitów,
- ANL - iloczyn logiczny,
- ORL - suma logiczna,
- XRL - różnica symetryczna.

Pierwsze dwie operacje są wykonywane wyłącznie w akumulatorze. Natomiast pozostałe mogą być wykonywane również poza akumulatorem na wszystkich rejestrach obszaru SFR i komórkach pamięci wewnętrznej RAM.

Na pojedynczych bitach można wykonywać operacje:

- CLR - zerowanie bitu,
- SETB - ustawianie bitu w stan 1,
- CPL - negacja bitu,
- ANL - iloczyn logiczny,
- ORL - suma logiczna.

Dwie ostatnie operacje mogą być wykonywane tylko z udziałem znacznika przeniesienia C. Należy zwrócić również uwagę, że na pojedynczych bitach nie można wykonywać operacji różnicy symetrycznej.

Zestaw pochodnych operacji logicznych jest podany w poniższych tabelach:

Operacje bajtowe:

Tabela 7-2 iloczyn logiczny

instrukcje:		zapis binarny instrukcji	wykonywane operacje
ANL	A,Rn	01011rrr	A ← A and Rn
ANL	A,adr	01010101 adr	A ← A and (adr)
ANL	A,@Ri	0101011i	A ← A and (Ri)
ANL	A,#dana	01010100 dana	A ← A and dana
ANL	adr,A	01010010 adr	(adr) ← (adr) and A
ANL	adr,#dana	01010011 adr dana	(adr) ← (adr) and dana

Tabela 7-3 suma logiczna

instrukcje:		zapis binarny instrukcji	wykonywane operacje
ORL	A,Rn	01001rrr	A ← A or Rn
ORL	A,adr	01000101 adr	A ← A or (adr)
ORL	A,@Ri	0100011i	A ← A or (Ri)
ORL	A,#dana	01000100 dana	A ← A or dana
ORL	adr,A	01000010 adr	(adr) ← (adr) or A
ORL	adr,#dana	01000011 adr dana	(adr) ← (adr) or dana

Tabela 7-4 różnica symetryczna

instrukcje:		zapis binarny instrukcji	wykonywane operacje
XRL	A,Rn	01101rrr	A ← A xor Rn
XRL	A,adr	01100101 adr	A ← A xor (adr)
XRL	A,@Ri	0110011i	A ← A xor (Ri)
XRL	A,#dana	01100100 dana	A ← A xor dana
XRL	adr,A	01100010 adr	(adr) ← (adr) xor A
XRL	adr,#dana	01100011 adr dana	(adr) ← (adr) xor dana

Tabela 7-5 zerowanie i negacja akumulatora

instrukcje:		zapis binarny instrukcji	wykonywane operacje
CLR	A	11100100	A ← 0
CPL	A	01110100	A ← not A

Operacje bitowe:

Tabela 7-6 iloczyn logiczny

instrukcje:		zapis binarny instrukcji	wykonywane operacje
ANL	C,bit	10000010 adres bitu	C ← C and bit
ANL	C,/bit	10110000 adres bitu	C ← C and (not bit)

Tabela 7-7 suma logiczna

instrukcje:		zapis binarny instrukcji	wykonywane operacje
ORL	C,bit	11110010 adres bitu	C ← C or bit
ORL	C,/bit	10100000 adres bitu	C ← C or (not bit)

Tabela 7-8 negacja bitu

instrukcje:		zapis binarny instrukcji	wykonywane operacje
CPL	C	10110011	C ← not C
CPL	bit	00110010 adres bitu	bit ← not bit

Tabela 7-9 ustawianie bitu w stan 1

instrukcje:		zapis binarny instrukcji	wykonywane operacje
SETB	C	11010011	C ← 1
SETB	bit	11010010 adres bitu	bit ← 1

Tabela 7-10 zerowanie bitu

instrukcje:		zapis binarny instrukcji	wykonywane operacje
CLR	C	11000011	C ← 0
CLR	bit	01110010 adres bitu	bit ← 0

gdzie: rrr - numer rejestru R0 ÷ R7 aktualnego banku rejestrów (rrr = 000 dla R0, rrr = 111 dla R7),

i - numer rejestru R0, R1 aktualnego banku rejestrów (i= 0 dla R0, i=1 dla R1),

adr - adres pierwszych 128 bajtów pamięci wewnętrznej RAM lub adresów rejestrów w obszarze SFR,

dana- zmienna 8-bitowa,

( ) - zawartość komórki pamięci danych lub rejestru w obszarze SFR o adresie podanym w nawiasach.

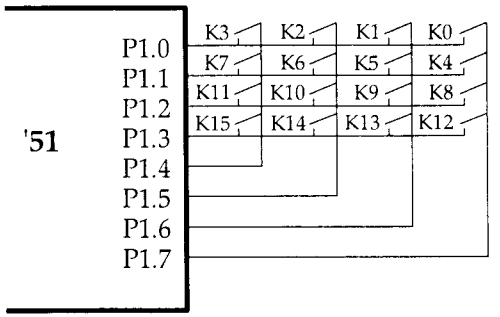
Operacje logiczne na zawartościach komórek pamięci wewnętrznej RAM umieszczonych powyżej adresu 127 (07Fh) mogą być wykonywane tylko przy pomocy rozkazów pośrednich - adresowanie poprzez rejestry R0 i R1.

Korzystając z rozkazów logicznych można ustawić w odpowiedni stan kilka linii portów jednocześnie. Na przykład, gdy linie 0, 3,5 i 6 portu P1 mają być ustawione w stan 1, to można to osiągnąć stosując rozkaz sumy logicznej ORL P1,#01101001b. Jeżeli odpowiednia dana znajduje się w akumulatorze, to można wykonać rozkaz ORL P1,A, który będzie o jeden bajt krótszy od poprzedniego. Pozostałe linie portu, którym odpowiadają wartości zerowe danej nie ulegną zmianie. Gdy trzeba odpowiednie linie portu ustawić w stan zera, należy zastosować rozkaz iloczynu logicznego, np. ANL P1,#53h. Po wykonaniu takiego rozkazu linie 2, 3, 5 i 7 będą miały stan 0, a pozostałe nie ulegną zmianie.

Jeżeli mają być ustawiane pojedyncze linie portu, to można użyć rozkazów operacji bitowych, np. CLR P3.5 - zerowanie linii 5 portu P3 lub SETB P1.1 - ustawienie w stan 1 linii 7 portu P1, co spowoduje włączenie przekaźnika z rysunku 7-5.

Dzięki temu że można dokonywać operacji na wybranych bitach portów, różne linie tego samego portu mogą równocześnie pracować jako linie wejściowe i wyjściowe

Poniżej jest podany przykład zastosowania portu P1 do odczytu klawiatury, rysunek 7-6. Klawiatura zawiera 16 przycisków oznaczonych od K0 do K15. Na liniach P1.0 ÷ P1.3 jest kolejno generowany stan 0. Naciśnięcie klawisza przenosi ten stan na linie P1.4 ÷ P1.7. Jeżeli żaden klawisz nie jest naciśnięty, to na liniach P1.4 ÷ P1.7 jest stan 1 wymuszony źródłami prądowymi układów wyjściowych portu. Program, który ma odczytać stan klawiatury, musi kolejno zadawać stany 0 na poszczególne linie P1.0 ÷ P1.3, odczytywać stan linii P1.4 ÷ P1.7 i na podstawie tej informacji określić numer naciśniętego klawisza.



Rys. 7-6. Schemat połączenia klawiatury z portem P1.

```
;*****************************************************************
; Program odczytu klawiatury
; Wykorzystane rejestry:  R2 - numer klawisza
;                         R3 - licznik pętli
; Po wyjściu z programu w rejestrze R2 znajduje się numer naciśniętego klawi-
;sza lub wartość 0FFh, jeżeli żaden klawisz nie został naciśnięty.
;*****************************************************************

        MOV   R2,#0         ;zerowanie rejestru z numerem klawisza
        MOV   R3,#4         ;licznik pętli
        MOV   P1,#0FEh      ;ustawienie w stan zera linii P1.0
        MOV   A,P1          ;odczyt końcówek portu P1

PETLA1:                     ;początek pętli w której kolejno są testowane stany
                            ;linii P1.7 ÷ P1.4
        RLC   A             ;rotacja w lewo zawartości akumulatora. Przy
                            ;pierwszym wejściu do pętli do znacznika C jest
                            ;wpisywany stan linii P1.7
        JNC   WYJSCIE       ;testowanie stanu znacznika C. Jeżeli C=0, tzn. że
                            ;został naciśnięty któryś z klawiszy K0 ÷K3 i
                            ;następuje wyjście z procedury testowania klawiszy.
```



```
        INC   R2            ;Jeżeli C=1, tzn. testowany klawisz nie został
                            ;naciśnięty, to jest zwiększana zawartość rejestru R2
                            ;(numer kolejnego klawisza)
        DJNZ  R3,PETLA1     ;odliczanie, czy zostały sprawdzone wszystkie linie
                            ;P1.7 ÷ P1.4. Jeżeli nie, to następuje skok do etykiety
                            ;PETLA1 - testowanie kolejnej linii, a jeżeli tak, to
                            ;przejście do sprawdzania klawiszy K4 ÷ K7.
        MOV   R3,#4         ;ponowne ustawienie licznika pętli
        MOV   P1,#0FDh      ;zero na linii P1.1
        MOV   A,P1

PETLA2:
        RLC   A
        JNC   WYJSCIE
        DJNZ  R3,PETLA2
        MOV   R3,#4
        MOV   P1,#0FBh      ;zero na linii P1.2
        MOV   A,P1

PETLA3:
        RLC   A
        JNC   WYJSCIE
        DJNZ  R3,PETLA3
        MOV   R3,#4
        MOV   P1,#0F7h      ;zero na linii P1.3
        MOV   A,P1

PETLA4:
        RLC   A
        JNC   WYJSCIE
        DJNZ  R3,PETLA4
        MOV   R2,#0FFh      ;żaden klawisz nie jest naciśnięty

WYJSCIE:                    ;koniec programu

END
```

Pytania i problemy

1. Co to są porty i jaka jest ich funkcja ?
2. Co to jest wielofunkcyjność portów ?
3. Jakie dodatkowe funkcje spełnia port P3 mikrokontrolerów rodziny '51 ?
4. Jakie rozkazy odczytują końcówkę portu, a jakie rejestr portu ?
5. Jakie jest powiązanie pomiędzy adresem portu a adresem poszczególnych znaczników portu ?
6. Na jakich rejestrów SFR można dokonywać operacji bitowych ?
7. Jakie operacje logiczne bajtowe i bitowe można wykonywać w mikrokontrolerach rodziny '51 ?
8. Jaka jest zasada przenoszenia danych bitowych pomiędzy znacznikami rejestrów i pamięci w mikrokontrolerach rodziny '51 ?

8. Przerwania.

Przerwanie (interrupt) to sposób pracy procesora polegający na tym, że pod wpływem sygnału zewnętrznego lub pochodzącego z układu wewnętrznego mikrokontrolera, procesor przerywa wykonywanie bieżącego programu i przechodzi do wykonywania podprogramu związanego z sygnałem przerwania. Podprogram obsługi przerwania jest zakończony specjalnym rozkazem, który powoduje, że procesor powraca do wykonywania przerwanego programu. Aby móc kontynuować przerwany program, przed wejściem do podprogramu obsługi przerwania, procesor musi zapamiętać miejsce, w którym nastąpiło przerwanie. Odbywa się to poprzez zapisanie zawartości licznika rozkazów do buforu pamięciowego nazywanym stosem.

W zależności od typu procesora czy mikrokontrolera, w momencie przejścia procesora do obsługi przerwania, na stosie oprócz zawartości licznika rozkazów mogą być składowane automatycznie również zawartości innych rejestrów. Ale w rodzinie '51 wywołanie przerwania, albo rozkaz CALL wywołujący podprogram, powoduje , że na stosie jest chowana tylko zawartość licznika rozkazów. Zawartości pozostałych rejestrów, np. rejestru PSW, DPTR itp. lub komórek pamięci muszą być chowane programowo.

Również w zależności od typu procesora, adres (interrupt vector) od którego rozpoczyna się podprogram obsługi przerwania jest określany w różny sposób. W jednych procesorach może być stały, umieszczony pod ściśle określonym adresem pamięci programu - tak jest w rodzinie '51, w innych może być określany programowo lub przez urządzenie zewnętrzne.

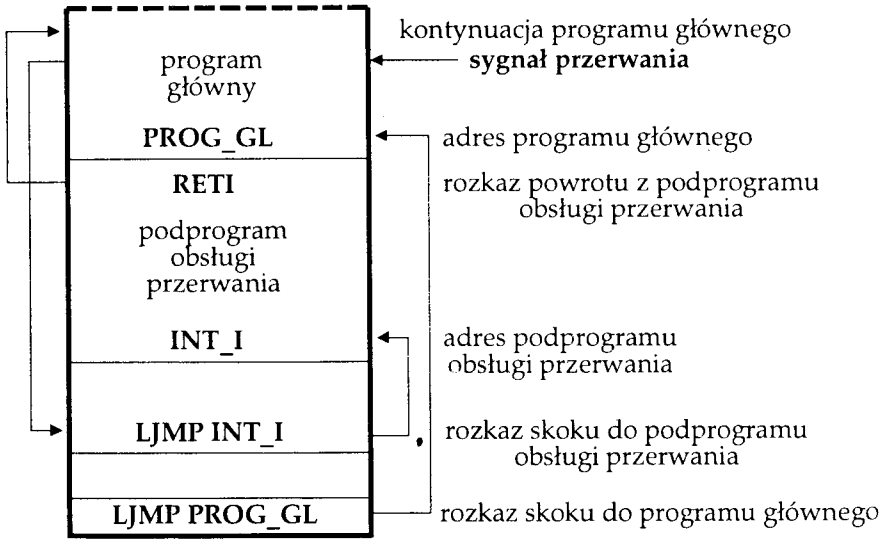
Tabela 8-1. Adresy obsługi przerwań w mikrokontrolerze 80C515/535.

Lp.	znacznik	źródło przerwania	adres
1	IE0	przerwanie zewnętrzne INT0	0003h
2	TF0	przerwanie od licznika T0	000Bh
3	IE1	przerwanie zewnętrzne INT1	0013h
4	TF1	przerwanie od licznika T1	001Bh
5	RI+TI	przerwanie od nad. i odb. portu szeregowego	0023h
6	TF2+EXF2	przerwanie od licznika T2 i wejścia T2EX	002Bh
7	IADC	przerwanie od przetwornika A/C	0043h
8	IEX2	przerwanie zewnętrzne INT2	004Bh
9	IEX3	przerwanie zewnętrzne INT3, komparator CRC	0053h
10	IEX4	przerwanie zewnętrzne INT4, komparator CC1	005Bh
11	IEX5	przerwanie zewnętrzne INT5, komparator CC2	0063h
12	IEX6	przerwanie zewnętrzne INT6, komparator CC3	006Bh

W mikrokontrolerach rodziny '51 adresy podprogramów obsługi przerwań są umieszczone w początkowym obszarze pamięci programu. Jako przykład, w tabeli 8-1 są podane adresy obsługi przerwań występujące w mikrokontrolerze 80C515/535.

Źródła przerwań oraz ich adresy z pozycji 1 ÷ 5 są wspólne dla całej rodziny '51. Przerwanie z pozycji 6 jest wspólne dla mikrokontrolerów 8xC52 i 80C515/535. Pozostałe źródła przerwań wraz z ich adresami obsługi występują w procesorze 80C515/535. W pozostałych procesorach tej rodziny, ponieważ struktura wewnętrzna jest inna, mogą występować inne źródła przerwań wraz z przypisanymi do nich adresami. Niezmienna pozostaje tylko zasada, że odstępy pomiędzy kolejnymi adresami obsługi przerwań są nie mniejsze niż 8 bajtów.

Takie rozmieszczenie adresów obsługi przerwań w mikrokontrolerach rodziny '51 powoduje, że dla ominięcia obszaru pamięci programu związanego z obsługą przerwań, pod adresem 0 programu umieszcza się rozkaz skoku do programu głównego, a pod adresami obsługi przerwań - najczęściej tylko rozkaz skoku do podprogramów obsługi przerwań, rysunek 8-1.



Rys. 8-1. Ilustracja działania przerwania

Jeżeli w trakcie wykonywania programu przyjdzie sygnał przerwania i zostanie on przyjęty przez układ przerwań, to wykonywanie programu zostanie zawieszone i nastąpi skok pod adres związany z występującym przerwaniem. Umieszczony tu rozkaz skoku powoduje skok do podprogramu obsługi przerwania. Po jego wykonaniu następuje powrót do przerwanego programu (rozkaz RETI), w miejsce gdzie nastąpiło przerwanie.

Każdy sygnał, który może wywołać przerwanie jest związany, w mikrokontrolerach rodziny '51, z odpowiadającym mu znacznikiem. W momencie, gdy sygnał przerwania jest aktywny, odpowiadający mu znacznik jest ustawiany w stan 1. Jeżeli dane przerwanie jest odblokowane, to nastąpi przejście do procedury obsługi tego przerwania. Efekt przerwania można również wywołać poprzez programowe ustawienie znacznika w stan 1.

Znaczniki te są umieszczone w różnych rejestrach.

Mikrokontrolery 8xC51/52 i 80C515/535:

rejestr TCON — adres 88h

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

rw = 0

- TF1 - znacznik przerwania od licznika T1
- TF0 - znacznik przerwania od licznika T0
- IE1 - znacznik przerwania zewnętrznego INT1
- IE0 - znacznik przerwania zewnętrznego INT0

rejestr SCON — adres 98h

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

rw = 0

- TI - znacznik przerwania od nadajnika portu szeregowego
- RI - znacznik przerwania od odbiornika portu szeregowego

Mikrokontroler 80C515/535:

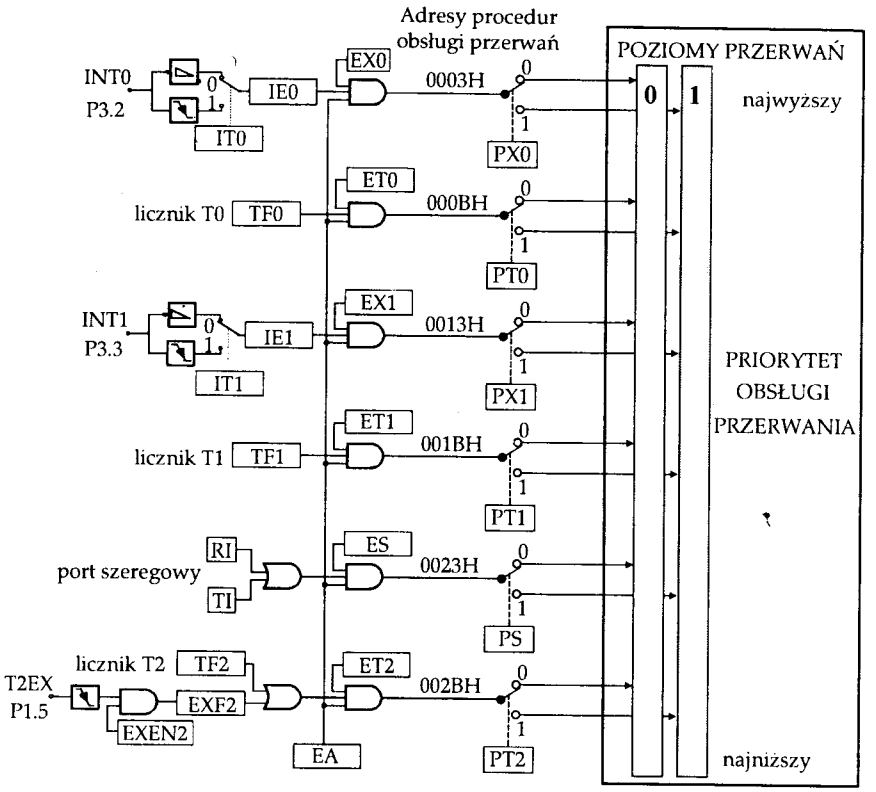
rejestr IRCON — adres 0C0h

EXF2	TF2	IEX6	IEX5	IEX4	IEX3	IEX2	IADC

rw = 0

- EXF2 - znacznik przerwania od zewnętrznego sygnału przeładowania licznika T2
- TF2 - znacznik przerwania od licznika T2
- IEX6 ÷ IEX2 - znaczniki zewnętrznych przerwań INT6 ÷ INT2.
- IADC - znacznik przerwania od przetwornika A/C

W mikrokontrolerach rodziny '51 część znaczników jest ustawiana w stan zera automatycznie po wejściu procesora do obsługi przerwania. Pozostałe znaczniki muszą być zerowane programowo. Do znaczników zerowanych automatycznie należą znaczniki TF0, TF1 oraz IE0 i IE1 jeżeli układ przerwań zewnętrznych reaguje na zbocze opadające sygnału wywołującego przerwanie.
Jeżeli układ przerwań zewnętrznych reaguje na poziom, a na wejściu jest wymuszany stan zera, to programowe zerowanie znaczników nie da żadnego efektu.



Rys. 8-2. System przerwań mikrokontrolera 80C51/52

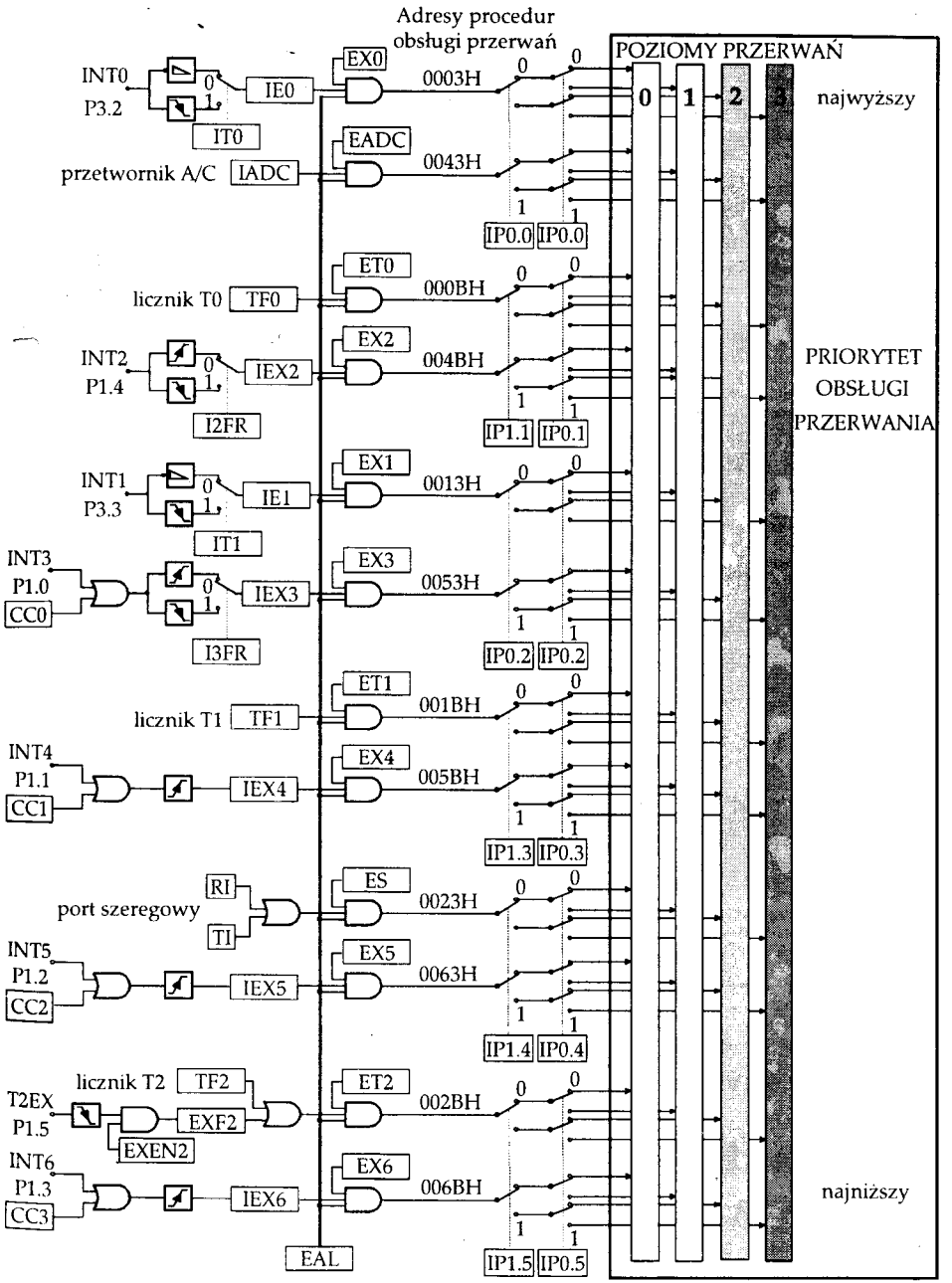
Na rysunku 8-2 jest przedstawiony schemat układu przerwań dla mikrokontrolerów 80C51/52, a na rysunku 8-3 - schemat dla mikrokontrolera 80C515. Na rysunkach są pokazane źródła sygnałów przerwań, znaczniki obsługujące przerwania oraz adresy wektorów obsługi przerwań.

Układy przerwań zewnętrznych INT0 i INT1 mogą reagować albo na poziom sygnału (poziom 0), albo na zbocze opadające sygnału. Zależy to od ustawienia znaczników IT0 i IT1 umieszczonych w rejestrze TCON. Jeżeli te znaczniki zostaną wyzerowane, to układ przerwań będzie reagował na poziom zera logicznego.

Układy przerwań zewnętrznych INT2 i INT3 mogą reagować albo na zbocze narastające albo na zbocze opadające sygnału zewnętrznego, w zależności od stanu znaczników I2FR i I3FR umieszczonych w rejestrze T2CON. Jeżeli znaczniki te są w stanie 1, to układ reaguje na zbocze narastające. W przeciwnym przypadku - na zbocze opadające.

rejestr T2CON — adres 0C8h, rw = 0

T2PS	I3FR	I2FR	T2R1	T2R0	T2CM	T2I1	T2I0



Rys. 8-3. Schemat układu przerwań mikrokontrolera 80C515.

Wykrywanie przez procesor zmiany poziomu sygnału na wejściu przerwań zewnętrznych polega na testowaniu stanu wejścia w każdym cyklu maszynowym. Jeżeli w jednym cyklu maszynowym występuje jeden stan, a w

ziomu. Dlatego, by układ poprawnie zareagował na sygnał przerwania, zarówno stan wysoki jak i niski powinny trwać przynajmniej przez jeden cykl maszynowy.

> Przy przerwaniach zewnętrznych reagujących na poziom, przed zakończeniem podprogramu obsługi przerwania, sygnał wejściowy powinien przejść do poziomu wysokiego lub należy zablokować to przerwanie, gdyż w przeciwny przypadku nastąpi ponowne wejście do obsługi tego przerwania.

W mikrokontrolerach rodziny '51 występuje jeden znacznik, którym można zablokować wszystkie przerwania równocześnie i znaczniki blokujące indywidualnie każde źródło przerwania. W zależności od liczby źródeł przerwań, znaczniki te mogą mieścić się w jednym lub dwóch rejestrach.

> W różnych mikrokontrolerach rodziny '51 rejestry zawierające te same znaczniki i mieszczące się pod tymi samymi adresami mogą mieć różne nazwy. Podobnie znaczniki pełniące tę sama funkcję mogą mieć również inną nazwę

Poniżej są podane rejestry i rozmieszczone w nich znaczniki blokujące przerwania dla mikrokontrolerów 8xC51/51 i 80C515/535.

> Ustawienie znaczników w stan zera powoduje zablokowanie przerwania. Przychodzące sygnały przerwań nie są pamiętane.

8xC51/52:

rejestr IE — adres 0A8h

EA	X	ET2	ES	ET1	EX1	ET0	EX0

rw = 0

80C515/535:

rejestr IEN0 — adres 0A8h

EAL	WDT	ET2	ES	ET1	EX1	ET0	EX0

rw = 0

rejestr IEN1 — adres 0B8h

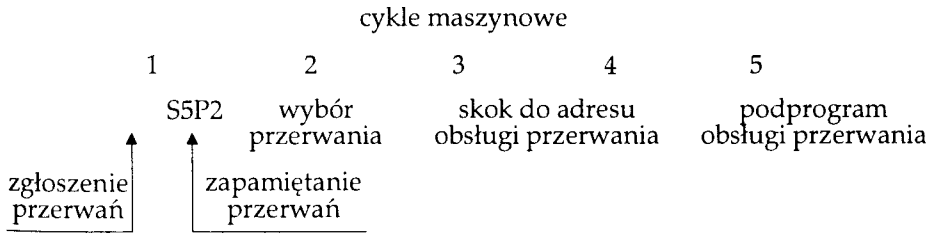
EXEN2	SWDT	EX6	EX5	EX4	EX3	EX2	EADC

rw = 0

- EA, EAL - znaczniki blokujące wszystkie przerwania.
- ET2 - znacznik blokujący przerwanie od licznika T2
- ES - znacznik blokujący przerwania od nadajnika i odbiornika portu szeregowego
- ET1 - znacznik blokujący przerwanie od licznika T1
- EX1 - znacznik blokujący zewnętrzne przerwanie INT1

- ET0 - znacznik blokujący przerwanie od licznika T0
- EX0 - znacznik blokujący zewnętrzne przerwanie INT0
- EXEN2 - znacznik blokujący przerwanie od zewnętrznego sygnału przeładowania licznika T2.
- EX6 ÷ EX4 - znaczniki blokujące zewnętrzne przerwania INT6 ÷ INT4 oraz przerwania od komparatorów CC3 ÷ CC1.
- EX3 - znacznik blokujący zewnętrzne przerwanie INT3 oraz przerwanie od komparatora CRC.
- EX2 - znacznik blokujący zewnętrzne przerwanie INT2.
- EADC - znacznik blokujący przerwanie od przetwornika A/C.

Stan znaczników wywołujących przerwania jest sprawdzany w fazie S5P2 każdego cyklu maszynowego, rysunek 8-4. W następnym cyklu maszynowym po wykryciu sygnału przerwania następuje rozpoznanie źródła przerwania oraz ustalenie kolejności obsługi przerwania, jeśli wystąpiło kilka sygnałów przerwań równocześnie. W dwóch kolejnych cyklach maszynowych jest zapamiętywana na stosie zawartość licznika rozkazów, do licznika rozkazów jest wpisywany adres przerwania i jest ewentualnie zerowany znacznik wywołujący przerwanie, jeżeli jest to znacznik zerowany sprzętowo.



Rys. 8-4. Wybór przerwania i wywołanie podprogramu jego obsługi.

Wejście do procedury obsługi przerwania może być opóźnione jeżeli:

1. jest obsługiwane przerwanie, które nie może być przerwane. W tej sytuacji musi być ono najpierw zakończone, potem jest wykonywany jeden rozkaz z przerwanego programu i dopiero wtedy może być obsłużone kolejne przerwanie.
2. nie został zakończony aktualnie wykonywany rozkaz. Najpierw musi być zakończony rozkaz, by procesor przeszedł do obsługi przerwania.

Z powyższego wynika wniosek, że czas od momentu nadejścia sygnału przerwania do momentu wejścia do procedury obsługi przerwania może być różny.

Mikrokontroler nie obsługuje sygnału przerwania, który przyszedł gdy przerwanie było zablokowane.

Jeżeli do trzeciego cyklu maszynowego procedury przerwania nadejdzie następny sygnał przerwania o wyższym poziomie niż aktualny, to zostanie on obsłużony w pierwszej kolejności. Po wejściu do procedury obsługi przerwania system przerwań zostaje zablokowany dla innych przerwań, chyba że nadejdzie przerwanie o wyższym priorytecie. Odblokowanie systemu następuje po rozkazie RETI. Rozkaz ten ponadto powoduje przepisanie dwóch bajtów ze szczytu stosu do rejestru rozkazów PC.

Zakończenie podprogramu obsługi przerwania rozkazem RET również odtworzy zawartość rejestru rozkazów, czyli nastąpi powrót do wykonywania przerwanego programu, ale nie nastąpi odblokowanie systemu przerwań dla następnych przerwań !

Gdy równocześnie nadejdzie kilka sygnałów przerwań żądających obsługi, to o kolejności ich obsłużenia decyduje priorytet ustalony arbitralnie przez producenta. Oznacza to, że przerwanie o wyższym priorytecie zostanie obsłużone przed przerwaniem o niższym priorytecie. W tabeli 8-2 zostały uszeregowane według priorytetu źródła przerwań dla mikrokontrolerów 80C515/535. Źródła przerwań dla mikrokontrolerów 8xC51/52 mają taki sam rozkład priorytetów.

Tabela 8-2. Uszeregowanie źródeł przerwań według priorytetów.

Lp.	Źródło przerwania	Priorytet
1.	IE0	najwyższy
2.	IADC	
3.	TF0	
4.	IEX2	
5.	IE1	
6.	IEX3	
7.	TF1	
8.	IEX4	
9.	RI + TI	
10.	IEX5	
11.	TF2 + EXF2	
12.	IEX6	najniższy

W mikrokontrolerach rodziny '51 wprowadzono mechanizm umożliwiający nadanie wybranym przerwaniom bezwzględnego pierwszeństwa, tzn. mają one możliwość przerwania wykonywanego podprogramu obsługi innego przerwania. Przerwaniom można nadać odpowiedni poziom. Jeżeli w trakcie obsługi przerwania nadejdzie sygnał przerwania o wyższym poziomie , to procesor

przerwie wykonywanie przerwania z poziomu niższego, wykona obsługę przerwania z poziomu wyższego i ponownie przejdzie do wykonywania obsługi przerwanego przerwania.

Do ustalania poziomów przerwań służą rejestry poziomu przerwań. W zależności od typu mikrokontrolera mogą występować jeden lub dwa rejestry.

W mikrokontrolerach 8xC51/52 występuje rejestr IP:

rejestr IP — adres 0B8h

X	X	PT2	PS	PT1	PX1	PT0	PX0

rw = 0

- X - zarezerwowane
- PT2 - licznik T2 (mikrokontroler 8xC52)
- PS - port szeregowy
- PT1 - licznik T1
- PX1 - przerwanie zewnętrzne INT1
- PT0 - licznik T0
- PX0 - przerwanie zewnętrzne INT0

Należy zwrócić uwagę, że w mikrokontrolerach 8xC51/51 rejestr IP znajduje się pod adresem, pod którym w mikrokontrolerach 80C515/535 znajduje się rejestr IEN1.

Ustawienie w rejestrze IP w stan 1 znacznika dla wybranego źródła przerwania powoduje, że przerwanie to osiąga wyższy poziom od przerwań, których znaczniki mają stan 0. W ten sposób wyróżnione przerwanie będzie obsłużone jako pierwsze, jeśli zgłosi się kilka przerwań, lub spowoduje przerwanie obsługiwanego przerwania i zostanie samo obsłużone. Jeśli kilka źródeł przerwań zostanie wyróżnionych wyższym poziomem, to, gdy zgłoszą się równocześnie, o kolejności ich obsłużenia zadecyduje priorytet, według podanej wyżej tabeli.

Ponieważ okazało się że dwa poziomy przerwań nie wystarczają do ustawienia hierarchii przerwań, w mikrokontrolerach 80C515/535 wprowadzono cztery poziomy ustalane przez ustawianie znaczników w dwóch rejestrach:

rejestr IP0 — adres 0A9h

X	WDTS	IP0.5	IP0.4	IP0.3	IP0.2	IP0.1	IP0.0

rw = 0

rejestr IP1 — adres 0B9h

X	X	IP1.5	IP1.4	IP1.3	IP1.2	IP1.1	IP1.0

rw = 0

Poziomy przerwań podaje tabela 8-3.

Tabela 8-3. Poziomy przerwań w mikrokontrolerach 80C515/535

Znaczniki		Poziom
IP1.x	IP0.x	przerwania
0	0	0
0	1	1
1	0	2
1	1	3

Ponieważ liczba znaczników jaka jest do dyspozycji jest za mała, by ustawić poziomy przerwań dla wszystkich źródeł, dlatego źródła przerwań zostały połączone w pary i przypisane do odpowiednich znaczników. Przedstawia to tabela 8-4.

Para źródeł przerwań znajduje się zawsze na tym samym poziomie i nie może wzajemnie przerywać podprogramu obsługi przerwań. Na przykład sygnał przerwania od przetwornika A/C (IADC) nie może przerwać podprogramu obsługi przerwania zewnętrznego INT0 i odwrotnie.

Tabela 8-4. Powiązanie źródeł przerwań ze znacznikami.

Znaczniki	Źródła przerwań	
IP1.0, IP0.0	IE0	IADC
IP1.1, IP0.1	TF0	IEX2
IP1.2, IP0.2	IE1	IEX3
IP1.3, IP0.3	TF1	IEX4
IP1.4, IP0.4	RI+TI	IEX5
IP1.5, IP0.5	TF2+EXF2	IEX6

Poniższy przykład ilustruje sposób wejścia do programu głównego i podprogramu obsługi przerwania zewnętrznego INT0.

```
;*********************************************************************
   ORG 0                    ;adres 0
   LJMP   PROG_GL           ;skok do programu głównego

   ORG 3                    ;adres 3, przerwanie INT0
   LJMP   PP_INT0           ;skok do podprogramu obsługi przerwania ze-
          ;wnętrznego INT0

PP_INT0:                    ;podprogram obsługi przerwania INT0
   PUSH   ACC               ;schowanie na stos akumulator
   PUSH   PSW               ;rejestru statusowego
   PUSH   DPL               ;rejestru
   PUSH   DPH               ;DPTR
   .....
```



```
   CLR    ET0               ;zerowanie znacznika przerwania. Przerwanie
                            ;jest ustawione na poziom
   POP    DPH               ;odtworzenie rejestru
   POP    DPL               ;DPTR
   POP    PSW               ;PSW
   POP    ACC               ;akumulatora
   RETI                     ;powrót z podprogramu

PROG_GL:                    ;program główny
   ORL    IE,#81h           ;ustawienie znacznika EA przerwania ogólnego
                            ;i znacznika EX0 przerwania zewnętrznego
   CLR    IT0               ;układ przerwania będzie reagował na poziom
                            ;sygnału na wejściu INT0. Rozkaz może być
                            ;pominięty, ponieważ zerowanie procesora
                            ;powoduje również wyzerowanie tego
                            ;znacznika.
   ;ciąg dalszy programu
   ;głównego
```

Pytania i problemy

1. Na czym polega działanie przerwania ?
2. Jak są rozmieszczone adresy przerwań w mikrokontrolerach rodziny '51?
3. Jakie znaczniki w rodzinie mikrokontrolerów '51 są automatycznie zerowane po wejściu do podprogramu obsługi przerwania ?
4. Ile cykli maszynowych może wykonać procesor w rodzinie '51 od momentu przyjęcia sygnału przerwania do momentu wejścia do procedury obsługi przerwania ?
5. Jaka jest różnica w działaniu mikrokontrolera rodziny '51 gdy podprogram obsługi przerwania zostanie zakończony rozkazem RET i RETI ?
6. W jaki sposób jest określana kolejność obsługi przerwania przy nadejściu kilku sygnałów przerwania jednocześnie ?
7. Z czym są związane poziomy przerwań ?

9. Liczniki

Liczniki służą do zliczania impulsów, pomiarów odcinków czasu, generowania sygnałów o określonym czasie trwania itp. Dlatego często znajdują się w strukturze wewnętrznej mikrokontrolerów. Ich budowa, liczba bitów, sposób sterowania, wpisywania i odczytywania zależą od typu mikrokontrolera. Również w rodzinie mikrokontrolerów '51 występują liczniki o różnej budowie i właściwościach, ale dwa podstawowe liczniki oznaczone jako T0 i T1 występują we wszystkich elementach tej rodziny w prawie nie zmienionej formie.

Są to liczniki 16-bitowe, które składają się z dwóch rejestrów ośmiobitowych oznaczonych odpowiednio dla licznika T0 - TH0 i TL0 oraz TH1 i TL1 dla licznika T1 (H oznacza część bardziej znaczącą, a L - mniej znaczącą). Liczniki te mogą pracować jako liczniki zliczające impulsy zewnętrzne (counter) lub impulsy z wewnętrznego zegara (timer). Liczą one „w przód", tzn. impuls wejściowy zwiększa stan liczników.

Licznik T0 ma cztery tryby pracy, licznik T1 - trzy, takie same jak licznik T0. Do obsługi liczników są przeznaczone rejestry TCON i TMOD

rejestr TCON — adres 88h

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	rw = 0
licznik T1		licznik T0						

Znaczniki TF0 iTF1 są ustawiane w stan 1 sprzętowo, gdy nastąpi przepełnienie odpowiedniego licznika T0 lub T1. Ustawienie ich w stan 1, również programowe, wywoła przerwanie, jeżeli będzie ono odblokowane. Mogą być zerowane programowo lub są zerowane automatycznie po wejściu mikrokontrolera do obsługi przerwania wywołanego ustawieniem znacznika w stan 1.

Znaczniki TR0 i TR1 sterują pracą liczników. Ustawienie ich w stan 1 odblokowuje liczniki i mogą one zliczać impulsy wejściowe. Pozostałe znaczniki są związane z przerwaniami zewnętrznymi.

Każdy ze znaczników rejestru TCON może być zmieniany indywidualnie przez program.

Tryby pracy liczników T0 i T1, źródło sygnałów wejściowych oraz sposób sterowania licznikiem ustala się przez wpisanie do rejestru TMOD odpowiedniego słowa. Rejestr TMOD znajduje się w obszarze SFR pod adresem niepodzielnym przez 8 i dlatego zmianę stanu jego znaczników należy dokonywać poprzez wpis całego bajtu.

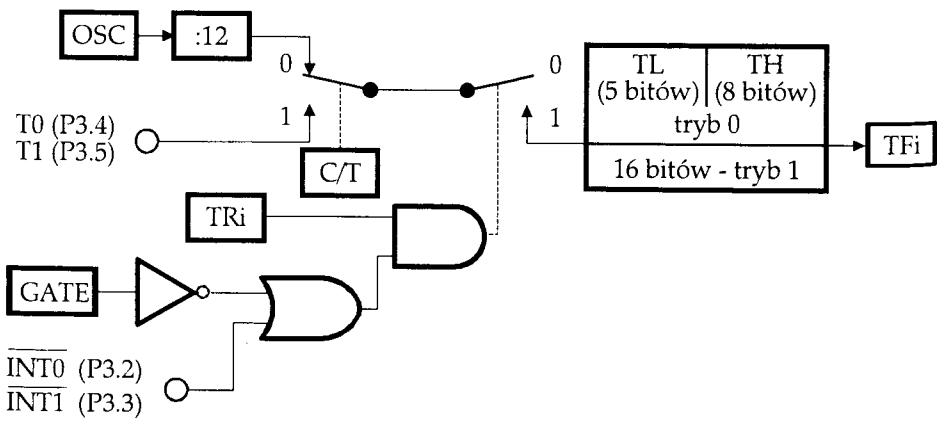
rejestr adres 89h

TMOD	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0	rw = 0
	licznik T1				licznik T0				

Obydwa liczniki mają takie same układy współpracujące, rysunek 9-1:

- przełącznik doprowadzający sygnał wejściowy do liczników albo z wewnętrznego oscylatora poprzez dzielnik :12, albo z zewnątrz mikrokontrolera poprzez końcówki T0 (P3.4) i T1 (P3.5) portu P3. Przełącznik jest przełączany znacznikiem C/$\overline{T}$. Gdy C/$\overline{T}$ = 0, to taktowanie liczników odbywa się z układu wewnętrznego, a gdy C/$\overline{T}$ = 1, to są zliczane impulsy zewnętrzne.
- zespół bramek blokujących pracę liczników. Znacznik TRi służy do programowego uruchamiania pracy licznika. Przy TRi = 0 licznik jest zatrzymany, a przy TRi = 1 - pracuje. Praca licznika może być również uruchamiana sygnałem zewnętrznym. Przy ustawieniu znacznika GATE = 1 licznik pracuje gdy wejście INTi (P3.2 lub P3.3) jest na wysokim poziomie, jest natomiast zatrzymana, gdy wejście to jest na poziomie niskim. Gdy znacznik GATE = 0 praca licznika nie zależy od stanu wejścia INTi.
- przerzutniki przepełnienia liczników TFi, umieszczone w rejestrze TCON. Przepełnienie licznika Ti (przejście ze stanu 1 do stanu 0 na wszystkich bitach licznika) powoduje wpisanie stanu 1 do przerzutnika TFi. Jeżeli przerwanie od licznika jest aktywne, to nastąpi wywołanie procedury obsługi tego przerwania.

gdzie i = 0 lub 1.



Rys. 9-1. Schemat układów liczników T0 i T1 pracujących w trybie 0 i 1.

Wejścia bramkujące liczników są wspólne z wejściami przerwań zewnętrznych dzięki czemu zewnętrzny sygnał zamykający bramkę licznika może uruchomić system przerwań.

Tryb pracy liczników wybiera się poprzez odpowiednie ustawienie znaczników M0 i M1:

- Tryb 0 M1 = 0, M0 = 0

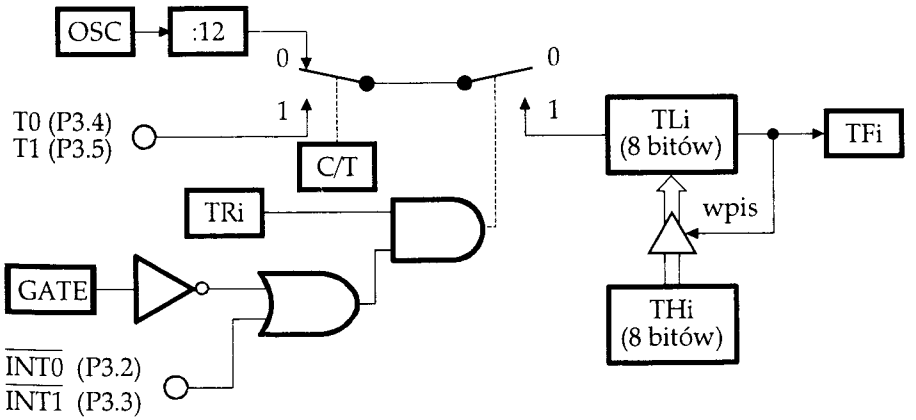
  Tryb 0 został wprowadzony dla zachowania kompatybilności z rodziną mikrokontrolerów MCS-48, pierwowzorem rodziny '51. W trybie tym rejestr TL pracuje w układzie dzielnika wstępnego o podziale przez 2^5 lub 5-bitowego licznika, a rejestr TH jako licznik lub dzielnik 8-bitowy, rysunek 9-1. Oba rejestry są połączone kaskadowo.

- Tryb 1 M1 = 0, M0 = 1

  Tryb 1, rysunek 9-1, jest taki sam jak tryb 0 z tą różnicą, że rejestr TL pracuje jako licznik lub dzielnik 8-bitowy. Obydwa rejestry są połączone kaskadowo tworząc 16-bitowy dzielnik lub licznik.

- Tryb 2 M1 = 1, M0 = 0

  W trybie 2 są połączone w konfiguracji dzielnika 8-bitowego z autoładowaniem, tzn. z zadawaniem stanu początkowego po przepełnieniu rejestru, rysunek 9-2.



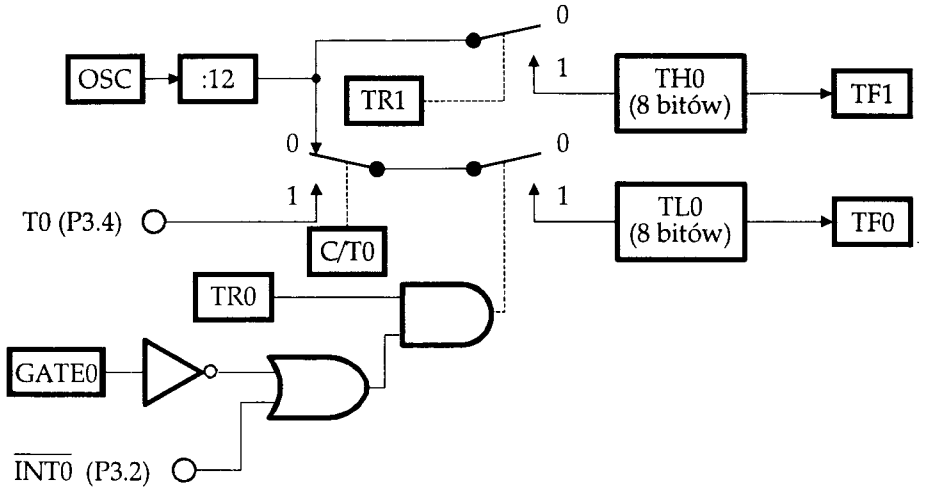
Rys. 9-2 Układ połączeń liczników T0 i T1 pracujących w trybie 2.

Rejestr TL pracuje jako dzielnik właściwy, natomiast rejestr TH służy jako bufor, którego zawartość jest przepisywana do rejestru TL gdy nastąpi jego przepełnienie. Operacja ta umożliwia otrzymanie dzielnika o zmiennym podziale, przy czym należy pamiętać, że liczniki liczą „w przód" i dlatego do bufora trzeba wpisywać uzupełnienie do 0FFh

liczby podziałowej. Ponadto, po uruchomieniu dzielnik może zawierać przypadkową wartość początkową, lub 0 po włączeniu zasilania lub zerowaniu procesora. Dlatego przed uruchomieniem dzielnika do rejestru TL należy wpisać taką samą wartość jaka jest wpisywana do rejestru TH. Licznik T1 pracujący w trybie 2 jest często stosowany do taktowania portu szeregowego mikrokontrolera.

- Tryb 3 M1 = 1, M0 = 1

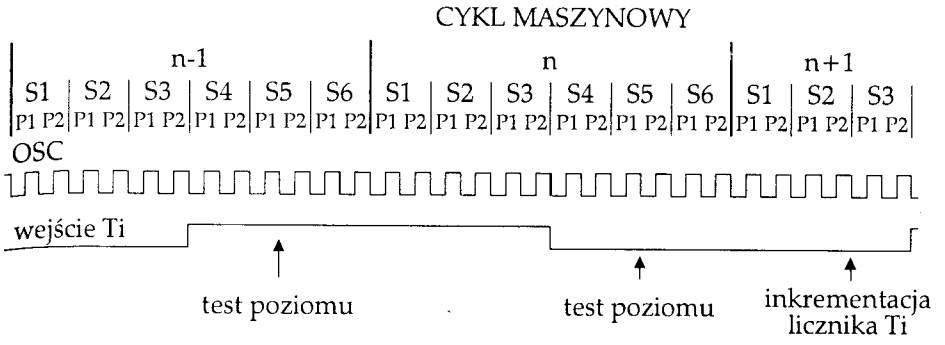
Tryb 3 występuje tylko dla licznika T0. W tym trybie rejestry TL0 i TH0 pracują jako dwa niezależne 8-bitowe liczniki/dzielniki, rysunek 9-3.



Rys. 9-3. Układ połączeń licznika T0 pracującego w trybie 3.

Rejestr TL0 pracuje w strukturze licznika T0 (znaczniki TR0, GATE, C/$\overline{T}$ oraz przerzutnik TF0). Może więc być sterowany i testowany jak licznik T0 w trybie 0 i 1. Natomiast rejestr TH0 jest połączony na stałe z wyjściem zegara wewnętrznego poprzez dzielnik :12 i jest bramkowany znacznikiem TR1. Wyjście tego rejestru jest połączone z przerzutnikiem TF1. Dlatego gdy licznik T0 pracuje w trybie 3, to licznik T1 może pracować w pozostałych trybach ale bez możliwości bramkowania jego wejścia i testowania przepełnienia. W tej sytuacji licznik T1 nadaje się praktycznie tylko do taktowania portu szeregowego. Wprowadzenie trybu 3 dla licznika T1 powoduje jego zatrzymanie.

Przy zliczaniu impulsów wewnętrznych (C/$\overline{T}$ = 0) rejestr licznika jest inkrementowany co jeden cykl maszynowy. Odpowiada to 1/12 częstotliwości oscylatora. Natomiast przy zliczaniu impulsów zewnętrznych (C/$\overline{T}$ = 1) stan odpowiedniego wejścia licznika jest taktowany podczas każdego cyklu maszynowego, rysunek 9-4.



Rys. 9-4 Testowanie poziomów sygnałów wejściowych liczników.

Jeżeli test wykaże stan wysoki wejścia w jednym cyklu maszynowym oraz stan niski w następnym cyklu maszynowym, to w kolejnym cyklu maszynowym nastąpi inkrementacja licznika. Dlatego maksymalna częstotliwość impulsów zewnętrznych zliczanych przez liczniki T0 i T1 musi być mniejsza od 1/24 częstotliwości oscylatora. Natomiast częstotliwość minimalna nie jest niczym ograniczona.

Poniższy program pokazuje jak wykorzystać licznik T0 do pomiaru czasu. Mierzony sygnał jest doprowadzony do wejścia INT0 (P3.2). Zbocze narastające tego sygnału otwiera bramkę, a zbocze opadające zamyka bramkę dla impulsów wewnętrznych zliczanych w liczniku. Przy częstotliwości oscylatora wynoszącej 12 MHz jeden impuls zliczony w liczniku odpowiada czasowi 1 µs. Występujące przepełnienia licznika, wywołujące przerwania, są zliczane w rejestrze B. W ten sposób otrzymuje się licznik 24 bitowy. Uruchomienie procedury pomiaru odbywa się przez ustawienie stanu 0 na wejściu INT1 (P3.3).

```
;*****************************************************************
; Program pomiaru czasu
; Wykorzystane rejestry:
;       B, licznik T0

NAME  Pomiar czasu

ORG   0
      LJMP   PROG_CZAS      ;ominięcie obszaru obsługi przerwań

ORG   0Bh                   ;podprogram obsługi przerwania od licznika T0
      INC    B              ;zwiększenie zawartości rejestru B o jeden
      RETI                  ;powrót z podprogramu

PROG_CZAS:
      ORL    TMOD,#9        ;licznik T0 - tryb 1, bramkowanie zewnętrzne
                            ;taktowanie wewnętrzne
      ORL    IE,#82h        ;odblokowanie przerwania od licznika T0
```

```
POM:                    ;procedura pomiarowa
    MOV  R0,#20h        ;adres pamięci RAM, pod którym będzie
                        ;przechowany wynik pomiaru
    CLR  A
    MOV  TL0,A          ;zerowanie licznika T0
    MOV  TH0,A          ;i
    MOV  B,A            ;rejestru B
    JB   P3.2,$         ;test stanu linii P3.2. Rozkaz jest powtarzany aż
                        ;stan linii będzie 0
    SETB TR0            ;odblokowanie licznika - licznik zacznie zliczać
                        ;gdy na wejściu P3.2 pojawi się stan 1
    JB   P3.2,$         ;oczekiwanie na zbocze opadające mierzonego
                        ;sygnału
    CLR  TR0            ;zablokowanie licznika T0
    MOV  @R0,TL0        ;przesłanie
    INC  R0             ;wyniku pomiaru
    MOV  @R0,TH0        ;do pamięci
    INC  R0             ;RAM
    MOV  @R0,B
    LJMP POM            ;skok etykiety POM - ponowne wykonanie
                        ;pomiaru
END
```

? Pytania i problemy

1. Jak są zbudowane liczniki T0 i T1 w mikrokontrolerach rodziny '51 ?
2. Jak pracują liczniki T0 i T1 w poszczególnych trybach pracy ?
3. Od czego zależy graniczna częstotliwość sygnałów wejściowych dla liczników T0 i T1 ?
4. Napisać program pomiaru czasu, w którym stany przepełnienia licznika T0 będą zliczane w liczniku T1.
5. Zmodyfikować powyższy program tak, by koniec pomiaru był wykrywany przez przerwanie.

10. Arytmetyka mikrokontrolerów

W systemach mikroprocesorowych i komputerowych podstawowym kodem zapisu liczb jest kod dwójkowy (binarny), wynikający z dwustanowej pracy wszystkich układów tworzących te systemy. Ponownie powraca problem rozumienia danych w sensie mikrokontrolera i człowieka, który myśląc o liczbach prawie zawsze myśli o liczbach dziesiętnych. Systemów zapisu liczb może być bardzo wiele. Dowolną, dodatnią liczbę całkowitą można zapisać w postaci sumy iloczynów:

$$\text{Liczba} = a_k * p^k + a_{k-1} * p^{k-1} + a_{k-2} * p^{k-2} + \ldots + a_2 * p^2 + a_1 * p^1 + a_0 * p^0$$

gdzie: p - podstawa systemu zapisu, np.: 2 (dwójkowy), 8 (ósemkowy), 10 (dziesiętny), 16 (szesnastkowy) itd.
a_k - wartości kolejnych pozycji, $a_k = 0 .. (p-1)$.
k - numery kolejnych pozycji, k = 0, 1, 2, ...,

W stosowanym powszechnie systemie dziesiętnym podstawa systemu p=10 i wartości kolejnych pozycji mieszczą się w zakresie a_k=0..9. Zapis dziesiętny liczby 1234 można przestawić jako:

$$1234 = 1 * 10^3 + 2 * 10^2 + 3 * 10^1 + 4 * 10^0$$

W systemie dwójkowym (binarnym) podstawa systemu p=2 powoduje, że wartości kolejnych pozycji mogą być równe: 0 lub 1. Liczba 1234 zapisana w kodzie dwójkowym ma postać:

$$1234d = 0 * 2^{11} + 1 * 2^{10} + 0 * 2^9 + 0 * 2^8 + 1 * 2^7 + 1 * 2^6 + 0 * 2^5 + 1 * 2^4 + 0 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 = 0100\ 1101\ 0010b$$

Ze względu na możliwość pomyłki w interpretacji ciągu zero-jedynkowego uproszczenia zapisu i układów dekodujących stosowany jest także kod szesnastkowy oraz kod binarno-dziesiętny BCD (Binarny Coded Decimal). W kodzie szesnastkowym (podstawa systemu p=16) stosowanych jest do zapisu liczb 10 cyfr (0..9) i 6 liter (A..F). Litery symbolizują kolejne wartości z przedziału: 10 (A), 11 (B), .. , 15 (F). Dziesiętna liczba 1234 zapisywana jest w postaci:

$$1234d = 4 * 16^2 + 13 * 16^1 + 2 * 16^0 = 4D2h$$

Przy zapisie liczb w kodzie BCD porcjom czterech kolejnych bitów przyporządkowuje się wartości z przedziału 0 .. 9. Dla kodu BCD typu 8421 przedstawiona liczba ma postać:

$$1234d = 0001\ 0010\ 0011\ 0100\ BCD$$

Z przytoczonych form zapisów liczb można wysnuć następujące wnioski:

- podając wartość liczby należy podać podstawę systemu w postaci pojedynczej litery: dla systemu binarnego B lub b, dla systemu ósemkowego Q lub q lub O lub o, dla systemu dziesiętnego D lub d, a dla systemu szesnastkowego H lub h; w systemie dziesiętnym można pominąć podstawę systemu (podstawa p=10 jest domyślna),
- w systemie szesnastkowym oprócz cyfr pojawia się 6 dodatkowych liter A..F lub a..f dla oznaczenia wartości z przedziału 10..15,
- w kodzie binarno-dziesiętnym BCD stosowane są tylko cyfry z przedziału 0..9.

Interpretacja liczb ujemnych zależy od sposobu zapisu znaku i wartości liczby, np. znak-moduł, kod uzupełnienia do dwóch itp. [6]. W mikroprocesorach przyjęto do zapisu N-bajtowych liczb ujemnych metodę kodu uzupełnienia do 2^N. W szczególności liczba ujemna 1-bitowa jest reprezentowana przez uzupełnienie do 2, stąd nazwa kodu. W metodzie tej najbardziej znaczący bit niesie informację o znaku liczby, a pozostałe o jej wartości w kodzie uzupełnienia do dwóch. Jeśli rozpatruje się liczby 8-bitowe to bit b_7 jest bitem znakowym, a bity $b_{6..0}$ są bitami wartości:

+ 3 = 0000 0011b, bit b_7 = 0, liczba dodatnia
− 3 = 1111 1101b, bit b_7 = 1, liczba ujemna

Zamiana liczby dodatniej na ujemną i odwrotnie (negacja arytmetyczna) dokonywana jest przez negację logiczną wszystkich bitów liczby, a następnie na dodaniu 1. Zamiana liczby +53 na liczbę ujemną realizowana jest następująco:

+53d = 35h =	0011	0101b
negacja logiczna:	1100	1010b
+1		1
−53d = 0CBh =	1100	1011b

W praktyce negację arytmetyczną liczby w kodzie uzupełnienia do 2 wykonuje się przez negację logiczną wszystkich bitów po pierwszym bicie niezerowym:

- testowanie bitów rozpoczyna się od bitu najmniej znaczącego (LSB),
- pozostawia się niezmienione wszystkie zera oraz pierwszą napotkaną jedynkę,
- neguje się logicznie wszystkie pozostałe bity.

W systemach mikroprocesorowych do zapisu porcji informacji oprócz bitów stosowane są bajty i słowa (zmienne 16-bitowe). Zakres zmiennych binarnych zależy od liczby bajtów wykorzystanych do zapisu liczby:

zmienna typu:	liczby całkowite bez znaku	liczby całkowite ze znakiem
bajt	0 .. 255	-128 .. +127
słowo	0 .. 65.535	-32.768 .. +32.767

10.1 Instrukcje arytmetyczne - rejestr statusowy PSW

Arytmetyka mikrokontrolerów jest arytmetyką binarną z jednym wyjątkiem, który dotyczy dodawania liczb w kodzie BCD. Zasadnicza grupa instrukcji arytmetycznych związana jest z akumulatorem A. Jeden z argumentów i wynik operacji zawarty jest zawsze w akumulatorze A, z wyjątkiem instrukcji inkrementacji (zwiększania o jeden) i dekrementacji (zmniejszania o jeden) zawartości rejestrów Rn, rejestrów specjalnych SFR oraz komórek wewnętrznej pamięci RAM. Wyjątkiem są również instrukcje mnożenia i dzielenia, z którymi związany jest rejestr B.

Związanie większości instrukcji arytmetycznych z akumulatorem A powoduje, że akumulator jest najbardziej obciążonym rejestrem mikrokontrolera. Również czas wykonania programu, który odwołuje się co chwilę do akumulatora jest długi.

W rejestrze słowa statusowego PSW znajdują się znaczniki (C, AC, OV i P), które zmieniane są w trakcie wykonywania instrukcji arytmetycznych. Przeznaczenie każdego z nich jest inne:

rejestr PSW	CY	AC	F0	RS1	RS0	OV	F1	P	adres 0D0h

- **znacznik przeniesienia C** sygnalizuje przekroczenie zakresu (0..0FFh) 8-bitowych liczb całkowitych bez znaku w operacji dodawania i odejmowania oraz przekroczenie zakresu 0..99d przy operacji korekcji dziesiętnej. Oznacza przeniesienie między kolejnymi dodawanymi bajtami lub pożyczkę przy wielobajtowym odejmowaniu.

Jeśli wykonane zostanie dodawanie dwóch zmiennych 1-bajtowych (7Bh i 93h) oraz obie zmienne potraktować jako liczby całkowite bez znaku, to wynik dodawania jest następujący:

```
  0111 1011b = 7Bh
+ 1001 0011b = 93h
-------------------
1 0000 1110b = 0Eh
C = 1
```

Ustawienie znacznika przeniesienia C (C=1) świadczy o przekroczeniu zakresu liczb całkowitych bez znaku 0..0FFh:

$$7Bh + 93h = 10Eh > 0FFh$$

- **znacznik przeniesienia połówkowego AC** ma podobne znaczenie jak znacznik przeniesienia C ale dotyczy przeniesienia między czterema mniej znaczącymi i czterema bardziej znaczącymi bitami akumulatora A. Jeśli wystąpi przeniesienie między bitami A_3 i A_4 w skrócie $A_{4|3}$, wówczas ustawiany jest znacznik AC:

```
  0010 0111b = 27 BCD
+ 0001 1001b = 19 BCD
---------------------
  0100 0000b ≠ 46 BCD
    AC = 1
```

Znacznik ten wykorzystywany jest tylko w operacjach arytmetycznych wykonywanych dla liczb zapisanych w kodzie BCD, którym towarzyszy instrukcja korekcji dziesiętnej DA A. Jeśli w akumulatorze znajduje się liczba 27 (liczba dziesiętna zapisana w kodzie BCD) to dodanie liczby 19 (też liczby dziesiętnej zapisanej w kodzie BCD) powinno dać wynik równy 46. Skorygowanie otrzymanego wyniku dodawania i uzyskanie poprawnego wyniku możliwe jest po wykonaniu instrukcji korekcji dziesiętnej DA A (patrz podrozdział 10.7).

- **znacznik nadmiaru OV** informuje o:
  - → przekroczeniu zakresu 8-bitowych liczb całkowitych ze znakiem, -128D..+127D w operacjach dodawania i odejmowania,
  - → iloczynie większym od 255 przy obliczaniu iloczynu,
  - → próbie dzielenia przez zero przy obliczaniu ilorazu.

```
  0110 1111b = 6Fh
+ 0001 0011b = 13h
-------------------
0 1000 0010b = 82h
  ↑    ↓
  └─1──┘
OV = C xor A7|6 = 1
```


Znacznik nadmiaru OV jest aktywny (przyjmuje wartość jedynki logicznej) jeśli wystąpiło przeniesienie między bitem A_6 i A_7, tzn. $A_{7|6}$, które zmieniło bit A_7. Stan znacznika nadmiaru (OV) można określić za pomocą wyrażenia:

$$OV = C \text{ xor } A_{7|6}$$

Druga i trzecia sytuacja występuje przy obliczaniu iloczynu zawartości akumulatora A i rejestru B oraz ilorazu obu rejestrów.

- stan znacznika parzystości **P** jest taki, aby liczba jedynek w akumulatorze A i znaczniku parzystości była parzysta. Znacznik ten ustawiany jest po **każdej instrukcji zmieniającej stan akumulatora:**

```
  0111 1111b = 7Fh
+ 0000 0001b = 01h
-------------------
  1000 0000b = 80h    P = 1
```

Ponieważ znacznik parzystości P jest modyfikowany po każdej instrukcji zmieniającej stan akumulatora A, dlatego również instrukcje przesłań mają na niego wpływ:

```
MOV   A,#0          ;P=0
MOV   A,#1          ;P=1
```

10.2 Dodawanie

Dodawanie dwóch 8-bitowych argumentów realizowane jest przez wykonanie instrukcji:

```
ADD   A,Rn          ;A ← A + Rn,    Rn=R0 .. R7
ADD   A,adr         ;A ← A + (adr)
ADD   A,@Ri         ;A ← A + (Ri),  Ri=0, 1
ADD   A,#dana       ;A ← A + dana
```

Przykładem sumy zawartości dwóch rejestrów R3 i R7 jest program:

```
Suma_R3_R7:
      MOV   A,R3        ;A ← R3
      ADD   A,R7        ;A ← A + R7,  wynik dodawania
```

Instrukcje te są przydatne tylko przy dodawaniu zmiennych 8-bitowych.

Jeśli przewiduje się dodawanie liczb wielobajtowych to przy dodawaniu bajtów najmniej znaczących wykorzystuje się rozkazy ADD ale już przy dodawaniu następnych bajtów trzeba uwzględnić znacznik przeniesienia C, użyć instrukcji ADDC.

Znacznik C jest przeniesieniem między kolejnymi bajtami. Instrukcje, które wykonują takie operacje są następujące:

```
ADDC   A,Rn      ;A ← A + C + Rn,      Rn=R0 .. R7
ADDC   A,adr     ;A ← A + C + (adr)
ADDC   A,@Ri     ;A ← A + C + (Ri),    Ri=0, 1
ADDC   A,#dana   ;A ← A + C + dana
```

Przy dodawaniu zawartości dwóch par rejestrów R4 (bajt mniej znaczący) i R5 oraz R6 (bajt mniej znaczący) i R7 program musi uwzględnić przeniesienie przy dodawaniu bardziej znaczących bajtów:

```
Suma_R4R5_R6R7:
    MOV    A,R4      ;A ← R4
    ADD    A,R6      ;A ← A + R6,
                     ;przeniesienie C zmienione przez sumę R4 + R6
    MOV    R6,A      ;R6 ← A, mniej znaczący bajt sumy
    MOV    A,R5      ;A ← R5, rozkazy:
                     ;MOV R6,A i MOV A,R5 nie zmieniają C
    ADDC   A,R7      ;A ← A + C + R7,
                     ;przeniesienie C z poprzedniego dodawania
    MOV    R7,A      ;R7 ← A, bardziej znaczący bajt sumy
```

Instrukcje **ADD i ADDC zmieniają** cztery znaczniki w rejestrze statusowym **PSW**: C, AC, OV i P.

W instrukcjach dodawania znajdują się także bardzo przydatne instrukcje dodawania 1, zwiększania o 1 (inkrementacji) zawartości rejestru lub komórki wewnętrznej pamięci RAM:

```
INC    A         ;A ← A +1
INC    Rn        ;Rn ← Rn +1
INC    adr       ;(adr) ← (adr) +1
INC    @Ri       ;(Ri) ← (Ri) +1
```

oraz jedyna instrukcja arytmetyczna, w której argumentem jest zmienna 16-bitowa, jaką jest wskaźnikowy rejestr danych DPTR:

```
INC    DPTR      ;DPTR ← DPTR +1
```

Instrukcje **INC nie zmieniają** znaczników w rejestrze statusowym **PSW**, z wyjątkiem znacznika P jeśli instrukcja zmienia stan akumulatora A.

10.3 Odejmowanie

Operacje odejmowania są mniej rozbudowane w stosunku do operacji dodawania. Występuje tylko jeden typ instrukcji SUBB odejmowanie z uwzględnianiem znacznika przeniesienia C. Odejmowanie dwóch zmiennych, z których jedna znajduje się zawsze w akumulatorze A, wykonywane jest w następujący sposób:

```
SUBB   A,Rn      ;A ← A - C - Rn,      Rn=R0 .. R7
SUBB   A,adr     ;A ← A - C - (adr)
SUBB   A,@Ri     ;A ← A - C - (Ri),    Ri=0, 1
SUBB   A,#dana   ;A ← A - C - dana
```

Instrukcje **SUBB mają wpływ na** wartość czterech znaczników w rejestrze statusowym **PSW**: C, AC, OV i P.

Znacznik przeniesienia C w operacjach odejmowania sygnalizuje pożyczkę (borrow). Odejmowanie zmiennych 1-bajtowych należy poprzedzić operacją kasowania znacznika przeniesienia C, np. stosując instrukcję CLR C.

Wynik odejmowania przesłany jest do akumulatora A. Znaczniki C, AC i OV przyjmują następujące wartości:

C = 1 jeśli odejmowanie bitu 7 odbyło się z pożyczką,
C = 0 jeśli warunek nie jest spełniony,

AC = 1 jeśli odejmowanie bitu 3 odbyło się z pożyczką,
AC = 0 jeśli warunek nie jest spełniony,

OV = 1 jeśli odejmowanie bitu 6 odbyło się z pożyczką i
odejmowanie bitu 7 odbyło się bez pożyczki
lub
jeśli odejmowanie bitu 6 odbyło się bez pożyczki i
odejmowanie bitu 7 odbyło się z pożyczką,
OV = 0 jeśli warunki nie są spełnione.

Znacznik przeniesienia C sygnalizuje przekroczenie zakresu liczb całkowitych bez znaku (0..255), a znacznik nadmiaru OV przekroczenie zakresu liczb całkowitych ze znakiem (-128..+127).

Poniższy przykład ilustruje odejmowanie dwóch zmiennych 2-bajtowych zawartych w komórkach pamięci RAM o adresach 30H i 40H. Różnica przesyłana jest w miejsce drugiej zmiennej. Obie wartości traktowane są jako liczby całkowite bez znaku:

```
Roznica_30_40:
        MOV     R1,#41H     ;R1 ← 41H,      adresuje mniej znaczący
                            ;               bajt drugiej zmiennej
        MOV     A,31H       ;A ← (31H),     przesłanie mniej znaczącego
                            ;               bajtu pierwszej zmiennej
        CLR     C           ;C ← 0,         zerowanie znacznika C
        SUBB    A,@R1       ;A ← A – C – (R1)
        MOV     @R1,A       ;(R1) ← A,      przesłanie mniej znaczącego
                            ;               bajtu różnicy
        DEC     R1          ;R1 ← R1 –1,    R1 = 40H
        MOV     A,30H       ;A ← (30H),     przesłanie bardziej znaczącego
                            ;               bajtu pierwszej zmiennej
        SUBB    A,@R1       ;A ← A – C – (R1)
        MOV     @R1,A       ;(R1) ← A,      przesłanie bardziej znaczącego
                            ;               bajtu różnicy
```

W instrukcjach odejmowania, podobnie jak w instrukcjach dodawania znajdują się instrukcje odejmowania 1, zmniejszania o 1 (dekrementacji) zawartości rejestru lub komórki wewnętrznej pamięci RAM:

```
DEC     A       ;A ← A – 1
DEC     Rn      ;Rn ← Rn – 1
DEC     adr     ;(adr) ← (adr) – 1
DEC     @Ri     ;(Ri) ← (Ri) – 1
```

Instrukcje **DEC nie zmieniają** znaczników w rejestrze statusowym **PSW**, z wyjątkiem znacznika P jeśli instrukcja zmienia stan akumulatora A.

Zmniejszenie o jeden zawartości 8-bitowego rejestru lub komórki wewnętrznej pamięci RAM wykonywane jest modulo 8 bitów:

```
Dec_R1:
        MOV     R1,#0       ;R1 ← 0
        DEC     R1          ;R1 ← R1 - 1,     R1 = 0FFh
```

W instrukcjach odejmowania jedności brak jest dekrementacji wskaźnikowego rejestru danych DPTR. Operację tę można na przykład zrealizować wykonując instrukcje:

```
        CLR     C           ;C ← 0
        MOV     A,DPL       ;A ← DPL,       dekrementacja rejestru DPL
        SUBB    A,#1        ;A ← A – C – 1
        MOV     DPL,A       ;DPL ← A
        MOV     A,DPH       ;A ← DPH
        SUBB    A,#0        ;A ← A – C – 0, jeśli wystąpiła pożyczka (C=1)
                            ;               to dekrementacja rejestru DPH
        MOV     DPH,A       ;DPH ← A
```


Przy **adresowaniu tablic** w pamięci programu ROM lub zewnętrznej pamięci danych RAM za pośrednictwem wskaźnikowego rejestru danych DPTR należy pamiętać aby **adresy** kolejnych komórek pamięci **narastały**, a nie malały. Jest to ważne ponieważ jest instrukcja inkrementacji wskaźnikowego rejestru danych DPTR (INC DPTR) ale brak jest dekrementacji tego rejestru

10.4 Porównania

Z instrukcjami odejmowania związane są rozkazy DJNZ (Decrement and Jump relative if Not Zero) będące połączeniem dekrementacji (zmniejszania o 1) i skoku warunkowego:

```
DJNZ    Rn,rel      ;Rn ← Rn – 1
                    ;jeśli Rn ≠ 0 to skocz do PC+rel
DJNZ    adr,rel     ;(adr) ← (adr) – 1
                    ;jeśli (adr) ≠ 0 to skocz do PC+rel
```

które stosowane są przeważnie przy powtórzeniach programu nie przekraczających 256 razy. Instrukcje DJNZ wykonywane są w dwóch etapach:

- zmniejszenie o jeden pierwszego argumentu, tj. zawartości rejestru Rn (R0 .. R7) lub komórki wewnętrznej pamięci RAM adresowanej bezpośrednio, także rejestrów specjalnych SFR (które adresowane są tylko bezpośrednio),
- wykonanie skoku do adresu będącego sumą zawartości licznika rozkazów PC (wskazuje zawsze pierwszy bajt następnej do wykonania instrukcji) i liczby (przesunięcia) rel. Liczba rel traktowana jest jako 8-bitowa liczba całkowita ze znakiem.

Pierwszy z argumentów (rejestr Rn lub komórka pamięci adresowana bezpośrednio o adresie adr) w powtórzeniach programu pełni rolę licznika powtórzeń.

Zamiast mozolnie obliczać przesunięcie rel stanowiące drugi z argumentów instrukcji DJNZ w praktyce podaje się odpowiednie etykiety, adresy instrukcji, do których ma wystąpić skok. Właściwą wartość przesunięcia rel oblicza asembler w trakcie asemblacji programu:

```
        .....
        MOV     R7,#4       ;rejestr R7 pełni rolę licznika powtórzeń
Skok:   XRL     P1,#0FFh    ;P1 ← P1,       początek pętli programu
        DJNZ    R7,Skok     ;R7 ← R7 –1
                            ;jeśli R7 ≠ 0 to skok do etykiety Skok
Dalej:  .....               ;dalsza część programu
```

Podany program 4-krotnie neguje stan linii portu P1. Ponieważ przesunięcie rel = Skok – Dalej jest liczbą całkowitą ze znakiem, dlatego adres skoku może

różnić się więcej niż o -128 .. +127 w stosunku do adresu następnego rozkazu występującego po instrukcji DJNZ. W podanym przykładzie różnica adresów (etykiet) nie może przekraczać wartości:

$$-128 < Skok - Dalej < 127$$

Jeśli stosowany debuger nie wyswietla użytych w asemblerze symboli, a podany program rozpoczyna się np. od adresu 01A0h, to po uruchomieniu debugera na ekranie komputera może pojawić się zapis:

pamięć kodu programu				postać symboliczna programu:			
adres:	zawartość:						
	.....				.....		
01A0h	7Fh	04h			MOV	R7,#4	
01A2h	63h	90h	0FFh	Skok:	XRL	P1,#0FFh	
01A5h	0DFh	0FBh			DJNZ	R7,Skok	
01A7h	.....			Dalej	.....		

Asembler obliczył różnicę adresów Skok – Dalej = 0FBh i podstawił ją jako drugi argument instrukcji DJNZ R7,0FBh.

Jeśli podany w instrukcji DJNZ warunek (Rn ≠ 0 lub (adr) ≠ 0) nie jest spełniony, to mikrokontroler wykonuje następną instrukcję programu.

Instrukcje **DJNZ nie wpływają** na stan znaczników słowa statusowego **PSW**, z wyjątkiem instrukcji DJNZ ACC,rel.

Maksymalna liczba powtórzeń pętli programu z użyciem instrukcji DJNZ wynosi 256. Aby to uzyskać należy wpisać do rejestru licznikowego **wartość początkową** równą **0**, ponieważ jako pierwsze wykonywane jest odejmowanie 1, a pózniej porównanie.

W wielu pętlach programu służących do odmierzania odcinków czasu wykorzystywana jest instrukcja NOP, np:

```
        MOV   R7,#0      ;rejestr R7 jest licznikiem pętli
Skok:   NOP              ;1 cykl maszyn.
        DJNZ  R7,Skok    ;2 cykle maszyn.
```

Instrukcja NOP trwa jeden cykl maszynowy i w tym czasie procesor nie zmienia zawartości żadnego z rejestrów lub komórek pamięci. Czas wykonywania pętli programu wynosi 256 * (1+2) = 768 cykli maszynowych.

Porównanie dwóch argumentów prowadzi zwykle do rozgałęzienia programu, czyli skoku do fragmentu programu jeśli wynikiem porównania jest

prawda lub realizacja dalszej części programu jeśli wynikiem porównania jest fałsz. Wynika z tego, że wykonywane są równocześnie dwie operacje: porównania i skoku warunkowego. Konstruktorzy mikrokontrolera 8051 i wszystkich następnych tej rodziny przewidzieli taką sytuację. W liście rozkazów znalazły się rozkazy pełniące podane funkcje oznaczone symbolem CJNE (Compare and Jump relative if Not Equal):

```
CJNE   A,adr,rel      ;jeśli A≠(adr) to skocz do PC+rel
CJNE   A,#dana,rel    ;jeśli A≠dana to skocz do PC+rel
CJNE   Rn,#dana,rel   ;jeśli Rn≠dana to skocz do PC+rel
CJNE   @Ri,#dana,rel  ;jeśli (Ri)≠dana to skocz do PC+rel
```

We wszystkich rozkazach **CJNE zmieniany jest znacznik przeniesienia C** w następujący sposób:

- **jeśli Argument_1 < Argument_2 to C=1,**
- **jeśli Argument_1 ≥ Argument_2 to C=0,**
- Argumentem_1 jest: A, Rn (R0 .. R7) lub Ri (R0, R1),
- Argumentem_2 jest: adr (komórka wewnętrznej pamięci RAM adresowana bezpośrednio albo rejestry specjalne SFR) lub 8-bitowa dana będąca drugim bajtem instrukcji.

Rozkazy CJNE umożliwiają w bezpośredni sposób wykonanie skoku jeśli oba argumenty są różne. Ponizsze przykłady wskazują jak wykonać porównanie i skok w przypadku gdy:

- Argument_1 < Argument_2, np. przy porównaniu zawartości akumulatora A i rejestru B należy wykonać skok do programu rozpoczynającego się od etykiety (adresu) Wykonaj jeśli A < B, a jeśli A ≥ B to do programu rozpoczynającego się etykietą Dalej:

```
        CJNE   A,B,Tutaj     ;jeśli A < B to C=1
Tutaj:  JC     Wykonaj       ;skok jeśli C=1
Dalej:
```

- Argument_1 ≥ Argument_2, np. przy porównaniu zawartości rejestru R0 i zmiennej 3Ch należy wykonać skok do programu rozpoczynającego się od etykiety Skok_1 jeśli R0 ≥ 3Ch, a jeśli R0 < 3Ch to do programu rozpoczynającego się etykietą Skok_2:

```
        CJNE   R0,#3Ch,Tutaj   ;jeśli R0 ≥ 3Ch to C=0
Tutaj:  JNC    Skok_1          ;skok jeśli C=0
Skok_2: .....
```

- Argument_1 > Argument_2, np. przy porównaniu zawartości akumulatora A i zmiennej 2Eh należy wykonać skok do programu

rozpoczynającego się od etykiety Duzy jeśli A > 2Eh, a jeśli A ≤ 2Eh to do programu rozpoczynającego się etykietą Maly:

```
        CJNE    A,#2Eh,Tutaj    ;jeśli A ≥ 2Eh to C=0
Tutaj:  JC      Maly            ;skok jeśli C=1
        JNZ     Duzy            ;skok jeśli A ≠ 0
Maly:   .....
```

10.5 Mnożenie

Operacja mnożenia MUL AB wykonuje mnożenie dwóch zmiennych 8-bitowych, z których jedna znajduje się w akumulatorze A, a druga w rejestrze B. Wynik wykonanej operacji przesyłany jest do rejestru B (8 bardziej znaczących bitów wyniku) i akumulatora A (8 mniej znaczących bitów wyniku):

```
MUL     AB              ;BA ← A * B
```

Znacznik przeniesienia C i nadmiaru OV przyjmują następujące wartości:

C = 0 zawsze

OV = 1 jeśli iloczyn jest liczbą większą od 255, tzn. zawartość rejestru B jest niezerowa;

OV = 0 jeśli warunek nie jest spełniony.

Obie zmienne traktowane są jako liczby całkowite bez znaku. Instrukcja mnożenia należy do najdłużej (wraz z instrukcja dzielenia) wykonywanych instrukcji. Trwa 4 cykle maszynowe.

Mnożenie dwóch stałych o wartościach 18 i 29 realizowane jest następująco:

```
Mul_AB:
        MOV     A,#18d          ;A ← 18, dziesiętnie,
        MOV     B,#29d          ;B ← 29, dziesiętnie
        MUL     AB              ;BA ← A * B,        18 * 29 = 522 = 20Ah
                                ;stan rejestrów:    A = 0Ah,
                                ;                   B = 2,
                                ;i znaczników:      C = 0 (zawsze zerowany),
                                ;                   OV = 1, ponieważ B ≠ 0
                                ;                   P = 0
```

Jeżeli operacje mnożenia dotyczą potęgi 2 (np. 2^k), to można je zastąpić rozkazami k-krotnego przesunięcia logicznego w lewo, np. przy mnożeniu przez 4 liczby przechowywanej w rejestrze R4:

```
        MOV     A,R4            ;A ← R4
        CLR     C               ;C ← 0
        RLC     A               ;przesunięcie akumulatora A w lewo
        RLC     A               ;przesunięcie akumulatora A w lewo
        MOV     R4,A            ;R4 ← A
```


W podanym przykładzie nie jest wykrywane i sygnalizowane przekroczenie zakresu liczby. Jeśli nie jest się pewnym zakresu mnożonej liczby, to bezpieczniejsze jest użycie instrukcji MUL AB.

Podobna w działaniu do rozkazu RLC A jest instrukcjia RL A, która także przesuwa w lewo zawartość akumulatora A ale w obrębie 8-bitów, a nie 9-bitów. Graficzne obie instrukcje przedstawić można w następujący sposób:

10.6 Dzielenie

Instrukcja dzielenia DIV AB wykonuje dzielenie 8-bitowej zmiennej znajdującej się w akumulatorze A przez zmienną 8-bitową znajdującą się w rejestrze B. Część całkowita ilorazu przesyłana jest do akumulatora A, a reszta z dzielenia do rejestru B:

```
DIV     AB              ;A ← Int( A/B ), iloraz
                        ;B ← Mod( A/B ), reszta
```

Znacznik przeniesienia C i nadmiaru OV przyjmują następujące wartości:

C = 0 zawsze

OV = 1 w przypadku próby dzielenia przez zero, stan rejestrów A i B jest nieokreślony,

OV = 0 jeśli dzielenie przebiegło prawidłowo

Obie zmienne traktowane są jako liczby całkowite bez znaku, a instrukcja wykonywana jest w ciągu 4 cykli maszynowych.

Przy zamianie 1-bajtowej liczby zapisanej w kodzie szesnastkowym z przedziału 0 .. 99 na liczbę zapisaną w kodzie BCD wykonywane są instrukcje:

```
Hex_BCD_99:
        MOV     A,#5Fh          ;A ← 5Fh,   zamieniana liczba w kodzie
                                ;           szesnastkowym,
        MOV     B,#10           ;B ←10
        DIV     AB              ;A ← Int(5Fh/10) = 9 jest liczbą dziesiątek,
                                ;B ← Mod(5Fh/10) = 5 jest liczbą jedności
```

Jeżeli operacje dzielenia dotyczą potęgi 2 (np. 2^k), to można je zastąpić rozkazami k-krotnego przesunięcia logicznego w prawo. Należy przy tym przeanalizować, która metoda da krótszy kod wynikowy lub która będzie szybsza. Równie ważnym zagadnieniem jest przy operacjach przesuwania logicznego odtworzenie znaku przesuwanej liczby. Bit najbardziej znaczący (MSB) jest znakiem liczby w kodzie uzupełnienia do 2 , np:

```
MOV   C,ACC.7      ;C ← ACC.7,   odtworzenie znaku liczby
                   ;             MSB = ACC 7
RRC   A            ;przesunięcie akumulatora A w prawo
```

W przykładzie wykorzystano instrukcję RRC A, która ma (podobnie jak instrukcja przesuwania w lewo RLC A) skróconą wersję przesuwania w prawo RR A. Różnica między obu rozkazami polega na tym, że rozkaz RR C przesuwa tylko zawartość akumulatora A, a rozkaz RRC A przesuwa zawartość akumulatora A z uwzględnieniem dodatkowego 9 bitu, którym jest znacznik przeniesienia C:

10.7 Korekcja dziesiętna

Ostatnią rozważaną instrukcją arytmetyczną jest instrukcja korekcji dziesiętnej DA A. Stosowana jest w operacjach dodawania liczb zapisanych w kodzie BCD, po instrukcji ADD lub ADDC. Jeśli w wyniku dodawania:

- wartość 4 mniej znaczących bitów akumulatora $A_{3..0} > 9$ lub znacznik przeniesienia połówkowego AC = 1 to:
  → A = A + 6, do akumulatora A następuje dodanie wartości 6; Dodawanie może ustawić znacznik przeniesienia C (C=1) ale nie może go wyzerować, tzn. C=0,
- wartość 4 bardziej znaczących bitów akumulatora $A_{7..4} > 9$ lub znacznik przeniesienia C = 1 to:
  → A = A + 60h, do akumulatora A następuje dodanie wartości 60h; dodawanie może ustawić znacznik przeniesienia C (C=1) ale nie może go wyzerować, tzn. C=0.

Instrukcja **DA A nie może być stosowana do bezpośredniej zamiany liczby zapisanej w kodzie binarnym na liczbę zapisaną w kodzie BCD.** Taka zamiana może być tylko wynikiem odpowiednio przygotowanego programu, który przedstawiony jest w [3].

Sposób wykorzystania instrukcji korekcji dziesiętnej przedstawia poniższy przykład, w którym występuje dodawanie dwóch zmiennych 2-bajtowych zapisanych w kodzie BCD i zawartych w wewnętrznej pamięci RAM:

pierwsza zmienna w komórce wewnętrznej pamięci RAM:
Adres_Dana_1 - adres bajtu mniej znaczącego,
Adres_Dana_1+1 - adres bajtu bardziej znaczącego,
295 BCD - wartość dziesiętna zmiennej zapisanej w kodzie BCD,

druga zmienna w komórce wewnętrznej pamięci RAM:
Adres_Dana_2 - adres bajtu mniej znaczącego,
Adres_Dana_2+1 - adres bajtu bardziej znaczącego,
685 BCD - wartość dziesiętna zmiennej zapisanej w kodzie BCD,

```
Suma_BCD:
        MOV   R0,#Adres_Dana_1   ;R0 adresuje pierwszą zmienną
        MOV   R1,#Adres_Dana_2   ;R1 adresuje drugą zmienną i wynik
        MOV   R2,#2              ;R2 jest licznikiem dodawanych bajtów
        CLR   C
Pla:    MOV   A,@R0              ;pobranie pierwszego argumentu
        ADDC  A,@R1              ;dodawanie z drugim argumentem
        DA    A                  ;korekcja dziesiętna
        MOV   @R1,A              ;przesłanie wyniku dodawania
        INC   R0                 ;przygotowanie do dodawania
        INC   R1                 ;kolejnych bajtów obu zmiennych
        DJNZ  R2,Pla             ;powtórzenie pętli dodawania
Dalej:                           ;dalsza część programu użytkownika
```

Wartości poszczególnych rejestrów w trakcie wykonywania programu są następujące:

```
                                C = 0
(Adres_Dana_1) = 95h = 1001 0101b   dodawanie mniej znaczących
(Adres_Dana_2) = 85h = 1000 0101b   bajtów
Suma:  A = 1Ah = 0001 1010b     C = 1, A7..4 ≤ 9,    AC = 0, A3..0 > 9
                                jeśli A3..0 > 9 lub AC=1 to dodanie 6
Korekcja dziesiętna A3..0 : 0000 0110b
Suma:  A = 20h = 0010 0000b     nie zmienia C, C=1 !
                                jeśli: A7..4 > 9 lub C=1 to dodanie 60h
```

Korekcja dziesiętna $A_{7..4}$: 0110 0000b		
Suma:	A = 80h = 1000 0000b	nie zmienia C, C=1 !
	(Adres_Dana_2) ← A	przesłanie sumy
	C = 1	z poprzedniej korekcji
	(Adres_Dana_1+1) = 2 = 0000 0010b	dodawanie bardziej znaczących
	(Adres_Dana_2+1) = 6 = 0000 0110b	bajtów
Suma:	A = 9 = 0000 1001b	C=0, $A_{7..4} \leq 9$, AC=0, $A_{3..0} \leq 9$
		brak korekcji dziesiętnej
	(Adres_Dana_2+1) ← A	przesłanie sumy

Suma dwóch liczb zawarta w komórkach wewnętrznej pamięci RAM o adresie Adres_Dana_2 (mniej znaczący bajt sumy) i Adres_Dana_2+1 (bardziej znaczący bajt sumy) jest równa 980 BCD. Należy pamiętać, że liczba ta zapisana jest w kodzie BCD.

Pytania i problemy

1. Czy prawdziwe są równości:

a) 100 = 64h b) 1001 0111b = 150 c) 1001 0111b = –105

d) 110 001o = 31h e) 1111 1111b = –128 f) 1000 0000b = –1

2. W których instrukcjach arytmetycznych zmieniany jest i w jaki sposób znacznik przeniesienia C?

3. Do czego służy znacznik nadmiaru OV ? Jaki jest sposób jego obliczania ?

4. Jaka jest różnica między instrukcjami dodawania: ADD A,#1 oraz INC A ?

5. Czym różnią się instrukcje: ADD A,@R1 oraz ADDC A,@R1 ?

6. Jaka jest różnica między instrukcjami: SUBB A,#1 oraz DEC A ?

7. Jak inaczej, niż to pokazano w przykładzie, wykonać dekrementację wskaźnikowego rejestru danych DPTR ?

8. Ile maksymalnie bajtów może liczyć pętla programu objęta instrukcją DJNZ ?

9. Jak wykonać skok do programu po porównaniu dwóch argumentów jeśli oba argumenty są sobie równe ?

10. Jaka może być maksymalna różnica między adresem instrukcji CJNE i początku programu, do którego ma nastąpić skok ?

11. Jak sprawdzić prawidłowy przebieg dzielenia dwóch liczb 8-bitowych ?

12. Czy możliwe jest, używając bezpośrednio instrukcji mnożenia MUL AB, wykonać mnożenie dwóch 1-bajtowych liczb całkowitych ze znakiem ?

13. Co należałoby zrobić, aby wykonać dodawanie dwóch zmiennych 16-bitowych zapisanych w kodzie BCD ? Podać przykład programu.

14. Jak wykonać odejmowanie dwóch liczb dziesiętnych zapisanych w kodzie BCD ?

11. Programy, podprogramy, segmenty

Wszystkie procesory, a więc i mikrokontrolery, działają według odpowiedniego programu umieszczonego najczęściej w pamięci stałej (ROM, EPROM) nazywanej pamięcią programu, znajdującej się na zewnątrz lub wewnątrz mikrokontrolera.

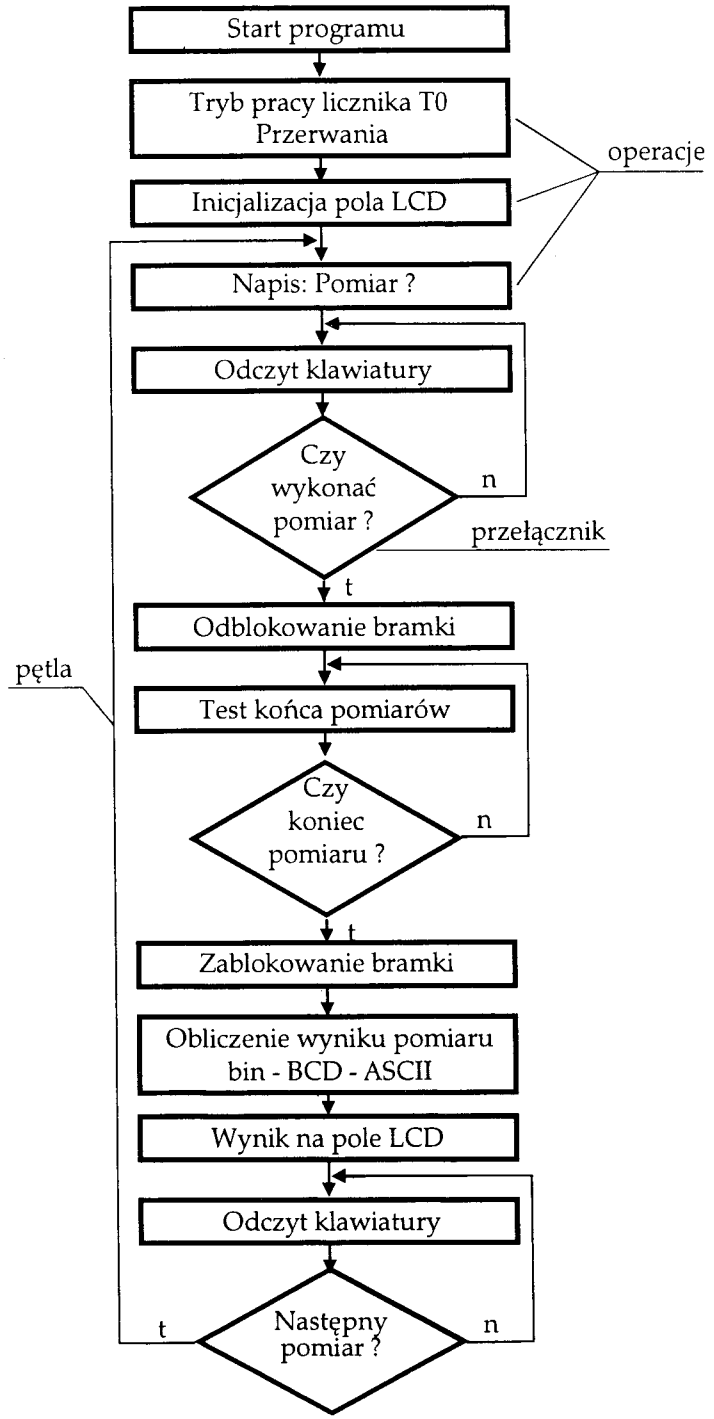
Zadaniem programisty systemu mikroprocesorowego jest napisanie odpowiedniego programu i umieszczenie go w pamięci programu procesora. Przed przystąpieniem do pisania programu należy mieć jasny obraz czynności, które powinien wykonywać procesor. Do tego celu służy algorytm.

Ogólnie algorytmem nazywa się ściśle określony sposób postępowania, doprowadzający do rozwiązania zadania. Operacje realizowane kolejno w czasie nazywa się krokami algorytmu. Algorytm przedstawia w prosty i czytelny sposób funkcje wykonywane przez procesor. Ułatwia pisanie i testowanie, zwłaszcza dużych programów. Przy bardzo dużych programach umożliwia podział programu na bloki, które mogą być pisane i testowane samodzielnie przez różnych programistów. Algorytm pokazuje jak powinien wyglądać program główny, jakie występują podprogramy i z którego miejsca w programie są wywoływane oraz ile wystąpi podprogramów obsługi przerwań.

Podstawowymi elementami algorytmu są:

- sekwencja operacji - jest to zbiór operacji realizowanych w określonej kolejności na określonych danych.
- przełącznik - jest elementem badającym spełnienie określonego warunku i wybierającym jedną z dwóch alternatywnych sekwencji operacji, w zależności od tego, czy warunek ten jest, czy nie jest spełniony.
- pętla - jest elementem umożliwiającym kolejną wielokrotną realizację określonej operacji na danych.

Poniżej, na rysunku 11-1 jest przedstawiony przykładowy algorytm do programu „pomiar odcinka czasu". Na początku programu należy ustalić tryb pracy licznika T0, w którym odbywa się zliczanie impulsów, odblokować przerwanie od licznika T0 i uaktywnić pole odczytowe. Następnie na polu odczytowym należy wyświetlić napis informujący, że urządzenie jest gotowe do pomiaru. Start pomiaru odbywa się po naciśnięciu klawisza, stąd pętla testująca klawiaturę. Po stwierdzeniu, że został naciśnięty odpowiedni klawisz następuje odblokowanie bramki. W celu określenia momentu zakończenia pomiaru jest testowany zewnętrzny sygnał bramkujący. Testowanie odbywa się przez cały czas pomiaru stąd kolejna pętla. Po zakończeniu pomiaru następuje przeliczenie wyniku na wymagane jednostki czasu w kodach ASCII, co umożliwia wyświetlenie go na polu odczytowym LCD.



Rys. 11-1 Algorytm do programu „pomiar odcinka czasu".

Ponieważ od tego momentu użytkownik przyrządu musi podjąć decyzję, czy należy wykonać następny pomiar, odbywa się testowanie stanu klawiatury. Po stwierdzeniu, że ma być wykonany następny pomiar, następuje skok do miejsca w programie, w którym na pole odczytowe jest wysyłany napis o gotowości przyrządu do pomiaru.

Jest to algorytm bardzo ogólny, ale pokazuje, że należy podjąć decyzję skąd będzie brany sygnał startu pomiaru, że będą potrzebne procedury obliczeniowe, itp.

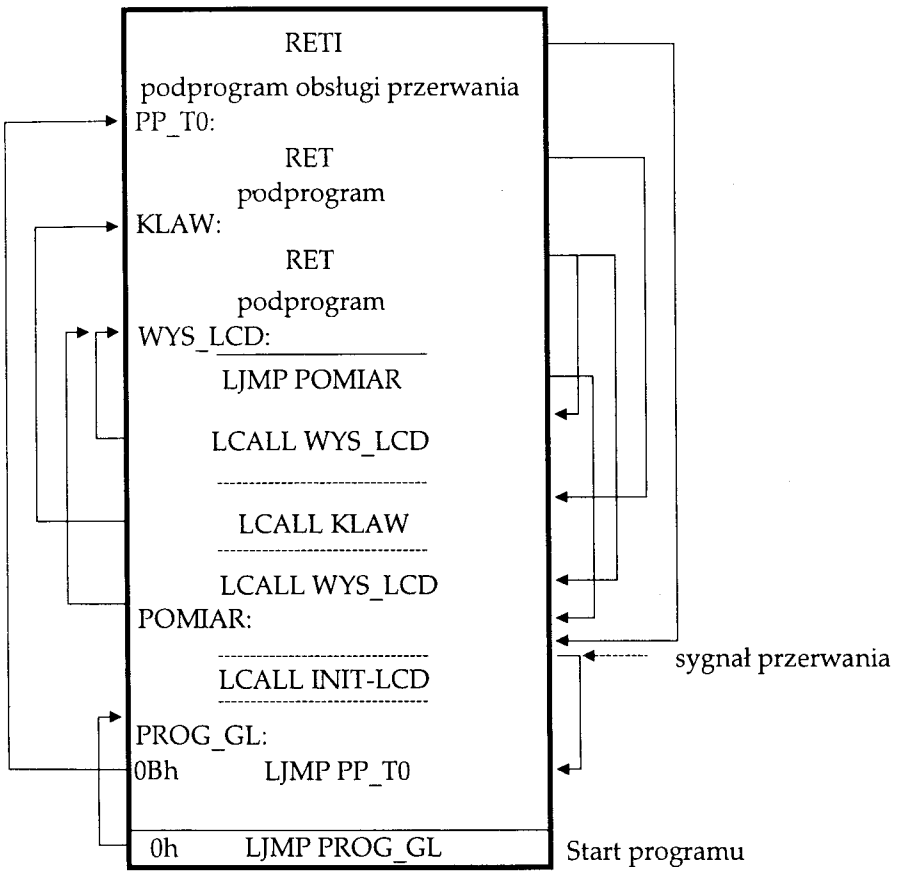
Jeżeli program jest złożony, to algorytmy wykonuje się dla podprogramów lub programów częściowych, na przykład do powyższego algorytmu może być potrzebny algorytm podprogramu określania momentu początku pomiaru lub algorytm podprogramu obsługi przerwania.

Program możemy podzielić na program główny oraz podprogramy. Program główny zawiera najczęściej procedury inicjalizujące parametry początkowe, programujące układy struktury wewnętrznej, na przykład tryb pracy licznika, ustawienie przerwań, itp. oraz wywołujące podprogramy.

W podprogramach umieszcza się zazwyczaj procedury obsługi elementów struktury zewnętrznej procesora, na przykład pola odczytowego, klawiatury, itp., lub wewnętrznej - przetwornika A/C, portu szeregowego oraz procedur obliczeniowych np. mnożenia, dzielenia. Umieszczenie tego typu procedur w podprogramach umożliwia tworzenie bibliotek, z których w razie potrzeby można wybierać określoną procedurę i dołączać ją do nowo utworzonego programu. W podprogramach umieszcza się także powtarzające się w programie sekwencje rozkazów. Skraca to długość programu ale wydłuża czas jego wykonania. Podprogramy wywołuje się rozkazami ACALL lub LCALL , natomiast podprogram musi być zakończony rozkazem RET lub RETI - jeżeli jest to podprogram obsługi przerwania. Na rysunku 11-2 jest przedstawiony przykład, powiązany z przedstawionym wyżej algorytmem, wywoływania podprogramów z programu głównego oraz przez układ przerwań.

Dla mikrokontrolerów rodziny '51 start programu, po zerowaniu procesora, zaczyna się zawsze od adresu 0. Pod tym adresem jest umieszczony zazwyczaj rozkaz skoku do programu głównego. Skok ten omija obszar przeznaczony na adresy obsługi przerwań. W mikrokontrolerach rodziny '51 występują następujące rozkazy skoków: AJMP, LJMP, SJMP i JMP @A+DPTR. W trakcie wykonywania programu mogą być wywoływane podprogramy. Na rysunku 11-2 pokazano wywołanie podprogramu INIT_LCD, inicjalizującego pole odczytowe LCD, oraz wywołanie podprogramów odczytu klawiatury KLAW i wyświetlania informacji na polu odczytowym WYS_LCD. W przypadku tych dwóch ostatnich podprogramów zostało pokazane wejście i wyjście do i z podprogramów. Za każdym razem po wykonaniu podprogramu następuje powrót do programu głównego i jest wykonywany rozkaz umieszczony tuż za rozkazem wywołują-

cym podprogram. Podprogramy te zostały umieszczone powyżej programu głównego. Na samym końcu został umieszczony podprogram obsługi przerwania od licznika T0. Program główny jest zakończony rozkazem skoku do procedury wyświetlania na polu odczytowym zapytania o kolejny pomiar.



Rys. 11-2 Wywoływanie podprogramów.

Jeżeli w trakcie wykonywania programu zostanie przyjęty sygnał przerwania, to zgodnie z wcześniej omówionymi zasadami, procesor przejdzie do obsługi tego przerwania wywołując odpowiedni podprogram rozkazem LJMP PP_T0.

Wywoływanie podprogramów może następować z dowolnego miejsca - z podprogramów obsługi przerwań jak i z innych podprogramów. Umieszczenie podprogramów względem programu głównego może być dowolne.

Rozkazy skoków, wywołania i powrotów z podprogramów są przedstawione w poniższych tabelach:

Tabela 11-1. Rozkazy skoków

instrukcje:	zapis binarny instrukcji	wykonywane operacje
AJMP adr_11	$A_{10}A_9A_8$00010 $A_7 \div A_0$	PC ← PC + 2, PC ← ADR_11
LJMP adr_16	00000010 $A_{15} \div A_8$ $A_7 \div A_0$	PC ← PC + 3, PC ← adr_16
SJMP rel	10000000 rel	PC ← PC + 2, PC ← PC + rel
JMP @A + DPTR	01110011	PC ← PC + 2, PC ← A + DPTR

Instrukcje skoków bezwarunkowych powodują wpisanie do licznika rozkazów PC adresu następnego wykonywanego programu. Adres może być podany w sposób bezpośredni, ale uniemożliwia to przesuwanie programów, lub w postaci etykiety, co jest najczęściej stosowane. Rozkaz ACALL operuje tylko adresem 11-bitowym. Umożliwia to wykonywanie skoków w obszarze 2 KB bloku pamięci programu. Rozkaz LCALL daje możliwość wykonywania skoków w całym 64 KB obszarze pamięci programu. Rozkaz SJMP traktuje wartość rel jako 8 bitową liczbę ze znakiem o zakresie -128 ÷ +127. W miejsce wartości rel można także wstawić etykietę, a asembler sam wyliczy odpowiednią wartość. W większości asemblerów wystarczy podać tylko rozkaz JMP, a asembler sam dobierze odpowiedni rozkaz do długości skoku. W rozkazie JMP @A+DPTR do rejestru PC jest wpisywana suma zawartości akumulatora i rejestru DPTR, przy czym liczba w akumulatorze jest traktowana jako liczba całkowita bez znaku, tzn. o zakresie 0 ÷ 255.

Tabela 11-2. Rozkazy wywołania podprogramów.

instrukcje:	zapis binarny instrukcji	wykonywane operacje
ACALL adr_11	$A_{10}A_9A_8$00001 $A_7 \div A_0$	PC←PC+2, SP←SP+1 (SP)←$PC_{7...0}$, SP←SP+1 (SP)←$PC_{15...8}$, $PC_{10...0}$←adr_11
LCALL adr_16	00010010 $A_{15} \div A_8$ $A_7 \div A_0$	PC←PC+3, SP←SP+1 (SP)←$PC_{7...0}$, SP←SP+1 (SP)←$PC_{15...8}$, PC←adr_16

Rozkazy wywołania podprogramów powodują schowanie na stos adresu, od którego będzie wykonywany program po powrocie z podprogramu, a do licznika jest wpisywany adres, od którego zaczyna się podprogram. Zakresy

adresów są takie same jak odpowiednio w rozkazach AJMP i LJMP. Również w większości asemblerów można podać skrót CALL, a asembler dobierze odpowiedni typ rozkazu.

Tabela 11-3 Rozkazy powrotu z podprogramów

instrukcje:	zapis binarny instrukcji	wykonywane operacje
RET	00100010	$PC_{15..8} \leftarrow (SP)$, $SP \leftarrow SP - 1$ $PC_{7..0} \leftarrow (SP)$, $SP \leftarrow SP - 1$
RETI	00110010	$PC_{15..8} \leftarrow (SP)$, $SP \leftarrow SP - 1$ $PC_{7..0} \leftarrow (SP)$, $SP \leftarrow SP - 1$

Rozkazy powrotu z podprogramu powodują przepisanie do licznika rozkazów PC adresu przechowywanego na stosie. Dodatkowo rozkaz RETI stosowany w podprogramach obsługi przerwania odblokowuje system przerwań.

W celu ułatwienia programiście operowania danymi, łączenia programów w określonej kolejności w większości narzędzi programowania jest możliwy podział programu na segmenty. W segmentach następuje przypisanie wybranej grupy zmiennych do określonego typu pamięci - pamięć danych zewnętrzna, wewnętrzna, pamięć programu. Również wybrane sekwencje rozkazów, najczęściej podprogramy, są przypisywane do segmentów.

Segmenty mogą być relokowalne lub nierelokowalne. Segmenty nierelokowalne są to segmenty umieszczane w obszarze pamięci od adresu zadeklarowanego przez programistę. W trakcie łączenia programów przez linker tego typu segmenty nie mogą być przesuwane do innego obszaru pamięci. Segmenty nierelokowalne są stosowane dla zadeklarowania adresów obsługi przerwań, przy pojedynczych programach, lub gdy program ma być umieszczony od zadanego adresu, np. w niektórych modułach mikroprocesorowych. W powyższym przykładzie w segmencie nierelokowalnym zostały umieszczone rozkazy skoku do programu głównego i podprogramu obsługi przerwania.

Segmenty relokowalne są to segmenty, które po asemblacji otrzymują adresy chwilowe, natomiast adresy ostateczne uzyskują po linkowaniu. Mają one bardzo istotne znaczenie, gdyż umożliwiają skorzystanie z wcześniej przygotowanych procedur (programy biblioteczne) dla potrzeb innych programów. Na przykład raz napisana procedura mnożenia lub dzielenia wielobajtowego może być dołączana do różnych programów w miejscu wskazanym przez programistę. W powyższym przykładzie, jeżeli podprogramy inicjalizacji pola LCD, odczytu klawiatury i wysyłania znaku na pole odczytowe będą dołączane do programu głównego przez linker, to będą one musiały być umieszczone w segmentach relokowalnych.

Do łączenia programów relokowalnych służy linker, w którym zadaje się adres początkowy programu i podaje kolejność rozmieszczenia łączonych programów. Adresy, pod którymi są umieszczone programy, poza początkowym, linker przydziela automatycznie.

Podobnie ma się z rozmieszczeniem danych w pamięci RAM. Po zadeklarowaniu odpowiednich segmentów relokowalnych linker łączy poszczególne segmenty w jedną całość, dzięki czemu programista nie musi kontrolować, czy obszary danych nie nakładają się na siebie, lub nie zostaje niewykorzystana wolna przestrzeń pamięci.

Sposób deklarowania segmentów jest podany w rozdziale 17 opisującym narzędzia programowania.

? Pytania i problemy

1. Co to jest algorytm i do czego służy ?
2. Jakie są podstawowe elementy algorytmu ?
3. Ułożyć algorytm dla regulatora temperatury.
4. Dlaczego programy są podzielone na podprogramy ?
5. Czym charakteryzuje się podprogram obsługi przerwania ?
6. Jak mogą być rozmieszczone podprogramy w pamięci programu procesora ?
7. Co to są segmenty ?
8. Czym charakteryzują się segmenty relokowalne ?
9. Czym charakteryzują się segmenty nierelokowalne ?

12. Port szeregowy

Zadaniem portu szeregowego jest zapewnienie komunikacji poprzez łącze szeregowe między mikrokontrolerem a urządzeniami zewnętrznymi. Dane są wysyłane bit po bicie poczynając od najmniej znaczącego bitu. Dane mogą być wysyłane i przyjmowane synchronicznie lub asynchronicznie.

Przy transmisji synchronicznej przesyłanym danym towarzyszy sygnał synchronizujący, względem którego określa się stany przesyłanych bitów. Zaletą transmisji synchronicznej jest jej odporność na dewiacje częstotliwości sygnału taktującego, natomiast wadą - dwie linie przęsyłowe: jedna dla danych a druga dla sygnału taktującego.

Przy transmisji asynchronicznej (UART) przesyłane są tylko dane, a sygnały taktujące są wytwarzane w nadajniku i odbiorniku niezależnie od siebie. Dlatego dla poprawnego odebrania danych nadajnik i odbiornik muszą być taktowane sygnałem o takiej samej częstotliwości. Wymaga to generatorów o wysokiej stabilności i uzgodnienia co do prędkości transmisji. Dla ułatwienia nawiązania łączności prędkości transmisji zostały unormowane. Są one podane w normie opisującej łącze szeregowe RS 232.

Opisany poniżej port szeregowy jest identyczny dla całej rodziny mikrokontrolerów '51. Oprócz niego, w poszczególnych typach tej rodziny mogą występować urządzenia do innych rodzajów transmisji, takich jak CAN, I^2C itp. W mikrokontrolerach spoza rodziny '51 transmisja szeregowa może być realizowana zupełnie inaczej.

W rodzinie '51 port szeregowy może pracować w czterech trybach. Tryby pracy różnią się między sobą rodzajem transmisji (synchroniczna, asynchroniczna), liczbą przesyłanych bitów w jednej ramce i źródłem sygnałów taktujących port szeregowy.

Tryb pracy portu szeregowego, źródło jego taktowania itd. są wybieranie poprzez odpowiednie ustawienie znaczników w rejestrze SCON:

rejestr SCON — adres 98h, rw = 0

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Rola znaczników jest następująca:

SM0, SM1 - wybór trybu pracy

SM0	SM1	tryb pracy	rodzaj transmisji	częstotliwość taktująca
0	0	0	synchroniczna, 8 bitów	$f_{osc}/12$
0	1	1	asynchroniczna, 8 bitów	zmienna
1	0	2	asynchroniczna, 9 bitów	$f_{osc}/64$ lub $f_{osc}/32$
1	1	3	asynchroniczna, 9 bitów	zmienna

SM2 - znacznik stosowany w transmisji wieloprocesorowej.

- W trybie pracy 0 znacznik nie ma wpływu na działanie portu szeregowego
- W trybie pracy 1, przy SM2 = 1, znacznik przerwania RI nie jest ustawiany w stan 1 przy braku bitu stopu.
- W trybach pracy 2 i 3, przy SM2 = 1, znacznik przerwania RI nie jest ustawiany w stan 1 jeżeli dziewiąty bit odbieranego słowa jest równy 1.

REN - znacznik uaktywnienia odbiornika. Gdy REN = 0 odbiornik portu szeregowego nie przyjmuje danych.

TB8 - znacznik używany w transmisji dziewięciobitowej. Stan tego znacznika jest wysyłany jako dziewiąty bit danych.

RB8 - w transmisji dziewięciobitowej do tego znacznika jest wpisywany stan dziewiątego bitu przyjmowanych danych. W trybie 1, gdy SM2 = 0, do RB8 jest wpisywany bit stopu.

TI - znacznik przerwania od nadajnika portu szeregowego. Jest ustawiany w stan 1 po wysłaniu ósmego bitu danych w trybie pracy 0 i na początku bitu stopu w pozostałych trybach, rysunek 12-2 i 12-4. Musi być zerowany programowo.

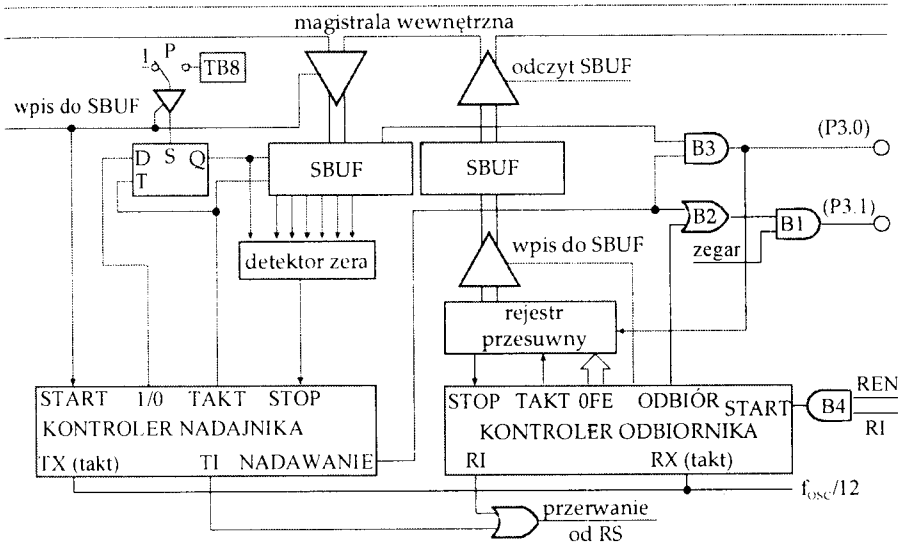
RI - znacznik przerwania od odbiornika porty szeregowego. Jest ustawiany w stan 1 po odebraniu ósmego bitu danych w trybie pracy 0, lub w połowie bitu stopu w pozostałych trybach, za wyjątkiem sytuacji związanej z ustawieniem bitu SM2.

Drugim rejestrem związanym z portem szeregowym jest rejestr SBUF, umieszczony pod adresem 99h w obszarze SFR. Jest to rejestr podwójny, do którego wpisuje się wysyłaną daną i z którego odczytuje się przyjętą daną.

a) Tryb pracy 0

W trybie pracy 0, pracy synchronicznej, nadawanie i odbieranie danych odbywa się przez końcówkę P3.0, a przez końcówkę P3.1 jest wysyłany sygnał taktujący, niezależnie, czy zachodzi nadawanie, czy odbieranie danych. Przesyłanych jest osiem bitów danych zaczynając od bitu najmniej znaczącego. Częstotliwość taktowania jest stała i wynosi $f_{osc}/12$. Schemat portu szeregowego, pracującego w trybie 0 jest przedstawiony na rysunku 12-1.

Proces wysyłania danych rozpoczyna się od momentu wpisania do rejestru SBUF wysyłanej danej. Rejestr SBUF jest rejestrem przesuwnym. Sygnał „wpis do SBUF" ustawia w stan 1 wyjście przerzutnika D oraz uaktywnia kontroler nadajnika. Po upływie jednego cyklu maszynowego następuje taktowanie kontrolera nadajnika. Na wyjściu NADAWANIE pojawia się stan 1, co umożliwia przesłanie sygnału z rejestru SBUF na końcówkę RXD i sygnału zegarowego na końcówkę TXD. Po każdym takcie następuje wysłanie kolejnego bitu z rejestru SBUF, natomiast do rejestru jest wpisywany stan wyjścia przerzutnika D. W pierwszym momencie jest to stan 1, a w pozostałych stan0. Gdy do rejestru SBUF zostanie wpisanych 6 zer następuje zadziałanie detektora zera. Na wyjście jest wysyłany jeszcze jeden bit, następuje ustawienie w stan 1 znacznika TI i zamknięcie bramek wyjściowych, co kończy proces wysyłania danych. Ustawienie znacznika TI w stan 1 jest sygnałem, że można wysłać kolejny bajt danych. Znacznik TI może być zerowany w trakcie wysyłania danych.

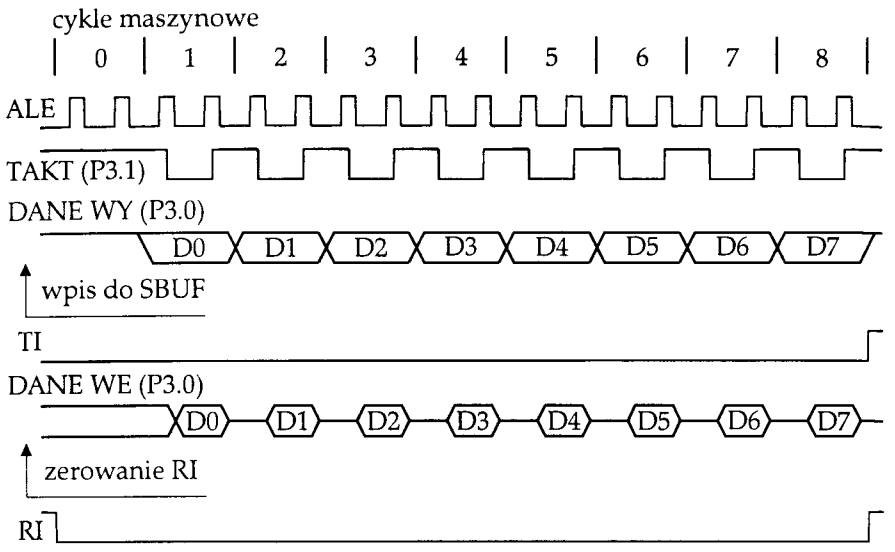


Rys. 12-1. Schemat układu portu szeregowego pracującego w trybie 0.

Odbiór danych może odbywać się pod warunkiem ustawienia znacznika REN w stan 1 i wyzerowania znacznika RI. W tym momencie do rejestru przesuwnego odbiornika jest wpisywana wartość 11111110, a z pewnym opóźnieniem sygnał ODBIÓR otwiera bramkę B1 dla sygnału zegarowego. Dana jest przyjmowana przez odbiornik portu szeregowego synchronicznie ze zboczem narastającym sygnału taktującego. Przyjęta dana jest wpisywana do rejestru przesuwnego, a zawartość rejestru jest przesuwana w lewo. Gdy zero wpisane podczas inicjalizacji odbiornika zostanie przesunięte na skrajną lewą pozycję, następuje jeszcze jeden odczyt stanu wejścia RXD, a następnie zawartość rejestru przesuwnego jest wpisywana do rejestru SBUF. Sygnał ODBIÓR jest ustawiany w stan 0, a znacznik RI w stan 1, co kończy proces odbierania danych.

Na rysunku 12-2 są pokazane przebiegi sygnałów w wybranych punktach układu nadajnika i odbiornika.

Gdy na wyjściu P3.0 znajduje się wysyłana dana, sygnał taktujący przechodzi ze stanu 1 do stanu 0 i ponownie do stanu 1. Umożliwia to współpracę mikrokontrolera z układami zewnętrznymi, które reagują na zbocze narastające lub opadające sygnału taktującego.



Rys. 12-2. Wysyłanie i przyjmowanie danych w trybie pracy synchronicznej

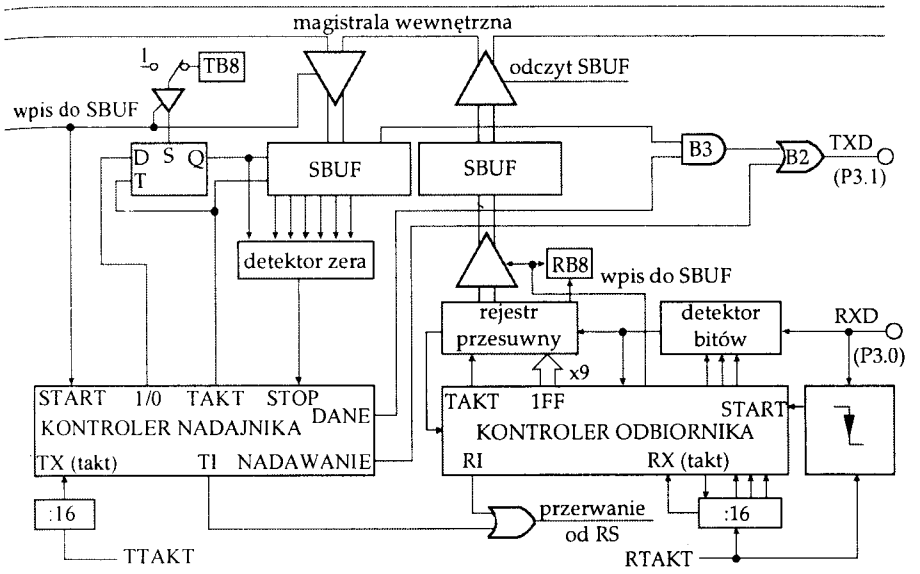
Transmisja synchroniczna ma zastosowanie przy przesyłaniu danych pomiędzy mikrokontrolerami, a modułami zewnętrznymi takimi jak sterowniki pól odczytowych LCD lub LED, przetworniki A/C i C/A potencjometry cyfrowe, itp.

b) Tryb pracy 1.

W trybie pracy 1 dziesięć bitów jest wysyłanych przez wyprowadzenie TXD (P3.1) lub przyjmowanych przez wyprowadzenie RXD (P3.0). Są to: bit startu (stan 0), 8 bitów danych i bit stopu (stan 1). We wszystkich mikrokontrolerach rodziny '51 port szeregowy w tym trybie pracy może być taktowany z wyjścia licznika T1. Ponadto w mikrokontrolerach 8xC52 można go taktować z licznika T2, a w mikrokontrolerach 80C515/535 z dodatkowego dzielnika :39. Schemat układu portu szeregowego pracującego w trybach transmisji asynchronicznej jest przedstawiony na rysunku 12-3.

Nadawanie rozpoczyna się w momencie przesłania do rejestru SBUF wysyłanej danej. Sygnał „wpis do SBUF". wpisuje 1 do przerzutnika D oraz uaktywnia kontroler nadajnika. Transmisja rozpoczyna się z momentem przepełnienia dzielnika :16. Na wyjściu NADAWANIE pojawia się stan 0 co powoduje wygenerowanie bitu startu (przy stanie 0 na wyjściu DANE). Kolejne przepełnienie licznika :16 zmienia stan na wyjściu DANE na 1 dzięki czemu jest otwarta bramka B3 dla bitów z rejestru SBUF. Proces wysyłania bitów danych jest dalej taki sam jak przy transmisji synchronicznej. Gdy zostanie wysłany ostatni bit danych na wyjściu DANE jest ustawiany stan0, a na wyjściu NADAWANIE

stan1, co powoduje wytworzenie bitu stopu. Jednocześnie w stan 1 jest ustawiany znacznik TI. Jest to informacja o zakończeniu procesu wysyłania danych.

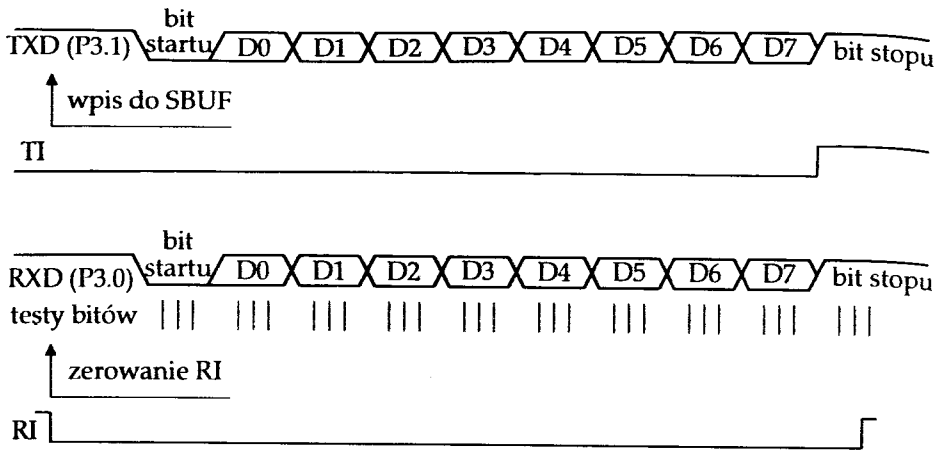


Rys. 12-3. Schemat układu portu szeregowego pracującego w trybach transmisji asynchronicznej.

Odbiór danej jest możliwy po ustawieniu w stan 1 znacznika REN oraz wyzerowaniu znacznika RI i rozpoczyna się w momencie wykrycia na wyprowadzeniu RXD zmiany stany z 1 na 0, co powoduje wyzerowanie dzielnika :16 i wpisanie do rejestru przesuwnego odbiornika stanu 111111111. Zerowanie dzielnika powoduje, że nadchodzące kolejno bity są testowane synchronicznie względem bitu startu.

Stan wejścia RXD jest testowany trzykrotnie w okresie czasu przeznaczonym dla każdego bitu, dzięki czemu zwiększa się odporność odbiornika na zakłócenia. Jeżeli test nie będzie pozytywny, to odbiornik rozpoczyna proces odbierania danych od początku oczekując na zmianę stanu na wejściu RXD z 1 na 0. Przy pozytywnym teście przyjęty bit danej jest wpisywany do rejestru przesuwnego, a zawartość rejestru jest przesuwana w lewo. Bity są przyjmowane tak długo, aż bit startu znajdzie się na wyjściu rejestru przesuwnego. W tym momencie, podobnie jak przy transmisji synchronicznej, nastąpi przyjęcie jeszcze jednego bitu, po czym nastąpi przepisanie przyjętej danej do rejestru SBUF oraz ustawienie znacznika RI w stan 1, co kończy proces przyjmowania danej przez port szeregowy. Bit stopu jest wpisywany do znacznika RB8.

Dana z rejestru SBUF musi być odczytana przed nadejściem bitu stopu kolejnej danej, inaczej będzie stracona.



Rys. 12-4. Transmisja danych w trybie pracy 1.

Wpis do SBUF, RB8 i ustawienie znacznika w stan 1 odbywa się wtedy i tylko wtedy, gdy są spełnione warunki:

RI = 0 **SM2 = 0 lub bit stopu = 1**

c) Tryby pracy 2 i 3.

W trybie pracy 2 i 3 proces wysyłania i przyjmowania danych jest taki sam jak w trybie 1, z tym że oprócz bitów startu i stopu jest przesyłanych 9 bitów danych. Przy nadawaniu, osiem bitów jest pobieranych z rejestru SBUF, a ostatni - 9 bit ze znacznika TB8. Dlatego przed wpisaniem danej do rejestru SBUF, co rozpoczyna proces wysyłania danych, należy najpierw odpowiednio ustawić znacznik TB8. Po odebraniu danych przez odbiornik portu szeregowego, oprócz odczytania zawartości rejestru SBUF należy również odczytać znacznik RB8.

Różnica między trybem pracy 2 i 3 polega na tym, że w trybie 2 prędkość transmisji, w zależności od ustawienia znacznika SMOD, może wynosić tylko 1/32 (SMOD = 1) lub 1/64 (SMOD = 0) częstotliwości oscylatora mikrokontrolera. Znacznik SMOD jest najbardziej znaczącym bitem w rejestrze PCON, który znajduje się pod adresem 87h w obszarze SFR. Dlatego ustawienie tego znacznika wymaga operacji bajtowych.

W trybie pracy 3 taktowanie portu szeregowego odbywa się tak jak w trybie 1. Tryb ten jest najczęściej stosowany przy połączeniach mikrokontrolerów ze sobą lub z komputerem, gdyż daje możliwość uzyskania jednej ze standardowych prędkości transmisji, a dziewiąty bit może być bitem parzystości ułatwiając w ten sposób kontrolę poprawności przesyłania danych.

Wybór trybu pracy 1 lub 3 powoduje, że nadajnik i odbiornik portu szeregowego jest taktowany z wyjścia licznika T1, chyba że konkretny mikrokontroler posiada możliwości taktowania z innego źródła.

W mikrokontrolerach 8xC52 takim dodatkowym źródłem taktującym jest licznik T2. W rejestrze T2CON tych mikrokontrolerów, który znajduje się pod adresem 0C8h w obszarze SFR, znajdują się znaczniki RCLK i TCLK odpowiadające za dołączenie wyjścia licznika T2 do nadajnika (TCLK = 1) i odbiornika (RCLK = 1) portu szeregowego. Są to jedyne mikrokontrolery rodziny '51, w których prędkości nadawania i odbierania danych mogą się różnić. Na przykład, przy ustawieniu znaczników RCLK = 1 i TCLK = 0 odbiornik będzie taktowany z licznika T2, a nadajnik z licznika T1.

W mikrokontrolerach 80C515/535 dodatkowym źródłem taktowania jest dzielnik :39. Jest on włączany znacznikiem BD, który jest najbardziej znaczącym bitem rejestru ADCON znajdującego się w obszarze SFR pod adresem 0D8h.

Jeżeli prędkość transmisji ma odpowiadać jednej z prędkości standardowych, to stopień podziału licznika i częstotliwość oscylatora powinny być tak dobrane, by rzeczywista prędkość transmisji nie różniła się od nominalnej więcej iż o ± 1,8%.

W trybie pracy 0 prędkość transmisji jest określona zależnością:

$$f_{tr} = \frac{f_{osc}}{12}$$

W trybie pracy 2 prędkość transmisji zależy od ustawienia znacznika SMOD:

$$f_{tr} = \frac{2^{SMOD}}{64} \cdot f_{osc}$$

W trybach pracy 1 i 3, przy taktowaniu portu szeregowego z licznika T1 pracującego w trybie 2 (8-bitowy licznik z autoładowaniem) prędkość transmisji jest określona zależnością:

$$f_{tr} = \frac{2^{SMOD}}{384} \cdot \frac{f_{osc}}{[256 - (TH1)]}$$

gdzie: (TH1) - liczba wpisana do rejestru TH1

Licznik T1 może również taktować port szeregowy pracując w pozostałych trybach, z sygnałem zewnętrznym lub wewnętrznym. W takim przypadku do ponownego załadowania licznika należy wykorzystać przerwanie wywołane przepełnieniem licznika, a czas obsługi przerwania musi być uwzględniony przy obliczaniu częstotliwości przepełnień licznika.

Jeżeli w mikrokontrolerach 8xC52 jako generator zostanie użyty licznik T2, to w rejestrze T2CON należy odpowiednio ustawić znaczniki RCLK, TCLK oraz znacznik C/T2 (taktowanie licznika impulsami zewnętrznymi C/T2 = 1 lub wewnętrznymi C/T2 = 0). Gdy licznik T2 pracuje jako 16-bitowy dzielnik z autoładowaniem i taktowaniem wewnętrznym ($f_{osc}/2$), to częstotliwość transmisji jest określona zależnością:

$$f_{tr} = \frac{f_{osc}}{32 \cdot [65536 - (RCAP2H, RCAP2L)]}$$

gdzie: (RCAP2H,RCAP2L) - liczba wpisywana do rejestru bardziej znaczącego RCAP2H i mniej znaczącego RCAP2L licznika T2.

W mikrokontrolerach 80C515/535, przy taktowaniu portu szeregowego z dzielnika :39 zależność na prędkość transmisji jest następująca:

$$f_{tr} = f_{osc} \cdot \frac{2^{SMOD}}{2496}$$

gdzie: f_{tr} - prędkość transmisji
f_{OSC} - częstotliwość oscylatora mikrokontrolera

Poniższy program umożliwia obserwację na oscyloskopie sygnałów wysyłanych przez port szeregowy mikrokontrolera 80C51. Przyjęto, że częstotliwość oscylatora mikrokontrolera wynosi 12 MHz, a prędkość transmisji wysyłanych znaków 1200 bodów (1200 bitów/sekundę). Przez port szeregowy jest wysyłany kod ASCII litery a. Po wysłaniu każdego znaku jest wprowadzona przerwa w transmisji, by łatwiej uchwycić początek transmisji. Wyliczenie wartości podziałowej wpisywanej do licznika T1 przeprowadzono na podstawie zależności podanej wyżej, a samo obliczenie wykonuje asembler.

```
;*****************************************************************
; Program wysyłania znaku litery a przez port szeregowy.
; Prędkość transmisji 2400 bodów,
; Częstotliwość oscylatora 12 MHz,
; Port pracuje w trybie 1, taktowanie z licznika T1,
; Opóźnienie zrealizowane na liczniku T0
;*****************************************************************
DANA    EQU  'a'              ;kod ASCII litery a
FTR     EQU  2400             ;prędkość transmisji
FOSC    EQU  12000000         ;częstotliwość zegara mikrokontrolera
NTH1    EQU  256-FOSC/(FTR*32*12)   ;wartość określająca częstotliwość
                              ;taktowania portu szeregowego
SET_T0T1 EQU 21H              ;tryby pracy liczników: T0 - tryb 1, T1 - tryb 2
                              ;taktowanie wewnętrzne
```



```
SET_RS  EQU  40H              ;port szeregowy tryb 1
        CSEG
        ORG  0                ;adres początku programu
    MOV  TMOD,#SET_T0T1
    MOV  TH1,#NTH1            ;wpis wartości podziałowej
    MOV  SCON,#SET_RS
    SETB TR1                  ;start licznika T1
PETLA:
    MOV  SBUF,#DANA           ;wysłanie danej
    JNB  TI,$                 ;oczekiwanie na koniec transmisji znaku
    CLR  TI                   ;kasowanie znacznika końca transmisji
;opóźnienie
    SETB TR0                  ;start licznika T0
    JNB  TF0,$                ;oczekiwanie na przepełnienie licznika T0
    CLR  TR0                  ;zatrzymanie licznika T0
    CLR  TF0                  ;kasowanie znacznika przepełnienia licznika
    SJMP PETLA                ;skok do ponownego wysłania znaku
END
```

Po uruchomieniu programu będzie on działał bez przerwy aż do wyzerowania mikrokontrolera.

Program z tego przykładu wysyła i przyjmuje przez port szeregowy tego samego mikrokontrolera łańcuch znaków. Łańcuch ten jest umieszczony w pamięci programu i jest przepisywany do tablicy umieszczonej w pamięci wewnętrznej RAM mikrokontrolera. Adres początku tablicy przyjmowanych danych jest umieszczony w rejestrze R0. Port szeregowy pracuje w trybie 1, taktowany z licznika T1. Prędkość transmisji 9600 bodów. Podział licznika został tak dobrany, by uzyskać minimalną częstotliwość oscylatora. Kolejny znak jest wysyłany po odbiorze poprzedniego. Znakiem końca łańcucha jest znak $. Żeby przesłać dane należy połączyć wyjście nadajnika (P3.1) z wejściem odbiornika (P3.0).

Minimalną częstotliwość oscylatora uzyskuje się przy podziale licznika T1 = 1 (TH1 = 255). Standardowo częstotliwość oscylatora dla mikrokontrolerów 80C51 nie powinna być mniejsza od 2,5 MHz. Przy tych założeniach i przy SMOD = 0 ze wzoru na częstotliwość transmisji portu szeregowego pracującego w trybie 1 częstotliwość kwarcu powinna wynosić 3,686 MHz.

```
;*****************************************************************
;Program przesyłający przez port szeregowy ciąg znaków z tablicy umieszczo-
;nej w pamięci programu do pamięci RAM. Początek tablicy w pamięci RAM ;od
adresu 20H. Prędkość transmisji 9600 bodów, częstotliwość kwarcu 3,686 MHz.
Użyte rejestry: R0, DPTR
;*****************************************************************
SET_RS    EQU   40h     ;tryb 1 pracy portu szeregowego
TAB_RAM   DATA  20h     ;adres tablicy przyjmowanych danych
L_PODZ    EQU   0FFh    ;liczba podziałowa wpisywana do rejestru TH1
SET_T1    EQU   20h     ;tryb 1 licznika T1, taktowanie wewnętrzne

      CSEG
      ORG   0

   MOV   SCON,#SET_RS
   MOV   TMOD,#SET_T1   ;tryb pracy licznika T1
   MOV   TH1,#L_PODZ    ;podział licznika T1
   SETB  TR1            ;start licznika T1
   SETB  REN            ;odblokowanie odbiornika
   MOV   R0,#TAB_RAM    ;adres tablicy przyjmowanych danych
   MOV   DPTR,#TAB      ;adres tablicy wysyłanych danych

N_ZN:
   CLR   A              ;zerowanie akumulatora
   MOVC  A,@A+DPTR      ;odczytanie danej z tablicy
   MOV   SBUF,A         ;wysłanie znaku
   JNB   RI,$           ;oczekiwanie na przyjęcie danej
   CLR   RI             ;kasowanie znacznika końca odbioru bajtu
   CLR   TI             ;kasowanie znacznika końca nadawania bajtu
   MOV   A,SBUF         ;odczytanie przyjętej danej
   MOV   @R0,A          ;wysłanie przyjętej danej do pamięci
   INC   R0             ;kolejny adres tablicy przyjmowanych danych
   INC   DPTR           ;kolejny adres tablicy wysyłanych danych
   CJNE  A,#'$',N_ZN    ;test czy ostatni znak
   CLR   REN            ;zablokowanie odbiornika portu szeregowego
   CLR   TR1            ;wyłączenie licznika T1
   AJMP  $              ;pętla końca programu

TAB:
   DB    'TEST PORTU SZEREGOWEGO$'
END
```

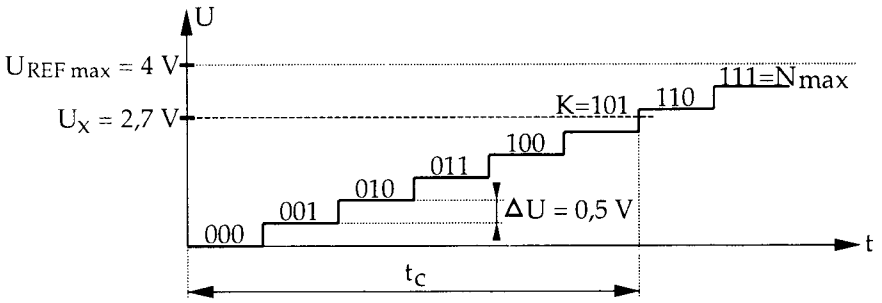

Pytania i problemy.

1. Do czego służy port szeregowy ?
2. Czym się różni transmisja synchroniczna od asynchronicznej ?
3. Czym charakteryzują się poszczególne tryby pracy portu szeregowego w mikrokontrolerach rodziny '51 ?
4. Jaka jest rola znaczników obsługujących port szeregowy ?
5. Do czego służy rejestr SBUF ?
6. Jakie są źródła taktowania portu szeregowego ?
7. W którym mikrokontrolerze można uzyskać różne prędkości nadawania i odbioru danych ? Jak to się realizuje ?
8. Zmodyfikować podany wyżej przykład tak, wysłać również bit parzystości wysyłanego znaku.
9. Zmodyfikować przykład tak, by port szeregowy pracował w trybie synchronicznym.
10. Zmodyfikować powyższy program tak, by opóźnienie było realizowane w podprogramie obsługi przerwania.

13. Przetwornik analogowo-cyfrowy w mikrokontrolerze SAB 80515/535

Mikrokontrolery 80515/535 wyposażone są w 8-wejściowy, 8-bitowy przetwornik analogowo-cyfrowy przeznaczony do pomiarów napięcia stałego lub wartości chwilowej napięcia zmiennego. W czasie przetwarzania analogowo-cyfrowego przetwornik realizuje trzy podstawowe operacje:

- *kwantowanie* polegające na porównaniu wartości ciągłego sygnału analogowego U_X z podzieloną na przedziały ΔU wartością wzorcowego napięcia U_{REF} (rysunek 13-1). Wynikiem kwantowania jest liczba K poziomów kwantowania ΔU, dla której wystąpiła równość $U_X = K * \Delta U$. Ponieważ porównywana jest całkowita liczba poziomów kwantowania K, dlatego podana równość spełniona jest z dokładnością $\pm\frac{1}{2}\Delta U$. Pojedynczy poziom kwantowania ΔU określany jest nazwą 1 LSB (Least Significant Bit) i odnosi się do sposobu kodowania wyniku przetwarzania.

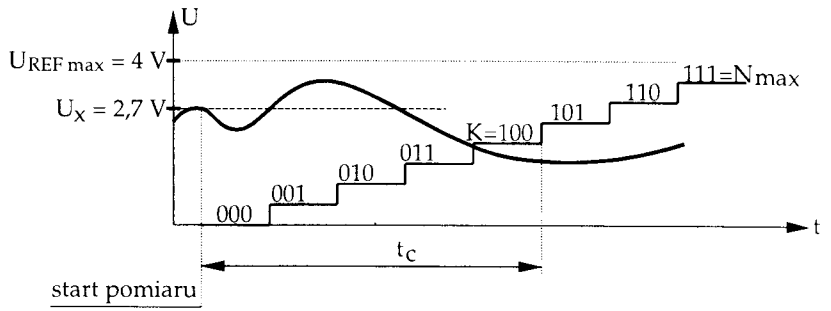


Rys. 13-1 Zasada kwantowania napięcia U_X.

- *kodowanie*, które jest przyporządkowaniem liczbie K poziomów kwantowania określonej kombinacji cyfr, najczęściej kombinacji binarnej. Taka kombinacja cyfr jest wynikiem przetwarzania analogowo-cyfrowego. Nie należy zapominać, że kodowaniu poddawane są wartości uprzednio skwantowane. Przykładowo przy pomiarze napięcia $U_X = 2{,}7$ V przetwornikiem o rozdzielczości 3-bitów, w którym pojedynczemu poziomowi kwantowania 1 LSB odpowiada napięcie $\Delta U = 0{,}5$ V, wynikiem przetwarzania jest liczba 5 (rysunek 13-1). W ogólnym przypadku, w N-bitowym przetworniku i zakresie przetwarzania U_{REFmax} pojedynczy poziom kwantowania ma wartość:

$$1\ \text{LSB} = \frac{U_{REFmax}}{2^N}$$

- *próbkowanie* stosowane w przypadku sygnałów zmieniających się w czasie. Kwantowanie sygnału analogowego i kodowanie wyniku wymaga czasu t_C, w którym wartość sygnału nie powinna ulegać zmianie. Zmiana tej wartości, jak na rysunku 13-2, powoduje powstanie dodatkowych błędów dynamicznych. Zmniejszenie tych błędów jest możliwe jeśli wykorzystuje się przetworniki A/C o krótkim czasie przetwarzania lub stosuje się dodatkowe układy próbkująco-pamiętające (Sample & Hold). Zadaniem tych układów jest zapamiętanie (w krótkim czasie) wartości chwilowej mierzonego sygnału na długi czas t_C przetwarzania przetwornika.



Rys. 13-2 Próbkowanie zmieniającego się napięcia U_X

Zamiana wartości mierzonego napięcia na odpowiadający kod cyfrowy w mikrokontrolerach 80515/535 dokonywana jest w oparciu o metodę kompensacji wagowej z wykorzystaniem matrycy kondensatorów jako elementu kompensującego. Dokładniej zasada pomiaru opisana jest w [2].

Przetworniki analogowo-cyfrowe w mikrokontrolerach 80515/535 firmy Siemens (mikrokontrolery wykonywane są w dwóch wersjach technologicznych: MYMOS - SAB 80515/535 i ACMOS - SAB 80C515/535) charakteryzują się następującymi parametrami:

- przetwarzanie U_X = 0..+5 V w 16 programowalnych podzakresach,
- 8-bitowa rozdzielczość przetwornika,
- 8 kanałów analogowych, a w SAB 80C515/535 dodatkowo port P6,
- czas przetwarzania analogowo-cyfrowego:
  → 15 cykli maszynowych w procesorze SAB 80515/535,
  → 13 cykli maszynowych w procesorze SAB 80C515/535,
- programowe wyzwalanie każdego pomiaru lub serii pomiarów,
- możliwość generowania przerwania po każdym pomiarze napięcia,
- wykorzystywane przez przetwornik rejestry specjalne SFR:
  → ADCON (adres 0D8h) - rejestr sterujący pracą przetwornika,
  → ADDAT (adres 0D9h) - 8 bitowy rejestr wyniku przetwarzania

  → DAPR (adres 0DAh) - rejestr programowania wzorcowych napięć $U_{IntAREF}$ i $U_{IntAGND}$.

Wewnętrzną strukturę przetwornika analogowo-cyfrowego dla obu typów mikrokontrolerów przedstawiono na rysunku 13-3.

Przy programowaniu trybu pracy przetwornika i numeru kanału analogowego należy zwrócić uwagę na znaczenie bitów BD i CLK w rejestrze ADCON:

- bit BD używany jest do określenia szybkości transmisji łącza szeregowego; w wielu systemach uruchomieniowych szybkość wymiany danych między komputerem i systemem mikroprocesorowym ustalana jest za pośrednictwem bitu BD, zmiana stanu bitu BD powoduje zmianę szybkości transmitowanych danych i 'zawieszenie' systemu,
- bit CLK umożliwia uzyskanie na wyprowadzeniu procesora P1.6 (oznaczonym symbolem CLKOUT) sygnału o częstotliwości $f_{CLKOUT} = \frac{f_{GEN}}{12}$ i wypełnieniu $\frac{1}{6}$ (f_{GEN} jest częstotliwością rezonatora kwarcowego dołączonego do mikrokontrolera).

13.1 Wybór wejścia pomiarowego

W mikrokontrolerach SAB 80(C)515/535 mierzone napięcie U_X dołączane jest do jednego z 8 wejść analogowych przetwornika oznaczonych symbolami AN0..AN7.

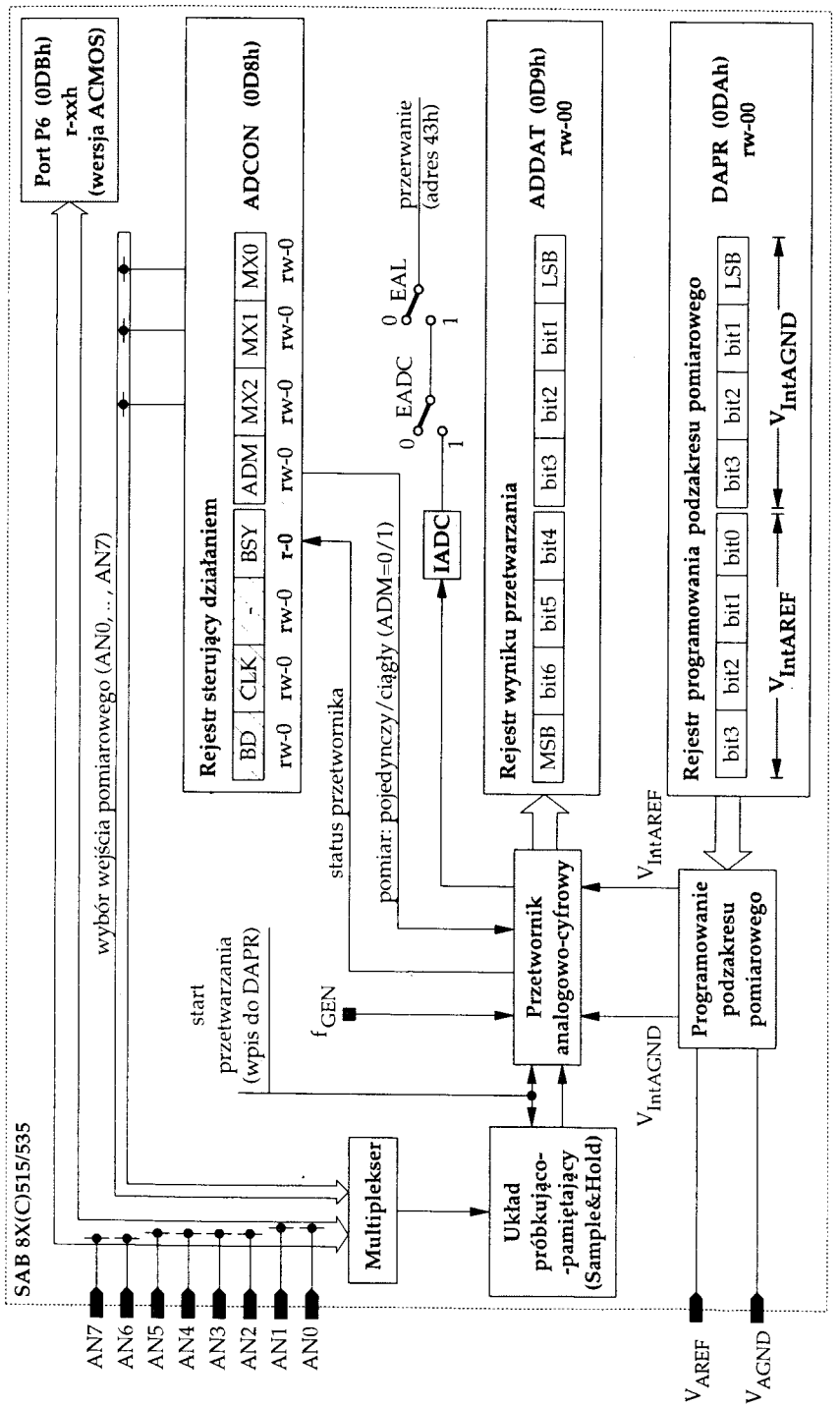
Wybór numeru wejścia pomiarowego, numeru kanału analogowego jest dokonywany przez wpisanie 3-bitowej wartości do rejestru ADCON (A/D converter CONtrol register) znajdującego się w rejestrach specjalnych SFR (MX0 .. MX2).

ADCON rejestr sterujący przetwornikiem (adres 0D8h):

BD	CLK	-	BSY	ADM	MX2	MX1	MX0
rw-0	rw-0	rw-0	**r-0**	rw-0	rw-0	rw-0	rw-0

Znaczenie poszczególnych bitów w rejestrze ADCON jest następujące:

- BSY - stan przetwornika w trakcie przetwarzania (BuSY flag), znacznik tylko do odczytu, ustawiany i kasowany sprzętowo:
  BSY = 1 w trakcie przetwarzania,
  BSY = 0 po zakończeniu przetwarzania,



Rys. 13-3 Struktura wewnętrzna przetwornika A/C.

- ADM - rodzaj przetwarzania (A/D conversion Mode):
  ADM = 1 seria pomiarów, przetwornik po uruchomieniu wykonuje pomiary, aż do chwili, w której ADM = 0,
  ADM = 0 pomiar pojedynczy, uruchamiany programowo,

- MX2..MX0 - wybór numeru kanału pomiarowego (analog MultipleXer) wg. danych podanych w tabeli 13-1.

Tabela 13-1 Sterowanie wyborem kanałów analogowych AN0..AN7.

MX2	MX1	MX0	Kanał	MYMOS	ACMOS
0	0	0	0	AN0	AN0/P6.0
0	0	1	1	AN1	AN1/P6.1
0	1	0	2	AN2	AN2/P6.2
0	1	1	3	AN3	AN3/P6.3
1	0	0	4	AN4	AN4/P6.4
1	0	1	5	AN5	AN5/P6.5
1	1	0	6	AN6	AN6/P6.6
1	1	1	7	AN7	AN7/P6.7

W procesorach wykonanych w technologii ACMOS (SAB 80C515/535) wejścia AN0..AN7 są równocześnie wejściami cyfrowego portu P6 (o adresie 0DBH). Oznacza to z jednej strony możliwość pomiaru wartości wejściowego sygnału cyfrowego, a z drugiej interpretację logiczną podanego sygnału.

13.2 Jaki podzakres pomiarowy ?

W mikrokontrolerach SAB 80(C)515/535 istnieje możliwość programowania podzakresu pomiarowego, który ustalany jest przez podział różnicy napięć doprowadzonych do wejścia V_{AREF} i napięcia V_{AGND}. W ten sposób tworzone są dwa wewnętrzne napięcia wzorcowe $V_{IntAREF}$ i $V_{IntAGND}$, które określają górną i dolną wartość napięcia podzakresu pomiarowego.

$$V_{IntAREF} = V_{AGND} + \frac{DAPR_{7..4}}{16}(V_{AREF} - V_{AGND}) \qquad \text{dla } DAPR_{7..4} > 3$$

$$V_{IntAGND} = V_{AGND} + \frac{DAPR_{3..0}}{16}(V_{AREF} - V_{AGND}) \qquad \text{dla } DAPR_{3..0} < 0DH$$

gdzie:

$DAPR_{7..4}$ są czterema najbardziej znaczącymi bitami, rejestru DAPR (D/A converter Program Register),

$DAPR_{3..0}$ są czterema najmniej znaczącymi bitami rejestru DAPR:

DAPR - rejestr programowania podzakresu pomiarowego i startu przetwornika (adres 0DAh), rw-00:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Napięcie $V_{IntAREF}$				Napięcie $V_{IntAGND}$			

Ze względu na konstrukcję wewnętrzną przetwornika analogowo-cyfrowego ustalając podzakres pomiarowy konieczne jest spełnienie dodatkowych warunków:

- tylko w czasie pracy przetwornika do wyprowadzeń procesora V_{AREF} i V_{AGND} musi być dołączone napięcie,
- jeśli napięcia V_{AREF} i V_{AGND} dołączone są do napięć zasilających mikrokontroler V_{CC} i V_{SS} to:
  $V_{AREF} = V_{CC} \pm 5\ \%$ i $V_{AGND} = V_{SS} \pm 0.2$ V,
- rezystancja wewnętrzna źródła napięcia wzorcowego R_{REF} (i źródła mierzonego napięcia R_X) nie może przekraczać wartości 5 kΩ,
- minimalna różnica wewnętrznych napięć $V_{IntAREF}$ i $V_{IntAGND}$ nie może być mniejsza niż 1 V ($V_{IntAREF} - V_{IntAGND} \geq 1$ V),
- jeśli napięcia: V_{AGND}= 0 V i V_{AREF}= +5 V, to graniczne napięcia w poszczególnych podzakresach pomiarowych podano w tabeli 12-2.

13-2

Analizując wartości napięć obu krańców podzakresów pomiarowych należy zwrócić uwagę na zerowe wartości bitów rejestru $DAPR_{7..4}$, którym odpowiada również zerowa wartość napięcia $V_{IntAREF}$. Jest to jedyny wyjątek w tabeli 13-2.

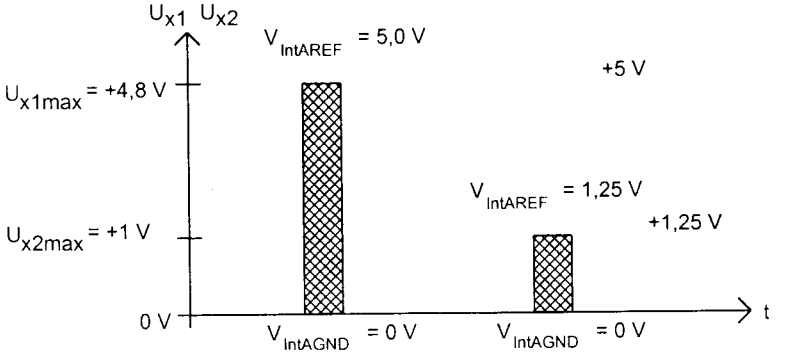
Programowa zmiana podzakresu pomiarowego umożliwia wybranie optymalnego podzakresu w stosunku do wartości i dopuszczalnej zmiany wartości mierzonego napięcia. Mierząc na przykład dwa różne napięcia:

- napięcie U_{x1} zmieniające się w zakresie 0..+4,8 V, dołączone do wejścia AN3,
- napięcie U_{x2} zmieniające się w zakresie 0..+1 V, dołączone do wejścia AN4

należy wybrać z 16 możliwych podzakresów dwa najbardziej odpowiednie. W przypadku pomiaru napięcia U_{x1} najwłaściwszym jest wybranie pełnego zakresu przetwarzania przetwornika 0..+5 V, w drugim przypadku, pomiaru napięcia U_{x2}, podzakresu 0..1,25 V (rysunek 13-4).

Tabela 13-2 Sterowanie podzakresami pomiarowymi.

Podzakres pomiarowy	Stan rejestru DAPR: $DAPR_{3..0}$ $DAPR_{7..4}$	Dolna wartość napięcia podzakresu pomiarowego: $V_{IntAGND}$ [V]	Górna wartość napięcia podzakresu pomiarowego: $V_{IntAREF}$ [V]
0	0000	0.0	5.0
1	0001	0.3125	-
2	0010	0.625	-
3	0011	0.9375	-
4	0100	1.25	1.25
5	0101	1.5625	1.5625
6	0110	1.875	1.875
7	0111	2.1875	2.1875
8	1000	2.5	2.5
9	1001	2.8125	2.8125
10	1010	3.125	3.125
11	1011	3.4375	3.4375
12	1100	3.75	3.75
13	1101	-	4.0625
14	1110	-	4.375
15	1111	-	4.6875



Rys. 13-4 Dobór podzakresu pomiarowego do zakresu zmian mierzonych napięć.

Konsekwencją takiego wyboru jest pomiar napięcia U_{x1} z rozdzielczością 8-bitów i pomiar napięcia U_{x2} również z rozdzielczością 8-bitów ale w węższym podzakresie 0..1,25 V. W pierwszym przypadku jeden poziom kwantowania 1 LSB_{x1} odpowiada napięciu:

$$1\ LSB(x1) = \frac{V_{IntAREF}(x1) - V_{IntAGND}(x1)}{2^8} = \frac{5\ V - 0\ V}{256} = 19{,}531\ mV,$$

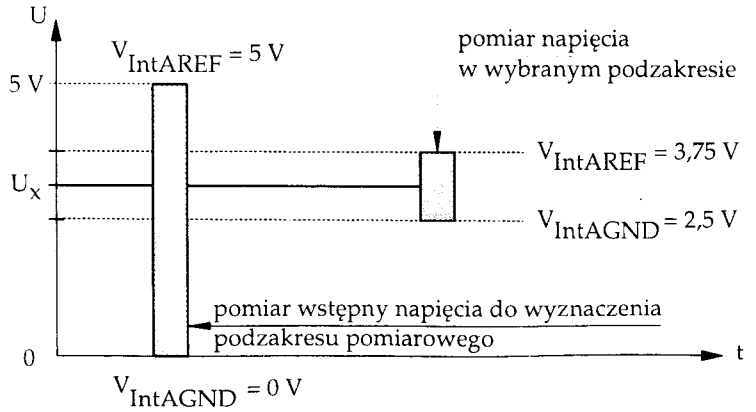
a w drugim przypadku:

$$1\ LSB(x2) = \frac{V_{IntAREF}(x2) - V_{IntAGND}(x2)}{2^8} = \frac{1{,}25\ V - 0\ V}{256} = 4{,}883\ mV$$

Zmniejszenie podzakresu pomiarowego zwiększa rozdzielczość wykonywanego pomiaru w stosunku do maksymalnego zakresu przetwornika, napięcia 0..5 V. Ten przykład pokazuje w jaki sposób zwiększyć rozdzielczość przetwornika analogowo-cyfrowego o dwa dodatkowe bity. Jeśli istnieje potrzeba pomiaru z rozdzielczością 10-bitów, to pomiar dowolnego napięcia U_X wykonywany jest dwukrotnie, tak jak na rysunku 13-5. W pierwszym pomiarze wybrany jest pełny zakres przetwarzania 0..+5 V i na podstawie wyniku pomiaru dobierany jest właściwy podzakres:

- różnica górnej i dolnej wartości napięcia wzorcowego, podzakresu $V_{IntAREF} - V_{IntAGND} \geq 1$ V,
- mierzone napięcie U_X nie może znajdować się na granicy podzakresu, co zafałszowuje wynik drugiego pomiaru.

Następnie wykonywany jest drugi pomiar. Wynik pomiaru wartości napięcia U_X obliczany jest na podstawie obu pomiarów. Jak zrealizować program pomiaru napięcia z 10-bitową rozdzielczością przedstawiono w [2 i 4].



Rys. 13-5 Dwuetapowy pomiar napięcia z rozdzielczością 10 bitów.

13.3 Pomiar

Wykonanie pojedynczego pomiaru lub serii pomiarów uzależnione jest od stanu bitu ADM znajdującego się w rejestrze ADCON:

- jeśli bit ADM = 0 to przetwornik po uruchomieniu wykonuje jeden pomiar,
- jeśli bit ADM = 1 to przetwornik po uruchomieniu wykonuje serię pomiarów tak długo jak długo znacznik ADM ma stan jedynki logicznej.

Uruchomienie czyli start przetwornika realizowane jest programowo przez wpisanie dowolnej wartości do rejestru DAPR, rejestru odpowiadającego za wybór podzakresu pomiarowego. Wynik pomiaru dostępny jest w rejestrze ADDAT. Jeśli w trakcie trwania pomiaru nastąpi jego ponowne wyzwolenie to proces przetwarzania analogowo-cyfrowego rozpoczyna się od początku.

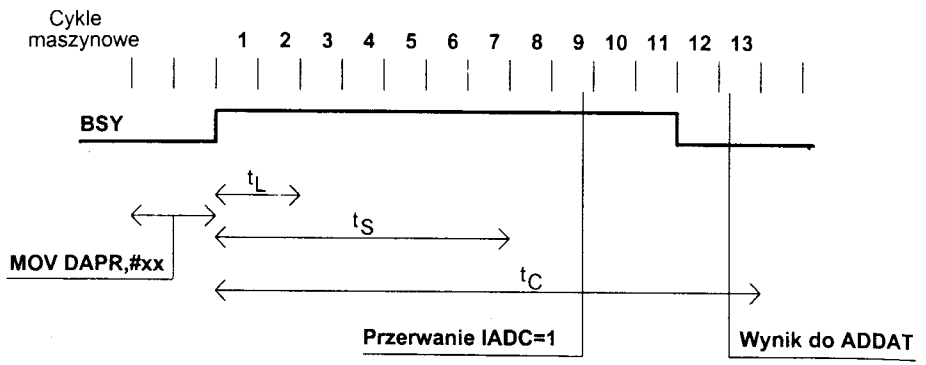
Przetwarzanie analogowo-cyfrowe składa się z trzech faz, które przedstawione zostały na rysunku 13-6:

- czasu ładowania t_L pojemności wejściowej układu próbkująco-pamiętającego C_{IO} = 10 pF do wartości mierzonego napięcia U_X trwa 2 cykle maszynowe,
- czasu próbkowania t_S zapamiętanego w pierwszej fazie napięcia U_X, dołączenie wewnętrznej matrycy kondensatorów i rozpoczęcie przetwarzania analogowo-cyfrowego,
- czasu przetwarzania t_C, zamiany wartości mierzonego napięcia na kod cyfrowy; całkowity czas przetwarzania t_C obejmuje czas ładowania pojemności wejściowej t_L i czas próbkowania t_S.

Zależności przedstawione na rysunku 13-6 dotyczą procesora SAB 80C515/535, w którym czas przetwarzania t_C równy jest 13 cyklom maszynowym (w SAB 80515/535 przetwarzanie trwa 15 cykli maszynowych). Z przedstawionych zależności wynikają trzy bardzo ważne informacje:

- przetwornik zgłasza zakończenie przetwarzania ustawiając znacznik przerwania IADC (w rejestrze IRCON, o 4 cykle maszynowe szybciej niż nastąpi rzeczywiste zakończenie przetwarzania; to przyspieszenie jest możliwe, jeśli rozważy się sposób przyjmowania przerwań i wykonanie pierwszej instrukcji w procedurze obsługi przerwania:
  → testowanie wewnętrznego priorytetu przerwań trwa 1 cykl maszynowy,
  → na wywołanie procedury obsługi przerwania od adresu 43h potrzebne są 2 cykle maszynowe,

→ najkrótsza instrukcja, np. przesłania wyniku pomiaru do akumulatora A (MOV A,ADDAT) wykonywana jest w 1 cyklu maszynowym. Znacznik przerwania IADC w rejestrze IRCON (o adresie 0C0h) musi być kasowany programowo w trakcie obsługi przerwania.



Rys. 13-6 Zależności czasowe w trakcie pracy przetwornika analogowo-cyfrowego w mikrokontrolerze SAB 80C515/535.

- przetwornik zgłasza zakończenie przetwarzania ustawiając znacznik stanu przetwarzania BSY w stan zera logicznego 2 cykle maszynowe przed zakończeniem przetwarzania; taka sytuacja wynika z faktu, że w programie do testowania znacznika BSY i pobrania wyniku przetwarzania wykorzystywane są najczęściej instrukcje:

```
Czekaj: JB    BSY,Czekaj    ;jeśli BSY = 1 to skok do adresu Czekaj
        MOV   A,ADDAT       ;A ← (ADDAT)
```

  Czas pierwszej instrukcji wynosi 2 cykle maszynowe, a drugiej 1 cykl maszynowy

- wynik przetwarzania wpisywany jest do rejestru ADDAT w pierwszej części ostatniego cyklu maszynowego, cyklu, w którym w jego drugiej części może być pobrana zawartość jednego z podanych rejestrów, np. MOV A,ADDAT. Ta dwu bajtowa instrukcja wykonywana jest w ciągu jednego cyklu maszynowego.

Wynik przetwarzania analogowo-cyfrowego dostępny jest po zakończonym pomiarze w rejestrze ADDAT (A/D converter DATa register):

ADDAT rejestr wyniku przetwarzania (adres 0D9h), rw-00:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
MSB							LSB

Ponieważ wynik pomiaru uaktualniany jest po każdym pomiarze, dlatego w trakcie wykonywania serii pomiarów wynik poprzedniego przetwarzania zastępowany jest następnym wynikiem. Jeśli po zgłoszeniu gotowości przez przetwornik, przez przerwanie lub przez testowanie znacznika BSY, nie zostanie pobrany wynik pomiaru, to wynik ten jest tracony.

Jeśli przetwornik analogowo-cyfrowy nie jest wykorzystywany to rejestr ADDAT może służyć jako rejestr ogólnego przeznaczenia.

13.4 Przykład pomiaru napięcia

Należy wykonać 5 pomiarów napięcia U_X dołączonego do kanału AN2 zmieniającego się w zakresie 0..+3V testując znacznik stanu przetwornika BSY. Wyniki pomiarów przesłać do wewnętrznej pamięci RAM od adresu 3Bh. Przykładowy program jest następujący:

```
A_C_Pomiar:
        MOV   R0,#3Bh           ;R0 ← 3Bh,  rejestr R0 adresuje
                                ;           wewnętrzną pamięć RAM
        MOV   R7,#5             ;R7 ← 5,    rejestr R7 jest licznikiem
                                ;           pomiarów
 ;rejestr ADCON:
 ;                 0 | 0 | 0 | BSY | ADM | MX2 | MX1 | MX0
        MOV   ADCON,#0000 1010b ;ADM ← 1, praca ciągła
                                ;MX2, MX1, MX0 ← 010b, kanał AN2
 ;rejestr DAPR:
 ;                 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0
        MOV   DAPR,#1010 0000b
;DAPR7..4 ← 1010b
;górna wartość napięcia podzakresu pomiarowego VIntAREF = 3.125 V,
;DAPR3..0 ← 0000b
;dolna wartość napięcia podzakresu pomiarowego VIntAGND = 0 V,
;wpis dowolnej wartości do DAPR rozpoczyna przetwarzanie,
;znacznik stanu przetwornika BSY = 1

Czekaj:
        JB    BSY,Czekaj        ;jeśli BSY = 1 to skok do etykiety
                                ;           (adresu) Czekaj
        MOV   @R0,ADDAT         ;(R0) ← (ADDAT), przesłanie wyniku
                                ;pomiaru do wewnętrznej pamięci RAM
        INC   R0                ;R0 ← R0 + 1
        DJNZ  R7,Czekaj         ;R7 ← R7 - 1,  jeśli R7 ≠0 to skok do
                                ;etykiety (adresu) Czekaj,
                                ;wykonanie następnego pomiaru
        ANL   ADCON,#1111 0111b ;ADM ← 0, koniec serii pomiarów
```

Pomiary wykonano w podzakresie 0..+3,125 V z rozdzielczością 8-bitów. Oznacza to, że aby podać wartość zmierzonego napięcia w kodzie dziesiętnym należy otrzymany wynik pomiaru pomnożyć przez wartość najmniejszego poziomu kwantowania wynoszącego:

$$1\,\text{LSB} = \frac{3{,}125\ \text{V}}{256} = 12{,}21\ \text{mV}.$$

? Pytania i problemy

1. Co oznaczają pojęcia: kwantowanie, kodowanie i próbkowanie sygnału analogowego ?

2. Jaka jest wartość najmniej znaczącego bitu (1 LSB) w przetworniku, którego zakres przetwarzania wynosi U_{FS} =10V (Full Scale range), a rozdzielczość przetwornika jest równa:

   a) N=8 bitów, b) N=10 bitów, c) N=12 bitów ?

4. Jakie jest przeznaczenie poszczególnych znaczników rejestru kontrolnego przetwornika ADCON ?

5. Przetwornik korzysta z zewnętrznych napięć wzorcowych. Jakie są to wartości ? Jaki jest ich wpływ na wynik przetwarzania ?

6. Jakie jest przeznaczenie rejestru DAPR, ADCON ?

7. Na czym polega 2-etapowy pomiar napięcia przy przetwarzaniu 10-bitowym ?

8. Dlaczego przetwornik sygnalizuje stan gotowości kilka cykli maszynowych wcześniej niż rzeczywiście zakończy przetwarzanie analogowo-cyfrowe ?

9. Jeśli obsługa przetwornika realizowana jest przez przerwania, to które znaczniki związane z przerwaniami muszą być zmienione i dlaczego ?

14. Licznik T2 w 8052 i SAB 80515/535

Problemy z licznikami T0 i T1, głównie z powodu braku 16-bitowych rejestrów pamiętających wartość początkową licznika w trybie autoładowania, oraz brak możliwości sprzętowego zapamiętywania wartości chwilowej licznika, spowodowały wprowadzenie w mikrokontrolerze 8052 i następnych trzeciego licznika oznaczonego symbolem T2. Przeważnie każdy z nowoopracowywanych mikrokontrolerów rodziny '51 ma zmienione lub rozszerzone funkcje licznika T2.

Program, który wykorzystuje licznik T2, nie może być bezpośrednio przeniesiony na inne typy mikrokontrolerów rodziny '51, a w szczególności na mikrokontrolery innych producentów.

Są oczywiście pewne podobieństwa między mikrokontrolerami tej samej firmy ale różnice między mikrokontrolerami różnych firm są bardzo duże. Przykładem, który to potwierdza jest porównanie dwóch liczników T2. Jednego z mikrokontrolera 8052 firmy Intel i drugiego z mikrokontrolera 80515/535 firmy Siemens lub AMD:

Tabela 14-1 Porównanie cech licznika T2 w 8052 i 80515/535.

8052	80515/535
• zliczanie impulsów wewnętrznych o częstotliwości: → $f_{GEN}/12$,	→ $f_{GEN}/12$ lub $f_{GEN}/24$,
• sprzętowe bramkowanie zliczanych, wewnętrznych impulsów:	→ sygnałem na wejściu P1.7/T2,
• zliczanie impulsów zewnętrznych doprowadzonych do: → wejścia P1.0/T2,	→ wejścia P1.7/T2,
• wartość chwilowa licznika może być zapamiętana w: → jednym 16-bitowym rejestrze RCAP,	→ czterech 16-bitowych rejestrach: CRC, CC1, CC2 lub CC3,
• wartość chwilowa licznika zapamiętywana jest sprzętowo jeśli: → wystąpi zbocze opadające sygnału na wejściu P1.1/ /T2EX,	→ wystąpi zbocze opadające lub narastające sygnału na wejściu P1.0/CC0/ $\overline{\text{INT3}}$, → wystąpi zbocze narastające sygnału na wejściu: P1.1/CC1/INT4, P1.2/CC2/ /INT5 lub P1.3/CC3/INT6,

8052	80515/535
→ zbocze opadające T2EX ustawia znacznik EXF2 i generuje przerwanie o adresie 02Bh,	→ zbocze opadające lub narastające P1.0/CC0/ $\overline{\text{INT3}}$ ustawia znacznik IEX3 i generuje przerwanie o adresie 53h, → zbocza opadające P1.1/CC1/ /INT4 .. P1.3/CC3/INT6 ustawiają znaczniki IEX4 .. IEX6 i generują przerwania o adresach: 5Bh .. 6Bh
• wartość chwilowa licznika zapamiętywana jest programowo jeśli: → wykonana zostanie instrukcja wpisu dowolnej wartości do mniej znaczącego rejestru: CRCL, CCL1, CCL2 lub CCL3	
• przepełnienie licznika ustawia znacznik przepełnienia TF2 i generuje przerwanie o adresie 02Bh,	
• wartość początkowa licznika przepisywana jest: → automatycznie po przepełnieniu licznika, → z jednego 16-bitowego rejestru RCAP,	→ z jednego 16-bitowego rejestru CRC, → zboczem opadającym P1.5/ /T2EX → zbocze opadające P1.5/T2EX ustawia znacznik EXF2 i generuje przerwanie o adresie 02Bh,
• wartość chwilowa licznika porównywana jest sprzętowo z: → czterema rejestrami: CRCL, CCL1, CCL2 lub CCL3 → stan równości licznika i wybranego rejestru (CRC, CC1, CC2 lub CC3) ustawia skojarzony z rejestrem znacznik (IEX3..IEX6) i generuje przerwanie ($\overline{\text{INT3}}$ ..INT6), → porównywanie wykonywane jest w dwóch trybach, → generowanie 4 sygnałów PWM (Pulse Width Modulation - modulacja szerokości impulsów) z rozdzielczością 16 bitów.	

Jak widać z podanej tabeli różnice między obu tak samo nazwanymi licznikami T2 są ogromne. Właściwie to co jest możliwe w mikrokontrolerze SAB 80515/535 nie jest możliwe w mikrokontrolerze 8052. Podobna sytuacja występuje w innych mikrokontrolerach rodziny '51. Tryby pracy licznika T2 określane są często mianem CCR:

- CCR - Compare, porównanie; tryb dostępny jedynie w SAB 80515/535,
- CCR - Capture, zapamiętanie wartości chwilowej,
- CCR - Reload, autoładowanie wartości początkowej licznika.

W obu mikrokontrolerach 8052 i SAB 80515/535 16-bitowy licznik T2 tworzą dwa 8-bitowe rejestry specjalne oznaczone symbolami:

- TL2 jest rejestrem mniej znaczącym o adresie 0CCh, rejestrem do odczytu i zapisu (po sprzętowym zerowaniu procesora TL2=0),
- TH2 jest rejestrem bardziej znaczącym o adresie 0CDh, rejestrem do odczytu i zapisu (po sprzętowym zerowaniu procesora TH2=0),

Sposób zliczania zewnętrznych impulsów taktujących jest taki sam jak w liczniku T0 lub T1.

Sygnał zewnętrzny jest testowany przez mikrokontroler w każdym cyklu maszynowym. Jeżeli w jednym cyklu wykryty został poziom wysoki sygnału, a w następnym poziom niski to zawartość rejestrów TH2, TL2 jest zwiększana o jeden. Inkrementacja licznika T2 dokonywana jest w następnym cyklu maszynowym w stosunku do cyklu, w którym wykryta została zmiana poziomu sygnału.

Ze sposobu testowania przez mikrokontroler poziomu zewnętrznego sygnału wynikają dwa ograniczenia. Po pierwsze czas trwania wysokiego poziomu sygnału musi być dłuższy niż 1 cykl maszynowy mikrokontrolera. Jest to warunek określający współczynnik wypełnienia impulsów. Po drugie minimalny okres zewnętrznego sygnału musi być dłuższy niż 2 cykle maszynowe.

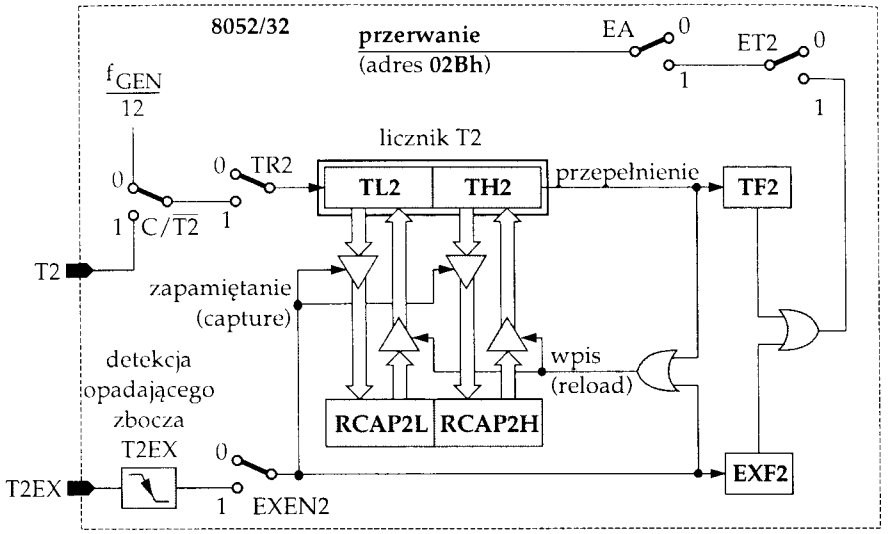
Jeśli mikrokontroler jest sterowany rezonatorem kwarcowym o częstotliwości 12 MHz to maksymalna częstotliwość zewnętrznego sygnału doprowadzonego do wejścia T2 nie może przekraczać 500 kHz.

Z licznikiem T2 związanych jest:

- 5 dodatkowych rejestrów w mikrokontrolerze **8052**: T2CON, RCAP2L, RCAP2H, IE oraz IP,
- 15 rejestrów specjalnych w mikrokontrolerze **SAB 80515/535**: T2CON, CCEN, CRCL, CRCH, CCL1, CCH1, CCL2, CCH2, CCL3, CCH3, IRCON, IEN0, IEN1, IP0 oraz IP1.

14.1 Licznik T2 w 8052

Schemat blokowy licznika T2 w mikrokontrolerze 8052 przedstawiono na rysunku 14-1.



Rys. 14-1 Schemat blokowy licznika T2 w mikrokontrolerze 8052.

Z licznikiem T2 współpracują następujące rejestry:

- 8-bitowy rejestr programujący T2CON (Timer 2 CONtrol register),
- 16-bitowy rejestrRCAP wartości początkowej lub wartości chwilowej licznika T2 (Reload CAPture register) złożony z dwóch 8-bitowych rejestrów: RCAP2L (rejestr mniej znaczący) i RCAP2H (rejestr bardziej znaczący),
- 8-bitowy rejestr przerwań IE,
- 8-bitowy rejestr priorytetów przerwań IP.

Wszystkie rejestry współpracujące z licznikiem T2 znajdują się w obszarze rejestrów specjalnych SFR:

- rejestr sterujący: **T2CON** (adres 0C8h), rw-00:

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$

TF2 - znacznik przepełnienia licznika T2,
EXF2 - znacznik ustawiany opadającym zboczem zewnętrznego sygnału T2EX jeśli EXEN2=1,
RCLK - znacznik przełączania generatora taktującego odbiornik łącza szeregowego pracującego w trybie 1 lub 3:

→ RCLK = 0, taktowanie odbiornika łącza szeregowego z licznika T1,
→ RCLK = 1, taktowanie odbiornika łącza szeregowego z licznika T2,

TCLK - znacznik przełączania generatora taktującego nadajnik łącza szeregowego pracującego w trybie 1 lub 3:
→ TCLK = 0, taktowanie nadajnika łącza szeregowego z licznika T1,
→ TCLK = 1, taktowanie nadajnika łącza szeregowego z licznika T2,

EXEN2 - znacznik blokujący zewnętrzne wejście T2EX,
TR2 - znacznik działania licznika,
C/$\overline{T2}$ - znacznik wyboru wejścia taktującego licznik,
CP/$\overline{RL2}$ - znacznik przełączania trybu pracy licznika:
→ CP/$\overline{RL2}$ = 0 dla trybu autoładowania (autoreload),
→ CP/$\overline{RL2}$ = 1 dla trybu zapamiętywania wartości chwilowej (capture),

- rejestr pomocniczy: **RCAP2L** (adres 0CAh), rw-00,
- rejestr pomocniczy: **RCAP2H** (adres 0CBh), rw-00,
- rejestr przerwań: **IE** (adres 0A8h), rw-00:

EA	-	ET2	ES	ET1	EX1	ET0	EX0

EA - znacznik odblokowania wszystkich przerwań,
ET2 - znacznik odblokowania przerwania od licznika T2.

- rejestr priorytetów przerwań: **IP** (adres 0B8h), rw-00:

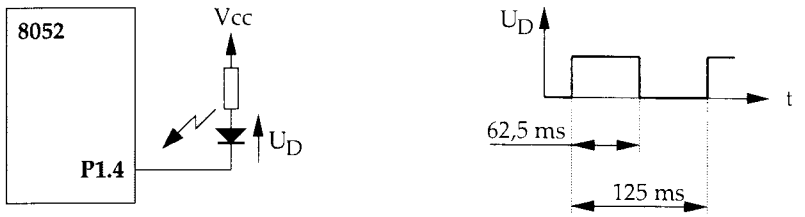
-	-	PT2	PS	PT1	PX1	PT0	PX0

PT2 - znacznik priorytetu przerwania od licznika T2.

Bardziej szczegółowe informacje dotyczące licznika T2 podane są w [1]. Licznik T2 w mikrokontrolerze 8052 w odróżnieniu od liczników T0 i T1 umożliwia sprzętowe generowanie stosunkowo długich odcinków czasu. W tym celu należy wykorzystać pomocniczy rejestr RCAP, w którym przechowywana jest wartość początkowa licznika po jego przepełnieniu. Jeśli

- do wyjścia linii portu P1.4 dołączyć diodę elektroluminescencyjną (LED) tak jak na rysunku 14-2,
- włączanie i wyłączanie diody ma następować z częstotliwością 8 Hz, czyli dioda świeci się lub jest zgaszona przez czas równy 62,5 ms = = 62.500 μs;
- częstotliwość rezonatora kwarcowego mikrokontrolera wynosi 12 MHz,

to program działania może być następujący:



Rys. 14-2 Generator astabilny o częstotliwości 8 Hz.

```
_RCAP    EQU    0FFFFh + 1 – 62500    ;przepełnienie licznika T2 wystąpi
                                      ;po stanie 0FFFFh+1
_T2CON   EQU    00000100b             ;programowanie licznika T2
```

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$

```
;   TF2 = 0, znacznik przepełnienia licznika T2,
;   EXEN2 = 0, blokowanie zewnętrznego wejścia T2EX,
;   TR2 = 1, znacznik działania licznika,
;   C/T2 = 0, taktowanie sygnałem wewnętrznym,
;   CP/RL2 = 0, tryb autoładowania po przepełnieniu licznika.
_IE      EQU    10100000b             ;programowanie przerwań
```

EAL	WDT	ET2	ES	ET1	EX1	ET0	EX0

```
;   EA =1, odblokowane wszystkie przerwania,
;   ET2 = 1, odblokowane przerwanie od przepełnienia licznika T2.

         CSEG   AT     0
Dioda_P1_4:
   MOV   TL2,#Low _RCAP          ;wartości początkowe licznika T2
   MOV   TH2,#High _RCAP
   MOV   RCAP2L,#Low _RCAP       ;i rejestru RCAP
   MOV   RCAP2H,#High _RCAP      ;
   MOV   IE,#_IE                 ;programowanie przerwań
   MOV   T2CON,# _T2CON          ;start licznika
   ..........                    ;dalsza część programu

   ORG   2Bh
INT_T2:                          ;procedura obsługi przerwania po przepełnieniu licznika
   CLR   TF2                     ;kasowanie znacznika przerwania
   CPL   P1.4                    ;negacja linii P1.4
   RETI                          ;koniec podprogramu
```


14.2 Licznik T2 w SAB 80515/535

W mikrokontrolerach SAB 80515/535 ze względu na rozbudowane funkcje wewnętrzne licznika T2, tryby zliczania, porównania, zapamiętania wartości chwilowej licznika, linie portu P1 sterujące działaniem licznika mają podwójne lub potrójne przeznaczenie (tabela 14-2). Jeśli licznik T2 nie jest wykorzystywany w trybie modulacji szerokości impulsów, wówczas linie P1.0 .. P1.3 można wykorzystać jako:

- wejścia przerwań: $\overline{INT3}$ .. INT6,
- standardowe wejścia/wyjścia portu P1 jeśli nie są wykorzystywane przerwania $\overline{INT3}$ .. INT6.

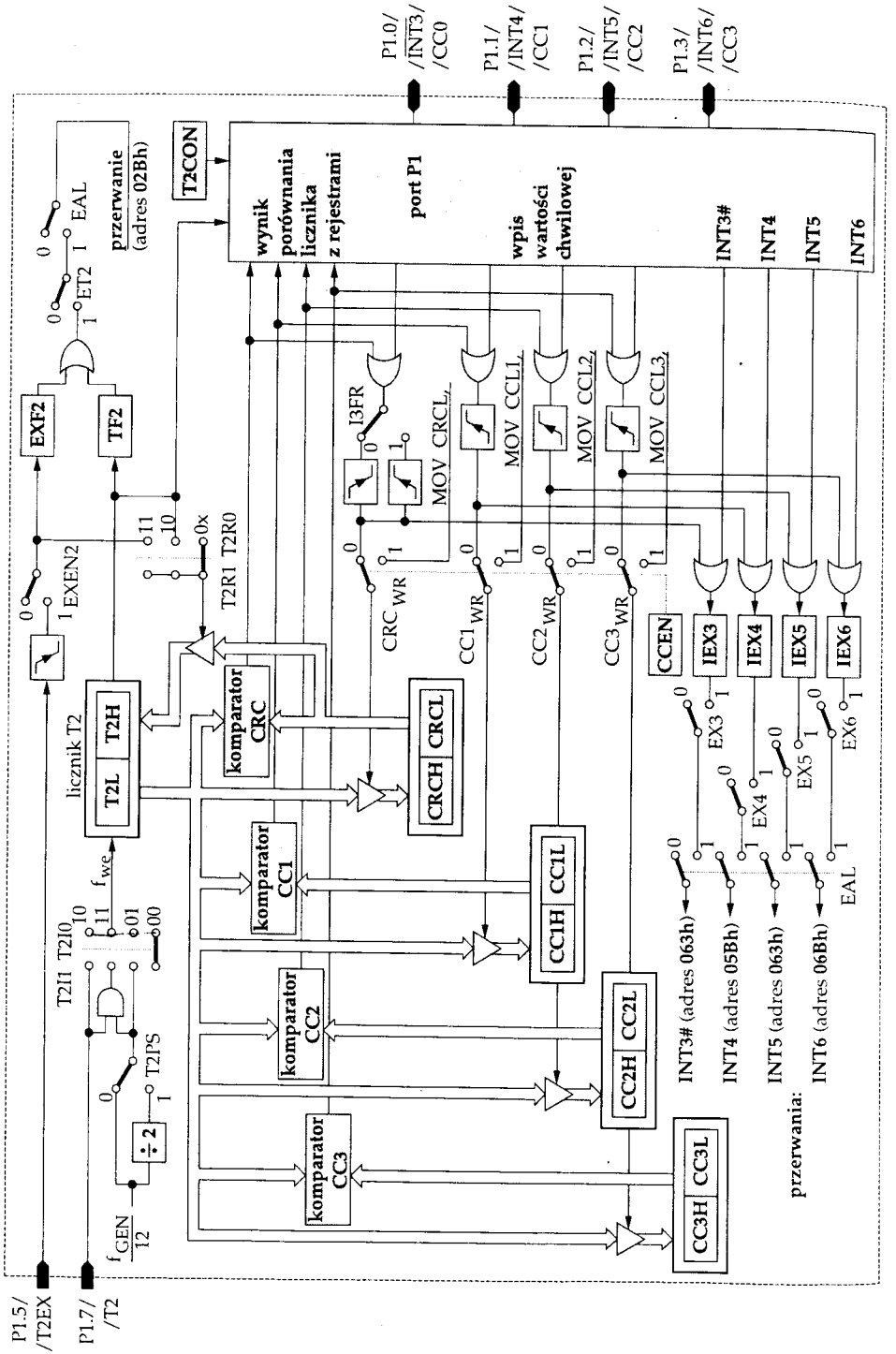
Ta sama zasada dotyczy linii:

- P1.7/T2, która może równocześnie pełnić rolę wejścia impulsów zewnętrznych zliczanych w liczniku T2 lub linii portu P1,
- P1.5/T2EX wykorzystywanej do sprzętowego ustalania momentu wpisu wartości początkowej do licznika T2.

Tabela 14-2 Funkcje alternatywne portu P1.0 .. P1.3.

Symbol	Typ	Funkcja
P1.0/ $\overline{INT3}$ / CC0	we/wy	linia portu P1.0/wejście przerwania $\overline{INT3}$ / /sprzętowy wpis wartości chwilowej licznika T2 do rejestru CRC/ /wyjście sterowane komparatorem CRC
P1.1/ INT4/ CC1	we/wy	linia portu P1.1/wejście przerwania INT4/ /sprzętowy wpis wartości chwilowej licznika T2 do rejestru CC1/ /wyjście sterowane komparatorem CC1
P1.2/ INT5/ CC2	we/wy	linia portu P1.2/wejście przerwania INT5/ /sprzętowy wpis wartości chwilowej licznika T2 do rejestru CC2/ /wyjście sterowane komparatorem CC2
P1.3/ INT6/ CC3	we/wy	linia portu P1.3/wejście przerwania INT6/ /sprzętowy wpis wartości chwilowej licznika T2 do rejestru CC3/ /wyjście sterowane komparatorem CC3

W mikrokontrolerze SAB 80515/535 z licznikiem T2 związanych jest 15 rejestrów: przechowujących wartości chwilowe (CRC, CC1 .. CC3) i wartość początkową w trybie autoładowania (CRC), rejestru odblokowującego tryb porównania/wpisu wartości chwilowej (CCEN), programującego tryb pracy licznika (T2CON), rejestrów uaktywniających strukturę i priorytety przerwań (IRCON, IEN0, IEN1, IP0 oraz IP1).



Rys. 14-3 Schemat blokowy licznika T2 w mikrokontrolerze 80515/535.

Schemat blokowy licznika T2 pokazano na rysunku 14-3. Symbole tych rejestrów, adresy i przeznaczenie przedstawiono poniżej.

- 16-bitowy rejestr **CRC** porównania/wartości początkowej/wartości chwilowej licznika T2:

CRCH (adres 0CBh), rw-00	CRCL (adres 0CAh), rw-00

- 16-bitowy rejestr **CC1** porównania /wartości chwilowej licznika T2:

CCH1 (adres 0C3h), rw-00	CCL1 (adres 0C2h), rw-00

- 16-bitowy rejestr **CC2** porównania /wartości chwilowej licznika T2:

CCH2 (adres 0C5h), rw-00	CCL2 (adres 0C4h), rw-00

- 16-bitowy rejestr **CC3** porównania /wartości chwilowej licznika T2:

CCH3 (adres 0C7h), rw-00	CCL3 (adres 0C6h), rw-00

- rejestr programowania trybów licznika T2 i sygnału taktującego: **T2CON** (adres 0C8h), rw-00:

T2PS	I3FR	I2FR	T2R1	T2R0	T2CM	T2I1	T2I0

T2PS - włączenie (T2PS=1)/wyłączenie (T2PS=0) dodatkowego dzielnika wstępnego przez 2,

I3FR - wybór zbocza narastającego (I3FR=1)/opadającego (I3FR=0) zewnętrznego przerwania $\overline{\text{INT3}}$, wyjścia komparatora CRC, wpisu wartości chwilowej licznika T2 do rejestru CRC,

T2R1,T2R0 - wybór trybu autoładowania licznika:
→ T2R1,T2R0 =0xb, zablokowane funkcje autoładowania,
→ T2R1,T2R0 =10b, tryb 0, autoładowanie po przepełnieniu licznika T2,
→ T2R1,T2R0 =11b, tryb 1, autoładowanie opadającym zboczem sygnału T2EX,

T2CM - wybór trybu 0 (T2CM=0) lub 1 (T2CM=1) porównania,

T2I1,T2I0 - wybór sygnału taktującego licznik:
→ T2I1,T2I0=00b zatrzymanie licznika,
→ T2I1,T2I0=01b taktowanie sygnałem wewnętrznym: $f_{GEN}/12$ (T2PS=0) lub $f_{GEN}/24$ (T2PS=1),
→ T2I1,T2I0=10b taktowanie sygnałem zewnętrznym,
→ T2I1,T2I0=11b bramkowanie wewnętrznego sygnału taktującego sygnałem zewnętrznym P1.7/T2.

- rejestr odblokowywania trybów porównania i wpisu wartości chwilowej licznika T2 do rejestrów CCx: **CCEN** (adres 0C1h), rw-00:

COCAH3	COCAL3	COCAH2	COCAL2

COCAH1	COCAL1	COCAH0	COCAL0

COCAH3,COCAL3 - wybór trybu dla rejestru CC3:
- 00 - zablokowany tryb porównania/wpisu wartości chwilowej,
- 01 - odblokowanie zapamiętania stanu licznika T2 zboczem narastającym sygnału na wejściu P1.3/INT6/CC3,
- 10 - odblokowany tryb porównania i wpisu wartości chwilowej,
- 11 - zapamiętanie wartości chwilowej licznika T2 po wpisie do rejestru CCL3,

COCAH2,COCAL2 - wybór trybu dla rejestru CC2:
- 00 - zablokowany tryb porównania/wpisu wartości chwilowej,
- 01 - odblokowanie zapamiętania stanu licznika T2 zboczem narastającym sygnału na wejściu P1.2/INT5/CC2,
- 10 - odblokowany tryb porównania i wpisu wartości chwilowej,
- 11 - zapamiętanie wartości chwilowej licznika T2 po wpisie do rejestru CCL2,

COCAH1,COCAL1 - wybór trybu dla rejestru CC1:
- 00 - zablokowany tryb porównania/wpisu wartości chwilowej,
- 01 - odblokowanie zapamiętania stanu licznika T2 zboczem narastającym sygnału na wejściu P1.1/INT4/CC1,
- 10 - odblokowany tryb porównania i wpisu wartości chwilowej,
- 11 - zapamiętanie wartości chwilowej licznika T2 po wpisie do rejestru CCL1,

COCAH0,COCAL0 - wybór trybu dla rejestru CRC:
- 00 - zablokowany tryb porównania/wpisu wartości chwilowej,
- 01 - odblokowanie zapamiętania stanu licznika T2 zboczem narastającym/opadającym sygnału na wejściu P1.0/ $\overline{\text{INT3}}$ /CC0,
- 10 - odblokowany tryb porównania i wpisu wartości chwilowej,
- 11 - zapamiętanie wartości chwilowej licznika T2 po wpisie do rejestru CRCL.

- rejestr systemu przerwań: **IRCON** (adres 0C0h), rw-00:

EXF2	TF2	IEX6	IEX5	IEX4	IEX3	IEX2	IADC

EXF2 - znacznik zewnętrznego autoładowania licznika T2,
TF2 - znacznik przepełnienia licznika T2,
IEX6 - znacznik przerwania INT6/CC3,
IEX5 - znacznik przerwania INT5/CC2,
IEX4 - znacznik przerwania INT4/CC1,
IEX3 - znacznik przerwania $\overline{\text{INT3}}$ /CRC.

- rejestr przerwań: **IEN0** (adres 0A8h), rw-00:

EAL	WDT	ET2	ES	ET1	EX1	ET0	EX0

EAL - znacznik odblokowania wszystkich przerwań,
ET2 - znacznik odblokowania przerwania od licznika T2.

- rejestr przerwań: **IEN1** (adres 0B8h), rw-00:

EXEN2	SWDT	EX6	EX5	EX4	EX3	EX2	EADC

EXEN2 - znacznik odblokowania przerwania zewnętrznego autoładowania licznika T2 zboczem opadającym T2EX,
EX6 - znacznik odblokowania zewnętrznego przerwania INT6/CC3,
EX5 - znacznik odblokowania zewnętrznego przerwania INT5/CC2,
EX4 - znacznik odblokowania zewnętrznego przerwania INT4/CC1,
EX3 - znacznik odblokowania zewnętrznego przerwania $\overline{\text{INT3}}$ /CRC.

- rejestr priorytetów przerwań: **IP0** (adres 0A9h), rw-00:

-	WDTS	IP0.5	IP0.4	IP0.3	IP0.2	IP0.1	IP0.0

- rejestr priorytetów przerwań: **IP1** (adres 0B9h), rw-00:

-	-	IP1.5	IP1.4	IP1.3	IP1.2	IP1.1	IP1.0

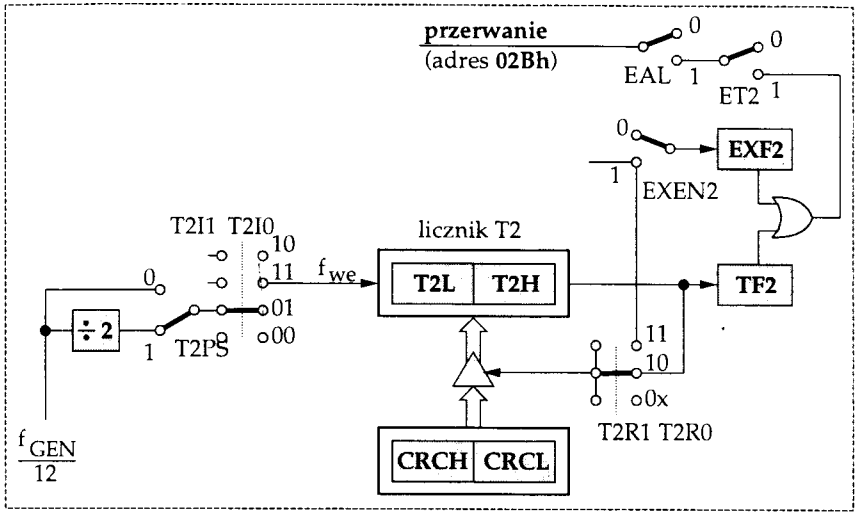
IP1.5, IP0.5 - priorytet dla przerwań: (TF2 + EXF2)/IEX6,
IP1.4, IP0.4 - priorytet dla przerwań: (RI + TI)/IEX5,
IP1.3, IP0.3 - priorytet dla przerwań: TF1/IEX4,
IP1.2, IP0.2 - priorytet dla przerwań: IE1/IEX3.

Licznik T2 w mikrokontrolerach 80515/535 jest bardzo złożony, w szczególności programy sterujące jego działaniem. Pełne informacje dotyczące działania licznika T2 znaleźć można w [2]. Poniżej przedstawiono trzy przykłady programów:
- sprzętowe generowanie długich odcinków czasu do 131 ms z rozdzielczością 16 bitów,
- 16-bitową modulację czasu trwania i szerokości impulsów zwaną PWM (Pulse Width Modulation),
- dynamiczne zapamiętywanie wartości chwilowej licznika.

14.2.1 Jak uzyskać długie odcinki czasu ?

Pod pojęciem długich odcinków czasu kryją się przedziały czasowe w zakresie do 131 ms (dokładnie 131,072 ms). Są to dwukrotnie dłuższe czasy niż możliwe do osiągnięcia w mikrokontrolerze 8052. Dzieje się tak dlatego, że w

mikrokontrolerze 80515/535 przed licznikiem T2 może być włączony dodatkowy dzielnik przez 2, tak jak przedstawiono to na rysunku 14-4.



Rys. 14-4 Struktura licznika T2 przy generowaniu długich odcinków czasu.

Tryb autoładowania, wpisywania wartości początkowej licznika T2 po jego przepełnieniu, determinowany jest:

- **16-bitowym licznikiem T2**, co oznacza, że możliwe jest zliczenie maksymalnie 65 536 wejściowych impulsów taktujących,
- **wewnętrzny sygnał taktujący** ma częstotliwość f_{GEN}**/12** lub f_{GEN}**/24** jeśli włączony jest dodatkowy dzielnik częstotliwości,
- po przepełnieniu licznik T2 generuje sygnał **przerwania** o adresie **2Bh**; w procedurze obsługi przerwania **znacznik TF2** musi być **kasowany programowo**, ponieważ źródłem przerwania może być również zewnętrzny sygnał ustawiający znacznik EXF2,
- po przepełnieniu wartość początkowa licznika T2 jest **automatycznie przepisywana** z pomocniczego rejestru **CRC**.

W poniższym przykładzie przyjęto, że w licznik T2 ma generować cyklicznie przerwanie co 100 ms w mikrokontrolerze sterowanym rezonatorem kwarcowym o częstotliwości f_{GEN} =12 MHz. Program realizujący podane założenia składa się z trzech części. W pierwszej części podane są deklaracje wartości używanych rejestrów. Należy pamiętać, że przepełnienie licznika T2 występuje gdy liczba zliczonych przez licznik impulsów wynosi 0FFFFh + 1. Ze względu na dołączenie dodatkowego dzielnika przez 2 czas 100 ms upływa po zliczeniu 50.000 wewnętrznych impulsów (100ms = 2 * 50.000µs). W drugiej części rozpoczynającej się od etykiety Czas_100_ms inicjalizowane są rejestry

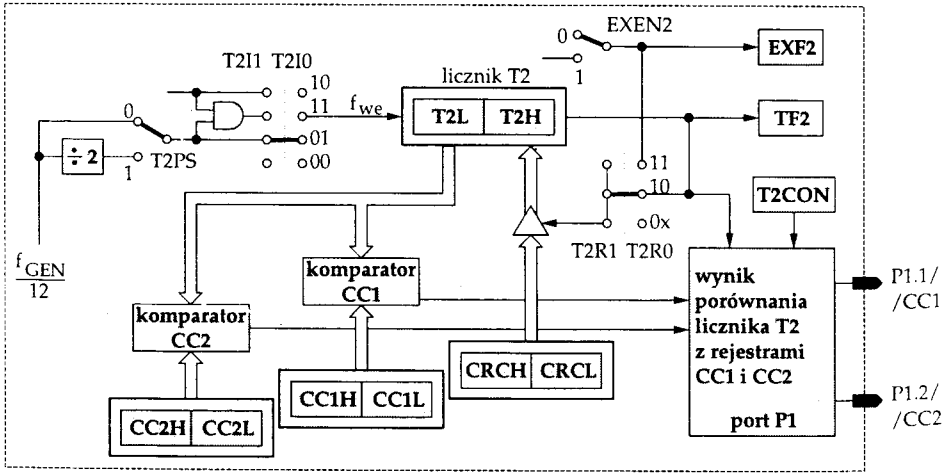
licznika T2 i rejestry współpracujące z licznikiem T2. Trzecia część rozpoczynająca się od etykiety INT_T2 jest początkiem procedury obsługi przerwania po przepełnieniu licznika T2.

```
Stan_T2     EQU  0FFFFh +1 – 50000  ;= 15536 = 3CB0h,   wartość
                                    ;początkowa licznika T2 po starcie
Stan_CRC    EQU  Stan_T2            ;stan licznika T2 po przepełnieniu
Stan_IEN0   EQU  10100000b          ;programowanie przerwań
;        | EAL | WDT | ET2 | ES | ET1 | EX1 | ET0 | EX0 |
;        EAL =1, odblokowane wszystkie przerwania,
;        ET2 = 1, odblokowane przerwanie od przepełnienia licznika T2.
Stan_T2CON  EQU  10010001b          ;programowanie trybu licznika T2
;        | T2PS | I3FR | I2FR | T2R1 | T2R0 | T2CM | T2I1 | T2I0 |
;        T2PS =1, włączony dodatkowy dzielnik przez 2,
;        I3FR = 0, znacznik nieistotny,
;        T2R1,T2R0 = 10b, wpis wartości początkowej po przepełnieniu,
;        T2CM = 0, znacznik nieistotny,
;        T2I1,T2I0 = 01b, taktowanie sygnałem wewnętrznym.

Czas_100_ms:
    ;inicjalizacja wartości początkowych rejestrów: T2, CRC i T2CON
    MOV  TH2,#High Stan_T2      ;TH2 ← 03Ch,    wartość począt-
    MOV  TL2,#Low Stan_T2       ;TL2 ← 0B0h,    kowa po starcie
    MOV  CRCH,#High Stan_CRC    ;CRCH ← 03CH,   wartość po prze-
    MOV  CRCL,#Low Stan_CRC     ;CRCL ← 0B0H,   pełnieniu licznika
    MOV  IEN0,#Stan_IEN0        ;programowanie kontrolera przerwań
    MOV  T2CON,#Stan_T2CON      ;programowanie i start licznika T2

         ORG  2Bh
INT_T2:
    ;początek procedury obsługi przerwania od licznika T2
    CLR  TF2                    ;kasowanie znacznika przerwania,
                                ;ponieważ przerwanie może być
                                ;zgłoszone także przez T2EX
    .....                       ;procedury użytkownika
    RETI                        ;koniec programu obsługi przerwania
```

14.2.2 Modulacja okresu i współczynnika wypełnienia impulsów (PWM)

PWM (Pulse Width Modulation) jest modulacją czasu trwania i szerokości impulsów. Sposób zmiany okresu generowanego sygnału z 16-bitową rozdzielczością został przedstawiony w poprzednim podrozdziale. Zmiana współczynnika wypełnienia impulsów jest drugą podstawową cechą licznika T2. Aby to uzyskać konstruktorzy mikrokontrolera wprowadzili cztery 16-bitowe komparatory porównujące stan licznika T2 z czterema 16-bitowymi rejestrami: CRC, CC1, CC2 i CC3. Zmiana współczynnika wypełnienia impulsów realizowana jest więc również z 16-bitową rozdzielczością. Ponieważ rejestr CRC jest używany do zapamiętywania wartości początkowej licznika T2 po przepełnieniu, dlatego praktycznie dostępne są tylko 3 komparatory i rejes-try (CC1, CC2 i CC3).

Przy porównywaniu wartości chwilowej możliwe są dwa tryby. Częściej wykorzystywany jest tryb wyłącznie sprzętowej modulacji szerokości impulsów - tryb 0. Drugi tryb (tryb 1) omówiony jest w [2]. Struktura wewnętrzna licznika T2 i towarzyszących mu rejestrów w trybie 0 porównania przedstawiona jest na rysunku 14-5.



Rys. 14-5 Struktura licznika T2 w trybie 0 porównania.

Minimalna i maksymalna szerokość generowanych impulsów wynosi 1/2 cyklu maszynowego. Obie wartości osiągane są gdy:

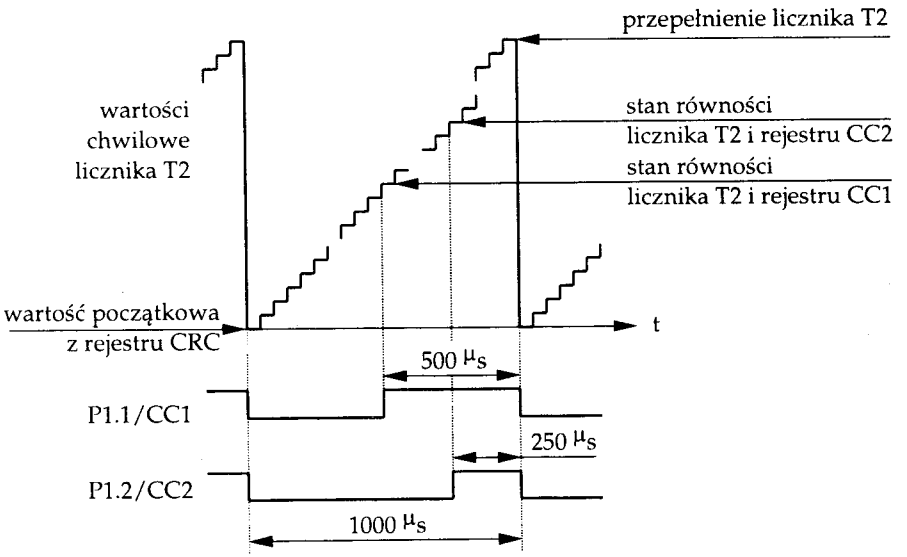
- stan rejestrów: CRC, CC1, CC2 lub CC3 jest równy 0000 (wartość minimalna),
- stan rejestrów: CRC, CC1, CC2 lub CC3 jest równy 0FFFFh (wartość maksymalna).

Wynika z tego, że modulacja szerokości generowanych impulsów możliwa jest w zakresie:

- 0,195% .. 99,805% przy modulacji 8-bitowej; wykorzystane są tylko rejestry mniej znaczące, np. CRCL, CCL1, CCL2 lub CCL3; stan rejestrów bardziej znaczących nie ulega zmianie i jest równy 0FFh (CRCH, CCH1, CCH2, CCH3),
- 0,00076% .. 99,9992% przy modulacji 16-bitowej; wykorzystane są w pełni 16-bitowe rejestry: CRC, CC1, CC2, CC3.

W trybie 0 porównania **stan równości** licznika T2 i wybranego rejestru (CRC, CC1 .. CC3) powoduje, że komparatory skojarzone z odpowiednimi rejestrami ustawiają linie portu: **P1.0 .. P1.3 w stan jedynki logicznej** oraz **ustawiają** znaczniki przerwań: **IEX3 .. IEX6**. Zmiana linii portu P1 dokonywana jest w tym samym cyklu maszynowym, w którym wystąpił stan równości zawartości licznika T2 i wybranego rejestru. Linie **portu P1 są kasowane** w momencie **przepełnienia licznika T2**.

W przypadku generowania dwóch sygnałów, tak jak przedstawiono to na rysunku 14-6, licznik T2 zlicza wewnętrzne impulsy wejściowe, a komparatory porównują chwilową wartość licznika z rejestrami CCx (CC1 .. CC3).



Rys. 14-6 Stan linii P1.1/CC1 i P1.2/CC2 w zależności od wartości chwilowej licznika T2 i wartości rejestrów: CRC, CC1 i CC2.

W trybie 0 porównania w momencie gdy chwilowa wartość licznika T2 jest równa zawartości któregoś z rejestrów (np. CC1) wówczas na wyjściu portu P1

skojarzonym z tym rejestrem (P1.1/CC1) pojawia się stan jedynki logicznej. Stan ten trwa dopóki nie nastąpi przepełnienie licznika. W tym momencie linia P1.1/CC1 przechodzi w stan zera logicznego. W przykładowym programie, który został przedstawiony poniżej drugi sygnał generowany jest na linii P1.2/CC2 (rysunek 14-6). W programie założono następujące warunki pracy mikrokontrolera:

- częstotliwość rezonatora kwarcowego mikrokontrolera wynosi 12 MHz,
- wewnętrzny sygnał taktujący ma częstotliwość $f_{GEN}/12$; nie jest włączony dodatkowy dzielnik częstotliwości,
- po przepełnieniu wartość początkowa licznika T2 jest przepisywana z pomocniczego rejestru CRC,
- sygnały generowane są na liniach portu P1.1/CC1 i P1.2/CC2,
- porównywanie chwilowej wartości licznika T2 i pomocniczych rejestrów CC1 oraz CC2 wykonywane jest w trybie 0,
- w programie nie wykorzystuje się przerwań; nie ma takiej potrzeby.

Program realizujący podane założenia złożony jest z dwóch części:

- pierwszą są deklaracje stałych,
- drugą program sterujący licznikiem T2 rozpoczynający się od etykiety Generator_1_2.

```
Okres       EQU  1000                ;okres obu sygnałów w µs
Stan_T2     EQU  0FFFFh +1 – Okres   ;= 0FC18h,  wartość początkowa
                                     ;           licznika T2
Stan_CRC    EQU  Stan_T2 ;wartość początkowa licznika po przepełnieniu
P1_1_H      EQU  500     ;czas trwania jedynki logicznej na wyjściu P1.1
Stan_CC1    EQU  0FFFFh + 1 – P1_1_H ;= 0FE0Ch,  wartość porównywa-
                                     ;           na w rejestrze CC1
P1_2_H      EQU  250     ;czas trwania jedynki logicznej na wyjściu P1.2
Stan_CC2    EQU  0FFFFh + 1 – P1_2_H ;= 0FF06h,  wartość porównywa-
                                     ;           na w rejestrze CC2
Stan_CCEN   EQU  00101000b       ;odblokowany tryb porównania
                                 ;dla rejestru CC1 oraz CC2
```

COCAH3	COCAL3	COCAH2	COCAL2

COCAH1	COCAL1	COCAH0	COCAL0

;COCAH2,COCAL2 - wybór trybu porównania wartości chwilowej licznika z rejestrem CC2,

;COCAH1,COCAL1 - wybór trybu porównania wartości chwilowej licznika z rejestrem CC1,

```
Stan_T2CON  EQU  00010001b       ;programowanie trybów licznika T2
```

T2PS	I3FR	I2FR	T2R1	T2R0	T2CM	T2I1	T2I0

```
;           T2PS =0, wyłączony dodatkowy dzielnik przez 2,
;           I3FR = 0, znacznik nieistotny,
```



```
;           T2R1,T2R0 = 10b, wpis wartości początkowej po przepełnieniu,
;           T2CM = 0, porównanie w trybie 0,
;           T2I1,T2I0 = 01b, taktowanie sygnałem wewnętrznym.

        CSEG  AT   0              ;deklaracja segmentu kodu programu
Generator_1_2:
        MOV   TH2,#High Stan_T2   ;TH2 ← 0FCh,   wpisanie wartości
                                  ;               początkowej do
        MOV   TL2,#Low Stan_T2    ;TL2 ← 18h,     rejestrów licznika
        MOV   CRCH,#High Stan_CRC ;CRCH ← 0FCh,   wartość początko-
                                  ;               wa w licznika T2
        MOV   CRCL,#Low Stan_CRC  ;CRCL ←18h,     po przepełnieniu
        MOV   CCH1,#High Stan_CC1 ;CCH1 ← 0FEh,   wartość porów-
        MOV   CCL1,#Low Stan_CC1  ;CCL1 ← 0Ch,    nania dla P1.1
        MOV   CCH2,#High Stan_CC1 ;CCH2 ← 0FFh,   wartość porów-
        MOV   CCL2,#Low Stan_CC1  ;CCL2 ← 06h,    nania dla P1.2
        MOV   CCEN,#Stan_CCEN     ;odblokowanie trybu porównania
        MOV   T2CON,#Stan_T2CON   ;programowanie i start licznika T2
Koniec_programowania:
        NOP                       ;miejsce na punkt zatrzymania programu
        END                       ;koniec kodu źródłowego programu
```

Raz zaprogramowany licznik T2 generuje sygnał aż do ponownego programowania, niezależnie od obliczeń wykonywanych przez jednostkę arytmetyczno-logiczną mikrokontrolera, zerowania mikrokontrolera (RESET#) lub wyłączenia zasilania. Efekt programowania licznika T2 można zaobserwować na oscyloskopie dołączonym do linii portu P1.1/CC1 i P1.2/CC2.

14.2.3 Zapamiętanie wartości chwilowej licznika T2

Każdy z czterech rejestrów CRC..CC3 może być użyty do zapamiętania wartości chwilowej licznika T2. Operacja ta realizowana jest sprzętowo. Gdyby wykonać tę operację programowo, to odczytane wartości mogłyby się różnić od wartości rzeczywistej o liczbę cykli maszynowych potrzebnych do wykonania tych instrukcji. Innym rozwiązaniem jest zatrzymanie na czas odczytu licznika T2 (np. T2I1=T2I0=0) i ponowne jego uruchomienie po wykonaniu odczytu. Gdyby licznik nie był zatrzymywany to jego zawartość zostałaby zwiększona o liczbę cykli maszynowych potrzebnych do wykonania programu:

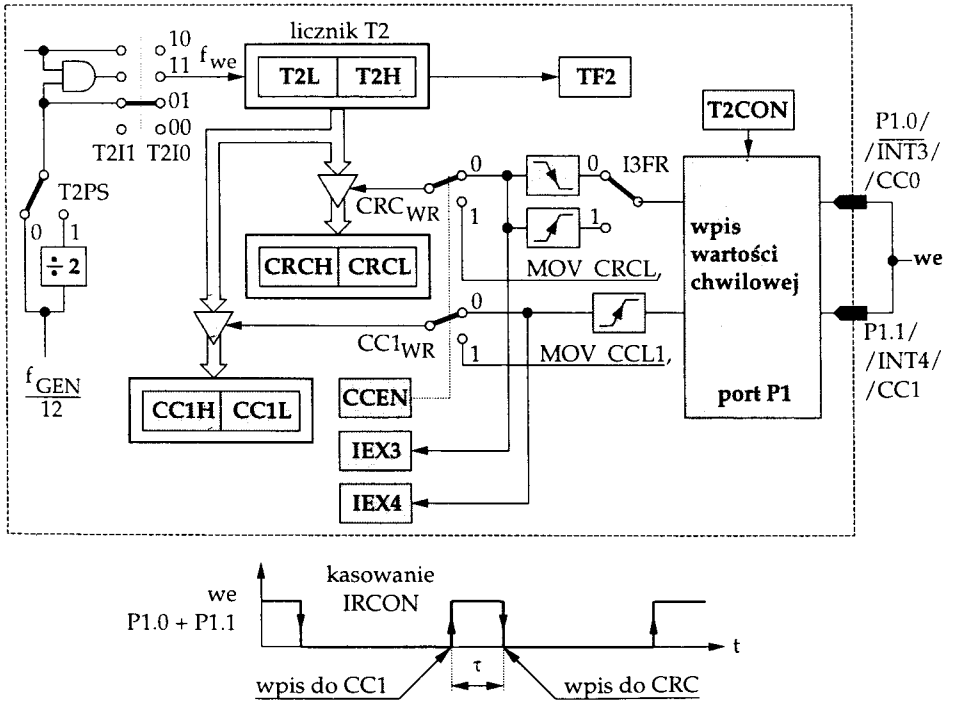
- zatrzymania licznika,
- odczytu zawartości licznika,
- ponownego uruchomienia licznika.

Operację dynamicznego odczytu wartości chwilowej licznika często określa się mianem odczytu 'w locie' (on-the-fly). Sytuacje tę przedstawiono na rysunku 14-7.

Zapamiętanie wartości chwilowej licznika T2 realizowane jest w:

- trybie 0, w którym **moment zapamiętania** wartości chwilowej wyznaczony jest **sygnałem zewnętrznym**; przykład działania licznika w tym trybie przedstawiony jest poniżej,
- trybie 1, w którym **wpis dowolnej wartości** do mniej znaczącego rejestru: **CRCL .. CCL3** wyznacza **moment zapamiętania** wartości chwilowej; tryb ten omówiony jest w [2].

Wybór właściwego **trybu** wykonywany jest przez zmianę zawartości rejestru **CCEN**. Aktywne zbocza sygnałów $\overline{\text{INT3}}$ .. INT6 powodują wpisanie wartości chwilowej licznika T2 do odpowiadającego sygnałowi rejestru CRC..CC3 oraz wpisania jedynki na pozycję znacznika przerwania IEX3 .. IEX6.



Rys. 14-7 Pomiar współczynnika wypełnienia sygnału impulsowego.

Jeśli system przerwań jest odblokowany i zgłaszane przerwanie jest przyjmowane, to po zawieszenie wykonywanego programu następuje skok do adresu odpowiadającego temu przerwaniu. Ze względu na sposób sterowania zewnętrznymi sygnałami zboczem aktywnym sygnału jest:

- zbocze narastające dla wejść przerwań INT4..INT6 skojarzonych z rejestrami CC1..CC3,
- zbocze narastające (I3FR=1) lub opadające (I3FR=0) dla wejścia $\overline{\text{INT3}}$ skojarzonego z rejestrem CRC; wybór rodzaju aktywnego zbocza dokonywany jest w rejestrze T2CON.6,
- ze względu na pojedyncze źródło przerwań, rozpoczęcie obsługi przerwania: $\overline{\text{INT3}}$ .. INT6 kasuje sprzętowo znacznik, który wywołał przerwanie (IEX3 .. IEX6).

Do bufora wyjściowego portu P1 musi być wpisana wartość jedynki logicznej. W przeciwnym przypadku nie można określić momentu pojawienia się aktywnego zbocza sygnału (linia w stanie zera logicznego). Sygnały na wejściach $\overline{\text{INT3}}$ .. INT6 testowane są przez mikrokontroler w każdym cyklu maszynowym. Wartość chwilowa licznika T2 jest dostępna w rejestrach CRC..CC3 w następnym cyklu maszynowym w stosunku do cyklu, w którym zostało wykryte aktywne zbocze sygnału.

W poniżej przedstawionym przykładzie programu licznik T2 użyto do pomiar czasu trwania impulsu. Pomiar wykonywany jest z dokładnością 1 µs. Z czterech, dostępnych do zapamiętywania wartości chwilowej licznika T2 rejestrów: CRC, CC1, CC2 i CC3 jedynie do rejestru CRC można przepisać stan licznika T2 zboczem opadającym sygnału dołączonym do wejścia P1.0/CC0. Do pozostałych 3 rejestrów CCx stan licznika T2 przepisywany jest zboczem narastającym.

Pomiar współczynnika wypełnienia τ jest następujący (rysunek 14-7):

- należy wybrać tryb 0 wpisywania wartości chwilowej licznika T2,
- wartość chwilowa licznika T2 będzie wpisywana zboczem:
  → narastającym do rejestru CC1,
  → opadającym do rejestru CRC,
- należy wyzerować rejestr przerwań IRCON, w którym znajduje się znacznik IEX3 sygnalizujący opadające zbocze sygnału wpisującego stan licznika T2 do rejestru CRC,
- uruchomić licznik T2 taktowany sygnałem wewnętrznym,
- po wykryciu zbocza opadającego zatrzymać licznik T2 i zablokować wpis kolejnych wartości chwilowych licznika T2 do rejestrów CRC i CC1,
- czas τ trwania jedynki logicznej mierzonego sygnału nie powoduje przepełnienia licznika T2.

Przykładowy program realizujący podane założenia jest następujący:

```
Stan_T2     EQU  0          ;wartość początkowa licznika T2
Stan_CCEN   EQU  00000101b  ;odblokowanie trybu 0 wpisu wartości
                            ;chwilowej licznika T2 do rejestrów: CRC, CC1
;           | COCAH3 | COCAL3 | COCAH2 | COCAL2 |
;           | COCAH1 | COCAL1 | COCAH0 | COCAL0 |
;COCAH1,COCAL1  - wybór trybu pamiętania wartości chwilowej w rejes-
                 trze CC1,
;COCAH0,COCAL0  - wybór trybu pamiętania wartości chwilowej w rejes-
                 trze CRC,
Stan_T2CON  EQU  00000001b  ;programowanie licznika T2:
;           | T2PS | I3FR | I2FR | T2R1 | T2R0 | T2CM | T2I1 | T2I0 |
;           T2PS =0, wyłączony dodatkowy dzielnik przez 2,
;           I3FR = 0, wpis do rejestru CRC zboczem opadającym,
;           T2R1,T2R0 = 00b, blokada autoładowania
;           T2CM = 0, porównanie w trybie 0,
;           T2I1,T2I0 = 01b, taktowanie sygnałem wewnętrznym.

      CSEG  AT   0              ;deklaracja segmentu kodu
Pomiar_τ:
      MOV   TH2,#High Stan_T2   ;TH2 ← 0,     wartość początkowa
      MOV   TL2,#Low Stan_T2    ;TL2 ← 0,     licznika T2
Czekaj_Start:                   ;oczekiwanie na zero logiczne
      JB    P1.0,Czekaj_Start   ;mierzonego sygnału
      MOV   IRCON,#0            ;zerowanie znacznika IEX3
Start_T2:
      MOV   CCEN,#Stan_CCEN     ;odblokowanie wpisu wartości
                                ;          chwilowej licznika T2
      MOV   T2CON,#Stan_T2CON   ;uruchomienie licznika T2
Czekaj_Pomiar:
      JNB   IEX3,$              ;oczekiwanie na zbocze opadające
                                ;          sygnału
      MOV   CCEN,#0             ;blokowanie wpisu kolejnej wartości
                                ;          chwilowej licznika T2
      ANL   T2CON,#0FCH         ;zatrzymanie licznika T2
Koniec:
      NOP                       ;koniec programu pomiaru

      END                       ;koniec kodu źródłowego programu
```

Zmierzoną wartość szerokości impulsu należy obliczyć wg. zależności:

τ = CRC - CC1 [μs]

Pytania i problemy

1. Co oznacza skrót CCR określający tryby pracy licznika T2 ?
2. Dlaczego kierunek przepływu danych przez linie portu P1 definiuje programista ?
3. Porównaj właściwości licznika T2 w mikrokontrolerze 8052 i SAB 80515/535.
4. Jakie sygnały wejściowe mogą być zliczane przez licznik T2 ? Jaki jest czas ich trwania ?
5. Które z rejestrów specjalnych SFR współpracują z licznikiem T2 w mikrokontrolerze:
   a) 8052 b) SAB 80515/535
6. Jakie jest przeznaczenie poszczególnych bitów w rejestrze T2CON w mikrokontrolerze 8052 ?
7. Jak interpretować kolejne znaczniki rejestru T2CON, CCEN i IRCON w mikrokontrolerze SAB 80515/535 ?
8. Wyjaśnij, patrząc na rysunki 14-1 i 14-4, w jaki sposób z rejestru RCAP2 i CRC przepisywane są wartości początkowe licznika T2
9. Przeanalizuj sposób generowania impulsów przedstawiony na rysunku 14-6 i działanie programu, w którym czas trwania impulsów wynosi: τ_1=500 μs i τ_2=250 μs. Jakie wartości muszą ulec zmianie, jeśli: τ_1=750 μs i τ_2=125 μs ?
10. Opierając się na rysunku 14-5, na którym przedstawiono sposób generowania dwóch sygnałów prostokątnych o programowanym czasie trwania impulsów, przedstaw własny program, który umożliwi wygenerowanie trzech sygnałów o programowanych współczynnikach wypełnienia impulsów.
11. Potwierdź własnymi obliczeniami podane wartości zakresu modulacji szerokości impulsów (0,195% .. 99,805% przy modulacji 8-bitowej i 0,00076% .. 99,9992% przy modulacji 16-bitowej).
12. W jaki sposób programowo zapamiętać wartość chwilową licznika T2 ? Przedstaw odpowiedni program działania mikrokontrolera. Przeanalizuj czas trwania programu i zmiany licznika T2, które mogą wystąpić w czasie działania tego programu.

13. W jaki sposób programowo odczytywać stan pracującego, 16-bitowego licznika, aby odczytana wartość była zawsze właściwa ?

14. Które zbocza sygnałów zewnętrznych, narastające czy opadające, są zboczami aktywnymi w trybie 0 wpisu wartości chwilowej licznika T2 w mikrokontrolerze SAB 80515/535 ?

15. Jakie jest opóźnienie między momentem wystąpienia aktywnego zbocza sygnału zewnętrznego lub instrukcji wpisującej stan licznika T2, a zmianą zawartości rejestru CCx ?

16. Opierając się na rysunku 14-7 przedstaw sposób pomiaru czasu trwania zera logicznego sygnału doprowadzonego do linii portu P1.0 i P1.1.

17. Jak zmienić program pomiaru czasu trwania impulsu (jedynki logicznej) aby w czasie pomiaru możliwe było przepełnienie licznika T2 ?

15. Obniżanie poboru mocy.

W układach mikroprocesorowych zasilanych z baterii lub akumulatorów dąży się do minimalizacji mocy pobieranej przez układ, by okres po którym należy wymienić lub zregenerować źródło zasilania był jak najdłuższy. Może się również okazać, że w układy mikroprocesorowe nadzorują obiekty, na przykład systemy grzewcze, których parametry zmieniają się wolno w stosunku do szybkości przetwarzania danych przez procesor. W sytuacjach awaryjnych, przy zaniku napięcia w sieci energetycznej, należy przejść na zasilanie rezerwowe, które powinno przynajmniej umożliwić zachowanie podstawowych danych tak, by po ponownym włączeniu zasilania głównego systemy mikroprocesorowe mogły podjąć przerwaną pracę. Podobnych sytuacji jak podane wyżej, wymagających od procesora zmniejszenia mocy zasilania, może być znacznie więcej. Dlatego producenci mikrokontrolerów wprowadzili różne mechanizmy umożliwiające obniżenie mocy pobieranej przez mikrokontroler.

Stosowane metody na obniżanie mocy pobieranej przez procesor mogą być różne, poczynając od zmniejszenia częstotliwości oscylatora - im niższa częstotliwość tym niższy prąd zasilania - poprzez wyłączenia kolejnych układów struktury wewnętrznej procesora, aż po całkowite wyłączenie czynności procesora - pozostaje tylko podtrzymanie zawartości rejestrów i pamięci wewnętrznej. Również różne mogą być systemy zabezpieczeń przed przypadkowym wyłączeniem procesora. Różnice mogą występować między elementami tej samej rodziny lub nawet dla takich samych typów ale wykonanych w różnych technologiach. Opisane poniżej tryby redukcji mocy zasilania mikrokontrolerów dotyczą tylko wersji CMOS - 80C51 i 80C515/535, gdyż są one stosowane znacznie częściej niż wersje HMOS.

W mikrokontrolerach 80C51 i 80C515/535 występują dwa tryby obniżenia poboru mocy:

- tryb „uśpienia" (Idle) polegający na tym, że zostaje wyłączona jednostka centralna procesora - nie są wykonywane rozkazy. Natomiast mogą pracować pozostałe układy struktury wewnętrznej - liczniki, port szeregowy, przetwornik A/C itp., a porty równoległe zachowują swój stan. W mikrokontrolerach 80C515/535 zostaje też zatrzymany Watchdog. Wyprowadzenie procesora ze stanu uśpienia może nastąpić albo przez wywołanie przerwania albo przez zerowanie procesora. Ponieważ w trybie uśpienia pracuje oscylator dlatego na uaktywnienie procesora wystarczą dwa cykle maszynowe.
- tryb wyłączenia (Power Down) polegający na całkowitym zatrzymaniu wszystkich funkcji mikrokontrolera łącznie z oscylatorem. Może być podtrzymana zawartość pamięci wewnętrznej oraz rejestry SFR, w tym również porty równoległe, przy czym napięcie podtrzymujące może mieć niż-

szą wartość, ale na ogół nie mniejszą niż 2 V. Jedynym sposobem wyjścia ze stanu wyłączenia jest zerowanie procesora. Ponieważ w stanie wyłączenia nie pracuje oscylator, to czas resetu powinien być taki sam jak podczas włączania zasilania.

Wprowadzenie procesora w tryby obniżonego poboru mocy odbywa się poprzez ustawienie odpowiednich znaczników w rejestrze PCON

Rejestr PCON w mikrokontrolerze 80C51:

rejestr PCON									adres 87h
	SMOD	X	X	X	GF1	GF0	PD	IDL	rw = 0

Rejestr PCON w mikrokontrolerach 80C515/535:

rejestr PCON									adres 87h
	SMOD	PDS	IDLS	X	GF1	GF0	PDE	IDLE	rw = 0

Wprowadzenie mikrokontrolera 80C51 w stan uśpienia odbywa się przez ustawienie w stan 1 znacznika IDL, np. rozkazem

```
ORL PCON,#1
```

Po tym rozkazie procesor jest wprowadzany w stan uśpienia. Jeżeli z tego stanu jest wyprowadzany poprzez przerwanie, to najpierw jest wykonywany podprogram obsługi przerwania a dopiero potem procesor przechodzi do wykonywania programu umieszczonego za instrukcją wprowadzającą procesor w tryb uśpienia.

Wprowadzenie mikrokontrolera w tryb wyłączenia odbywa się poprzez ustawienie w stan 1 znacznika PD. Jest to ostatnia instrukcja wykonywana przez procesor.

W mikrokontrolerach 80C515/535 wprowadzono dodatkowe zabezpieczenia, sprzętowe i programowe, przed przypadkowym wprowadzeniem procesora w tryby redukcji mocy. Zabezpieczenie sprzętowe jest związane z wyprowadzeniem PE, na którym musi występować stan 0 by procesor można było wprowadzić w tryby redukcji mocy. Natomiast zabezpieczenie programowe polega na konieczności wykonania kolejno dwóch rozkazów:

```
ORL PCON,#1     ;ustawienie w stan 1 znacznika IDLE
ORL PCON,#20H   ;ustawienie w stan 1 znacznika IDLS
```

dla trybu uśpienie i

```
ORL PCON,#2     ;ustawienie w stan 1 znacznika PDE
ORL PCON,#40h   ;ustawienie w stan 1 znacznika PDS
```

dla trybu wyłączenia.

Znacznik SMOD jest związany z przełączaniem częstotliwości taktowania portu szeregowego, a znaczniki GF0 i GF1 są znacznikami ogólnego zastosowania. Trzeba tylko pamiętać, że ponieważ znajdują się one w rejestrze o adresie niepodzielnym przez 8 dlatego ich stan nie może być zmieniany rozkazami bitowymi.

? Pytania i problemy.

1. Dlaczego w mikrokontrolerach są stosowane rozwiązania umożliwiające obniżanie poboru mocy ?
2. Jak zależy wartość prądu zasilania procesora od częstotliwości jego oscylatora ?
3. Na czym polega tryb „uśpienia" IDLE ?
4. Na czym polega tryb wyłączenia Power Down ?
5. Jak wprowadza się mikrokontrolery 80C51 i 80C515 w tryb IDLE i jak z tego trybu można wyjść ?
6. Jak wprowadza się mikrokontrolery 80C51 i 80C515 w tryb Power Down i jak można z tego trybu wyjść ?

16. Watchdog w SAB 80515/535

Zakłócenia impulsowe indukowane w ścieżkach przewodzących sygnały elektryczne na płytce drukowanej, niedoskonała filtracja napięcia w zasilaczach (stabilizatorach napięcia), w szczególności w zasilaczach impulsowych, zmiany fizyko-chemiczne połączeń (utlenianie, korozja elektrochemiczna) mogą prowadzić do zmiany zawartości przesyłanych bajtów lub błędnego adresowania pamięci. Zmiana treści programu powoduje realizację zadań, które nie występują w oryginalnym programie. Jeśli są to tylko procedury wew-nętrznego przetwarzania to efektem takiego programu są błędy obliczeń. Jeśli mikrokontroler steruje obiektem, to zakłócenia jego pracy mogą mieć bardzo poważne konsekwencje. Przygotowując oprogramowanie wprowadza się pewne elementy samokorekcji programu. Ale jeśli zasadniczy program może zostać zniekształcony, to również tak samo podatne na zakłócenia są programy korekcyjne. Najskuteczniejszym sposobem są środki sprzętowe. Takim rozwiązaniem jest zastosowanie dodatkowego układu nadzorującego działanie programu, który charakteryzowałby się następującymi cechami:

- układ raz uruchomiony nie mógłby być programowo zatrzymany,
- program musiałby okresowo zmieniać stan tego układu, odświeżać go, w przeciwnym razie układ generowałby sygnał przerwania o najwyższym dla mikrokontrolera priorytecie; takim sygnałem dla każdego procesora jest sygnał RESET.

Oznacza to, że istniałyby dwa różne źródła zerowania mikrokontrolera:

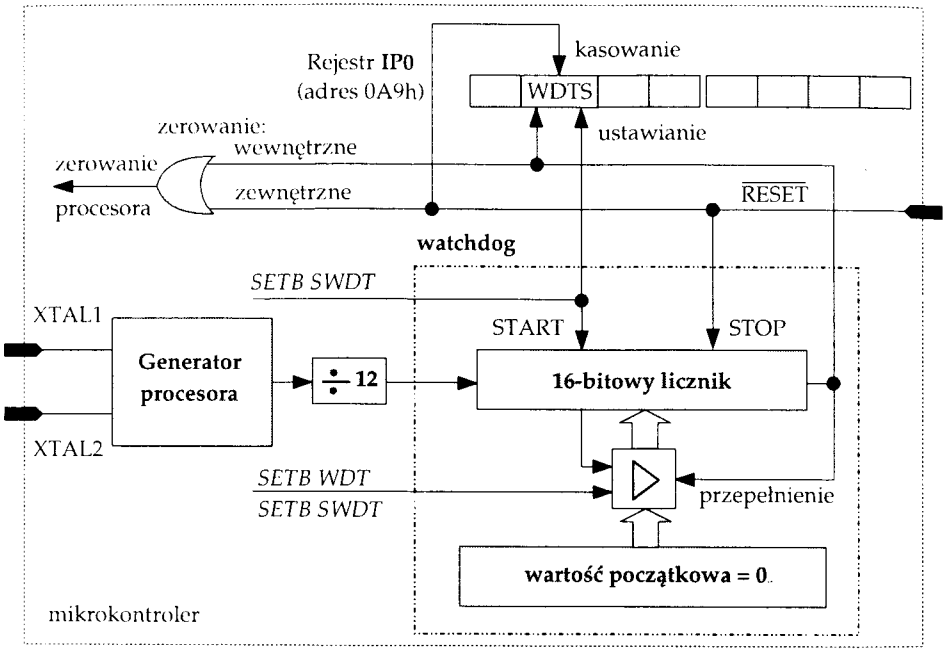
- zerowania zewnętrznym sygnałem doprowadzonym do wypro-wadzenia procesora oznaczonego jako RST (8051/52) lub $\overline{\text{RESET}}$ (SAB 80515/535),
- zerowania wewnętrznym sygnałem generowanym przez układ nadzorujący działanie programu.

W mikrokontrolerach 8051/52 nie ma wewnętrznego układu, który spełniałby powyższe wymagania. Jedynym rozwiązaniem jest dołączenie zewnętrznego układu. W takim przypadku nie ma możliwości rozróżnienia źródła zerowania mikrokontrolera. Dołączenie zewnętrznego elementu do układu przerwań nie do końca rozwiązuje problem. Jeśli zasadniczy program może być zakłócony, to także procedury obsługi przerwań mogą działać nieprawidłowo. W mikrokontrolerach rodziny '51 nie ma przerwań niemaskowalnych.

Konstruktorzy mikrokontrolerów SAB 80515/535 rozwiązali postawiony problem i wyposażyli mikrokontrolery w watchdog.

Watchdog jest układem nadzorującym działanie wykonywanego programu, którego podstawowym elementem jest licznik. Przepełnienie licznika generuje sygnał wewnętrznego zerowania mikrokontrolera.

Licznik watchdoga zlicza każdy cykl maszynowy procesora, po każdym impulsie sterującym, licznik zwiększa swoją zawartość o jeden (jest inkrementowany). Nadzorowany program aby nie dopuścić do przepełnienia licznika zeruje cyklicznie licznik watchdoga. Jeśli z powodu zakłóceń program nie zmieni stanu licznika, to po przepełnieniu licznik generuje wewnętrzny sygnał zerujący procesor.



Rys. 16-1 Struktura wewnętrzna watchdoga.

Programowe uruchomienie watchdoga polega na wykonaniu instrukcji wpisującej jedynkę logiczną do rejestru IEN1 na pozycję znacznika SWDT (Start/refresh WatchDog Timer flag):

```
SETB   SWDT              ;SWDT = IEN1.6 ← 1
```

Znacznik SWDT dostępny jest w rejestrze przerwań **IEN1** o adresie 0B8h (rw-00):

EXEN2	SWDT	EX6	EX5	EX4	EX3	EX2	EADC

Po uruchomieniu licznika Watchdoga wpisywana jest jedynka na pozycję znacznika WDTS (WatchDog Status flag) i licznik jest inkrementowany od wartości początkowej równej zero. Znacznik WDTS dostępny jest w rejestrze priorytetów przerwań **IP0** o adresie 0A9h (rw-00):

-	WDTS	IP0.5	IP0.4	IP3	IP0.2	IP0.1	IP0.0

Zatrzymanie i zerowanie licznika Watchdoga możliwe jest tylko przez zewnętrzne zerowanie mikrokontrolera linią $\overline{\text{RESET}}$. Równocześnie z zerowaniem rejestrów specjalnych SFR kasowany jest także znacznik WDTS.

W celu zwiększenia niezawodności działania mikrokontrolera i zmniejszenia prawdopodobieństwo przypadkowego, okresowego zerowania zawartości licznika watchdoga konieczne jest wykonanie następujących instrukcji w podanej kolejności:

```
SETB   WDT               ;WDT = IEN0.6 ← 1
SETB   SWDT              ;SWDT = IEN1.6 = ← 1
```

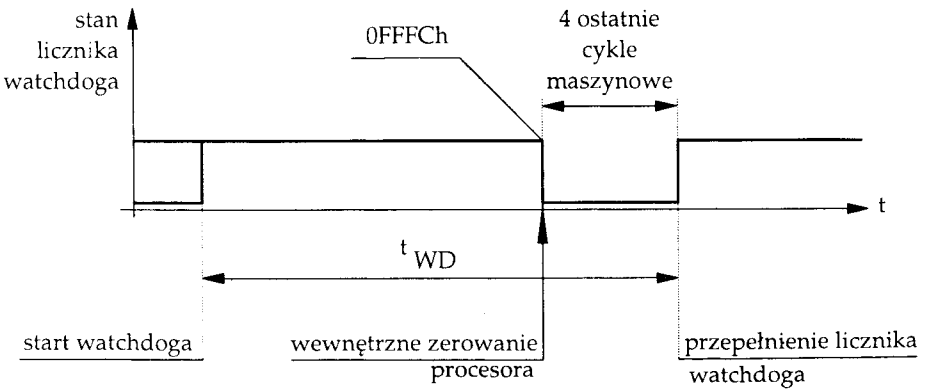
Znacznik WDT dostępny w rejestrze IEN0 jest automatycznie kasowany przez procesor w trzecim cyklu maszynowym po jego ustawieniu.

rejestr przerwań **IEN0** (adres 0A8h), rw-00:

EAL	WDT	ET2	ES	ET1	EX1	ET0	EX0
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Cztery cykle maszynowe przed przepełnieniem licznika generowany jest wewnętrzny sygnał zerowania procesora (rysunek 16-2). Sygnał ten wywołuje inny skutek w stosunku do sprzętowego zerowania procesora linią $\overline{\text{RESET}}$:

- sprzętowe zerowanie linią $\overline{\text{RESET}}$ powoduje:
  → zatrzymanie licznika watchdoga,
  → zerowanie wybranych rejestrów specjalnych SFR, w tym również znacznika WDTS w rejestrze IP0,
- wewnętrzne zerowanie od licznika watchdoga:
  → nie ma wpływu na działanie licznika watchdoga,
  → zeruje rejestry specjalne SFR ale nie zmienia stanu znacznika WDT = 1 w rejestrze IP0.



Rys. 16-2 Zależności czasowe w liczniku watchdoga.

Taki sposób reakcji procesora na oba sygnały zerujące umożliwia programową interpretację źródła zerowania mikrokontrolera:

- jeśli znacznik WDTS = 0, to procesor został wyzerowany sprzętowo linią $\overline{\text{RESET}}$,
- jeśli znacznik WDTS = 1, to procesor został wyzerowany wskutek przepełnienia licznika watchdoga.

Zerowanie licznika watchdoga musi być zrealizowane przed upływem czasu, po którym licznik ten osiągnie stan przepełnienia (pomniejszonego o 4). Jeśli mikrokontroler sterowany jest rezonatorem kwarcowym o częstotliwości drgań własnych równych f_{GEN} = 12 MHz to jeden cykl maszynowy trwa 1 μs. Oznacza to, że jeśli w programie nie zostaną wykonane obie instrukcje wpisujące jedynki logiczne na pozycje bitów WDT i SWDT, to po

65 536 - 4 = 65 532 cyklach maszynowych czyli po 65 532 μs

licznik watchdoga generuje sygnał wewnętrznego zerowania mikrokontrolera. Do okresowego odświeżania licznika watchdoga wykorzystuje się jeden z dostępnych w procesorze liczników ogólnego przeznaczenia, np. licznik T0, T1 lub T2. Licznik taki generuje cykliczne przerwanie, którego zadaniem jest odświeżanie stanu licznika watchdoga. Przerwanie to może zostać wykorzystywane na przykład do odmierzania odcinków czasu, testowania klawiatury, dynamicznej obsługi pola odczytowego, cyklicznego sterowania obiektami itp.

Jeśli do nadzorowania poprawności działania programu w procesorze SAB 80515/535 zastosowano watchdog, to program ten powinien być jak najrzadziej przerywany procedurami zerowania licznika watchdoga. Należy pamiętać, że wewnętrzne zerowanie procesora generowane jest o 4 cykle maszynowe wcześniej niż przepełnienie licznika. W przykładzie do odmierzania czasu, po którym licznik watchdoga ma być zerowany zastosowano licznik T1 mikrokontrolera. Przy obliczaniu okresu t_{WD} powtarzania przerwania od licznika T1 wyrażonego w cyklach maszynowych jako wartość początkową należy przyjąć maksymalną wartość wyrażenia:

$$t_{WD} = (65\,536 + L - TH1, TL1) < 65\,532$$

gdzie L jest liczbą cykli maszynowych potrzebnych do przyjęcia przerwania od licznika T1 i programowego zerowania licznika watchdoga. Jeśli założyć, że wykonywane są tylko instrukcje trwające 2 cykle maszynowe to L=10.

Spełniając powyższe wymagania należy przyjąć TH1, TL1 = 14. Przykład programu spełniającego podane założenia przedstawiono poniżej.


```
        ORG   1BH          ;adres obsługi przerwania od licznika T1
WD_Obsluga_T1:
        MOV   TH1,#0       ;TH1 ← 0,   wartość początkowa
        MOV   TL1,#14      ;TL1 ← 0EH,       licznika T1
        SETB  WDT          ;WDT = IEN0.6 ←1, zerowanie licznika
        SETB  SWDT         ;SWDT = IEN1.6 ←1, watchdoga
        RETI               ;zakończenie obsługi przerwania od licznika T1

WD_Init_1:                 ;inicjalizacja przerwań i licznika T1
        ;rejestr IEN0 o adresie 0A8h:
```

EAL	WDT	ET2	ES	ET1	EX1	ET0	EX0

```
        MOV   IEN0,#88h    ;IEN0 ← 88h = 1000 1000b,
                           ;odblokowanie wszystkich przerwań
                           ;i przerwania od licznika T1
        MOV   TH1,#0       ;TH1 ← 00,  wartość początkowa
        MOV   TL1,#14      ;TL1 ← 0Eh,       licznika T1
        ;rejestr TMOD o adresie 89h:
```

GATE	C/T#	M1	M0	GATE	C/T#	M1	M0
licznik T1				licznik T0			

```
        MOV   TMOD,#50h    ;TMOD ← 50h = 0101 0000b
        SETB  TR1          ;TR1 ←1,          start licznika T1
        SETB  SWDT         ;SWDT ←1,         start watchdoga
        .....              ;dalsza część programu
```

Pytania i problemy.

1. Jaki jest cel stosowania watchdoga w mikrokontrolerach ?
2. W jaki sposób uruchomić i zatrzymać watchdog ?
3. Na czym polega odświeżanie licznika watchdoga i w jaki sposób tego dokonać ?
4. Czym różni się wewnętrzne (od watchdoga) i zewnętrzne (linią $\overline{\text{RESET}}$) zerowanie mikrokontrolera ?
5. W jaki sposób uprościć procedurę WD_Obsluga_T1 obsługi licznika T1 ?

6. W przykładzie założono, że w programie użytkownika wykonywane są tylko instrukcje trwające 2 cykle maszynowe. Jeśli wykonywane są dowolne instrukcje mikrokontrolera, to jaką wartość ma L ?

17. Narzędzia programowania

Każdy procesor posługuje się językiem wewnętrznym, zwanym językiem maszynowym. Język ten charakteryzuje się stosowaniem binarnych kodów rozkazów i binarnych adresów komórek pamięci. Dlatego program działania procesora jest umieszczony w pamięci w postaci kodu binarnego. Pisanie programu bezpośrednio w języku maszynowym, co było robione na początku ery maszyn cyfrowych, jest bardzo uciążliwe ze względu na konieczność obliczania adresów skoków oraz łatwość popełniania błędów. Dlatego opracowano programy komputerowe, które tłumaczą tekst programu napisanego w odpowiednim języku na kod maszynowy procesora.

Język może być prosty, niskiego poziomu. Jest to asembler lub makroasembler. W asemblerze jednemu rozkazowi maszynowemu procesora odpowiada jeden rozkaz symboliczny. Wszystkie podane w tej książce rozkazy, np. MOV A,#51h, poza ich przedstawieniem w postaci kodu binarnego, są właśnie rozkazami symbolicznymi. Oprócz rozkazów symbolicznych w asemblerach stosuje się pseudorozkazy ułatwiające pisanie programów, jak na przykład pseudorozkazy dzielenia, mnożenia itp. przerzucające na asembler wyliczenie pewnych wartości, z których korzysta program. Makroasembler ma wszystkie właściwości asemblera, a ponadto umożliwia zastąpienie powtarzających się sekwencji rozkazów jednym makrorozkazem w postaci zdefiniowanego przez programistę skrótu mnemotechnicznego.

Język asemblera jest bardzo prosty, łatwy do opanowania ale jego wadą jest to, że wszystkie procedury operowania danymi muszą być napisane przez programistę. Wady tej są pozbawione języki wyższego rzędu, na przykład język C, w którym istnieją gotowe procedury mnożenia, dzielenia, obliczania wartości funkcji trygonometrycznych itp.

Napisany program należy sprawdzić, czy będzie działał zgodnie z założeniami. Sprawdzenie częściowe, obejmujące fragmenty programu nie komunikujące się z otoczeniem, można przeprowadzić wykorzystując symulatory, tzn. programy komputerowe symulujące działanie procesora.

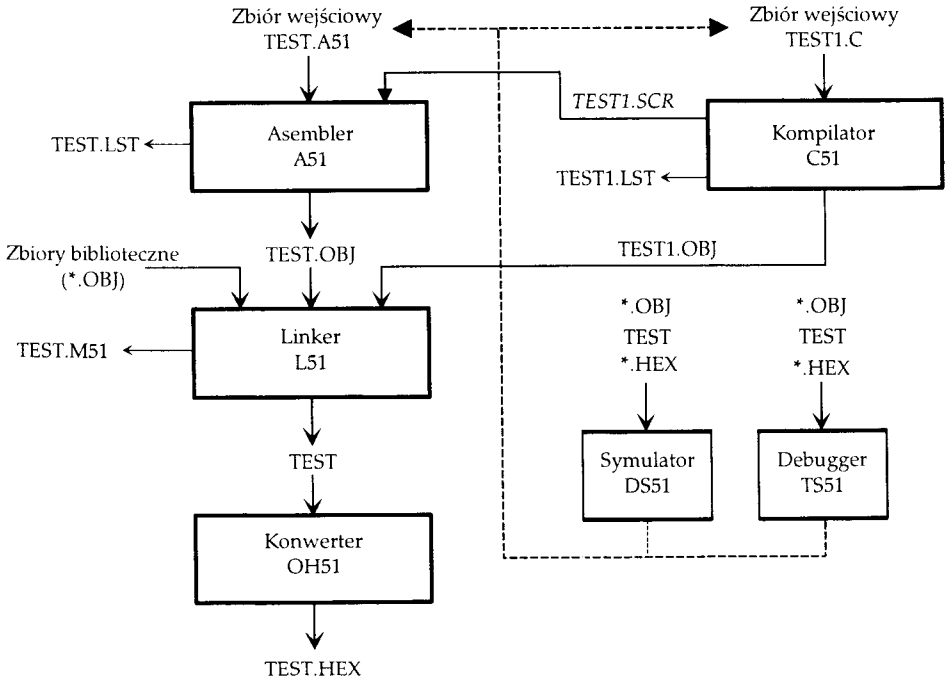
Możliwość sprawdzenia poprawności współpracy procesora z otoczeniem można, przy pewnych ograniczeniach, przeprowadzić przy pomocy debuggerów płytkowych. Są to moduły zawierające procesor, pamięć programu z programem nazywany monitorem i pamięć RAM, która może być traktowana przez procesor jak pamięć programu. Moduł współpracuje z komputerem, za pośrednictwem którego użytkownik może wpisywać do pamięci RAM testowany program. Program monitor zapewnia łączność pomiędzy komputerem a modułem i nadzoruje proces testowania programu użytkowego. W takim systemie jest możliwe wykonanie każdej instrukcji krok po kroku, sprawdzanie działania

programu od dowolnego rozkazu, podglądanie i modyfikacja rejestrów procesora oraz pamięci RAM itp. Do płytki procesora można podłączyć elementy zewnętrzne takie jak klawiatura, pole odczytowe itd.

W przypadku gotowego urządzenia, gdzie nie można skorzystać z debuggera płytkowego, w celu testowania programu, a zwłaszcza współpracy procesora z elementami otoczenia, można zastosować emulator. Jest to urządzenie zawierające specjalny procesor, który oprócz właściwości procesora użytego w przyrządzie, jest wyposażony w dodatkowe mechanizmy kontroli jego pracy. W trakcie testowania urządzenia w podstawkę mikrokontrolera wkłada się złącze łączące urządzenie z emulatorem. Program jest wykonywany przez specjalny procesor emulatora, którego pracę można nadzorować poprzez odpowiednio oprogramowany komputer.

Po sprawdzeniu i przetestowaniu programu należy umieścić go w pamięci programu, wewnętrznej lub zewnętrznej mikrokontrolera. Odbywa się to przy pomocy programatorów, które wymagają podania programu w odpowiednim formacie. Najczęściej tym formatem jest INTEL-STANDARD "Hex".

Jednym z często stosowanych zestawów narzędzi do pisania i testowania programów dla mikrokontrolerów rodziny '51 są narzędzia firmy Keil. Na rysuneku 17-1. jest pokazany zestaw narzędzi dla wersji 4.0.



Rys. 17-1. Zestaw narzędzi programowania firmy Keil.

Zestaw narzędzi firmy Keil składa się z asemblera, kompilatora języka C, linkera, konwertera, symulatora, debuggera i biblioteki z podprogramami. Zbiory wyjściowe z asemblera lub kompilatora z rozszerzeniem .obj mogą być testowane w symulatorze DS51 lub debuggerze TS51. Debbuger TS51 może współpracować z modułem mikroprocesorowym, a więc ma możliwości testowania programu na obiekcie rzeczywistym.

Linker służy do łączenia zbiorów wejściowych w jeden zbiór wynikowy zawierający cały program działania mikrokontrolera. Dzięki temu można łączyć zbiory otrzymane po asemblerze i kompilatorze lub dołączać zbiory biblioteczne.

Konwerter przetwarza zbiór wyjściowy z linkera na zbiór zapisany w standardzie INTEL-HEX stosowanym w programatorach pamięci EPROM lub narzędziach testujących innych firm.

17.1. Asembler firmy Keil

Poniżej są podane podstawowe zasady wymagane przy pisaniu programów dla asemblera firmy Keil. Zasady te mogą być również obowiązujące dla innych asemblerów.

1. o podstawie liczby użytej w programie decyduje ostatni znak liczby:

   - H, h - liczba szesnastkowa 0 ÷ 0FH, np. 0D3Eh

   - B, b - liczba binarna 0 ÷ 1, np. 1101001111B

   - D, d - liczba dziesiętna 0 ÷ 9, np. 2245D

   jeżeli ostatni znak liczby nie jest żadnym z wyżej wymienionych, to przyjmuje się, że jest to liczba dziesiętna,

2. liczby muszą zaczynać się od cyfry. Dlatego liczby w kodzie szesnastkowym zaczynające się od litery muszą być poprzedzone cyfrą 0, np. 0F3h,

3. jeżeli w rozkazie jest wpisana wartość, to musi być poprzedzona znakiem #. W przeciwnym razie asembler potraktuje tę wartość jako adres komórki pamięci wewnętrznej RAM lub rejestru specjalnego, na przykład rozkaz **MOV A,#32h** spowoduje wpisanie do akumulatora liczby o wartości 32 szesnastkowo, a rozkaz **MOV A,32h** przepisze do akumulatora zawartość komórki pamięci wewnętrznej RAM o adresie 32 szesnastkowo.

4. symbole predefiniowane, którymi są nazwy rejestrów, np. A, R0 - R7, TCON itd., elementy rozkazów, np. DIV, MOV, LCALL itd., są słowami zastrzeżonymi. Nie mogą być użyte poza rozkazami, np. jako etykiety,

5. pozostałe symbole (nazwy segmentów, nazwy określające adresy lub wartości liczbowe) nie mogą mieć więcej niż 31 znaków, przy czym pierwszym znakiem musi być litera lub znak podkreślenia '_' albo znak zapytania '?'. Na kolejnych miejscach mogą znajdować się również cyfry.

6. etykiety - symboliczne adresy (nazwy umieszczone w miejscu programu do którego ma nastąpić skok) muszą zaczynać się od lewego marginesu bez spacji i muszą być zakończone dwukropkiem,

7. komentarze muszą być poprzedzone średnikiem. Komentarz nie jest przetwarzany przez asembler, ale jest przenoszony do zbioru generowanego przez asembler zawierającego listing programu lub do narzędzi testujących program. Służy do dokumentowania programu.

Linia programu źródłowego ma najczęściej następującą postać:

[etykieta:] mnemonik rozkazu [operand][,operand][,operand] [;komentarz]

elementy w nawiasach kwadratowych są opcjonalne, liczba operandów zależy od rozkazu, na przykład:

```
PETLA:   DJNE  R0,PETLA   ;opóźnienie w wykonywaniu programu
```

Nazwa wejściowego zbioru źródłowego dla asemblera A51 powinna posiadać rozszerzenie ponieważ zbiór wyjściowy z linkera nie zawiera rozszerzenia i nakryłby się ze zbiorem źródłowym, chyba że przy wywołaniu linkera zostałaby zadeklarowana nazwa zbioru wyjściowego z rozszerzeniem. Rozszerzenie nazwy zbioru źródłowego nie może być takie samo jak rozszerzenia zbiorów wytwarzanych w trakcie przetwarzania zbioru źródłowego do formatu INTEL-HEX, rysunek 17.1. Jeżeli zbiór źródłowy ma nazwę TEST.ASM, to podstawowe wywołanie asemblera, z pominięciem opcji, jest następujące:

A51 TEST.A51

Asembler sprawdza czy zbiór źródłowy nie ma błędów w składni, w pisowni rozkazów i adresowaniu, za nazwy stałych podstawia odpowiednie wartości liczbowe, a za etykiety - odpowiednie adresy (dla segmentów relokowalnych są to adresy tymczasowe do momentu ich ustalenia przez linker). Następnie generuje dwa zbiory wyjściowe. Jeden z rozszerzeniem .lst (listing), w którym zbiór źródłowy jest rozszerzony o adresy i kody rozkazów, numerację wierszy oraz tablicę symboli użytych w zbiorze źródłowym wraz z określeniem ich typu i wartością lub adresem pod którym występują. Zawartość zbioru do listingu zależy od użytych przełączników przy wywołaniu asemblera lub poleceń asemblera umieszczonych w zbiorze źródłowym. Drugi zbiór z rozszerzeniem .obj (object) jest zbiorem wejściowym dla linkera. Może być testowany narzędziami DSCOPE-51 (symulator DS51 i debugger TS51) firmy Keil.

17.2. Wybrane polecenia asemblera

Tekst źródłowy dla asemblera firmy Keil tworzy jeden moduł zaczynający się nazwą modułu i zakończony poleceniem **END**. Nazwa modułu może być zadeklarowana poleceniem **NAME**, na przykład:

```
NAME POMIAR_CZASU
```

Nazwa modułu nie może zawierać spacji, nie może zaczynać się od cyfry i nie może mieć więcej niż 40 znaków. Jeżeli nazwa nie jest deklarowana poleceniem NAME, to jako nazwa modułu zostanie przyjęta nazwa zbioru źródłowego bez rozszerzenia.

Polecenie END jest ostatnim elementem zbioru źródłowego analizowanym przez asembler. Wszystko co jest poza tym słowem jest przez asembler pomijane. Brak tego polecenia jest sygnalizowane przez asembler jako błąd.

Asembler A51 jest bezpośrednio przystosowany do przetwarzania programów odwołujących się do rejestrów mikrokontrolera 8051. Przy pisaniu programów dla mikrokontrolerów zawierających dodatkowe rejestry należy w programie źródłowym powiązać ich nazwy z odpowiadającymi im adresami w obszarze SFR. Można to zrobić albo przez bezpośrednią deklarację używanych rejestrów, np. wprowadzić linie:

```
P5      DATA    0F8H
ADCON   DATA    0D8H
```

albo przez dołączenie odpowiednim poleceniem zbioru zawierającego listę przypisań adresów do rejestrów dla danego mikrokontrolera. Stosowane polecenia zależą od wersji asemblera. Dla wersji V4.4 dołączenie zbioru odbywa się następująco:

```
$NOMOD51
$INCLUDE  (reg515.inc)
```

gdzie: NOMOD51 - dyrektywa asemblera wyłączenia przypisań adresów do rejestrów i znaczników dla mikrokontrolera 80(C)51,
INCLUDE - dyrektywa asemblera dołączenie zbioru o podanej nazwie, w tym przypadku zbioru reg515.inc, zawierającego odpowiednie przypisania dla mikrokontrolera SAB80(C)515.
$ - przełącznik

Zawartość dołączonego zbioru zostanie również umieszczona w zbiorze wyjściowym z rozszerzeniem .lst. Można tego uniknąć przez użycie odpowiedniego polecenia:

```
$NOLIST
$NOMOD51
$INCLUDE  (reg515.inc)
$LIST
```

gdzie: NOLIST - dyrektywa asemblera wskazująca, że wszystkie elementy występujące po niej nie będą dołączone do zbioru z rozszerzeniem .lst.

LIST - dyrektywa asemblera wskazująca że wszystkie elementy występujące po niej będą dołączone do zbioru z rozszerzeniem .lst.

Poleceniem INCLUDE można do zbioru źródłowego dołączać również inne zbiory, zawierające na przykład fragmenty programów.

17.3. Deklaracje segmentów

Elementy programu mogą być podzielone na segmenty związane z obszarem pamięci, w którym mają być umieszczone. Segmenty mogą być absolutne lub relokowalne. Segmenty relokowalne są to segmenty, których adresy są ustalane na etapie łączenia programów przez linker.

Segmenty absolutne są to segmenty, których adresy są ustalone przez programistę na poziomie asemblera. Można je zadeklarować w dwojaki sposób. Pierwszy sposób polega na podaniu typu segmentu z opcją **AT** i adresem od którego segment się rozpoczyna:

- **CSEG AT** adres_absolutny ;segment w obszarze pamięci programu
- **DSEG AT** adres_absolutny ;segment w obszarze wewnętrznej pa-
  ;mięci RAM adresowanej bezpośrednio.
- **ISEG AT** adres_absolutny ;segment w obszarze wewnętrznej pa-
  ;mięci RAM adresowanej pośrednio.
- **XSEG AT** adres_absolutny ;segment w obszarze zewnętrznej pa-
  ;mięci RAM
- **BSEG AT** adres_absolutny ;segment w obszarze pamięci adresowa-
  ;nej bitowo.

Opcja AT kończy rezerwację pola poprzednich obszarów absolutnych pamięci i rozpoczyna nowy obszar z bazą wskazaną 'adres_absolutny'. Taka postać przywołania segmentu absolutnego umożliwia pozostawienie pamięci dla segmentów relokowalnych poniżej wskazanego adresu. Na przykład:

```
CSEG    AT    100h    ;w obszarze od 0 do 100h można umieścić
                      ;segmenty relokowalne
.....                 ;program
```

Program zostanie umieszczony od adresu 100 szesnastkowo.

Drugi sposób polega na podaniu typu segmentu i polecenia **ORG** wraz z adresem określającym początek segmentu. Przy takim zdefiniowaniu segmentu

absolutnego, poniżej adresu wskazanego poleceniem ORG nie można umieszczać segmentów relokowalnych. Na przykład:

```
CSEG
ORG     100h      ;w obszarze od 0 do 100h nie można umieścić
                  ;modułów relokowalnych
.....             ;program
```

W tym przypadku program zostanie również umieszczony od adresu 100 szesnastkowo.

Należy pamiętać, że w przypadku składania segmentów absolutnych sprawdzanie, czy obszary pamięci składanych segmentów nie nachodzą na siebie należy w zasadzie do programisty. Tylko niektóre linkery zgłaszają błąd nakładania się na siebie programów.

W celu umożliwienia łączenia fragmentów programu i optymalizacji zagospodarowania obszarami pamięci moduły programów w zbiorach źródłowych przetwarzanych przez asembler należy podzielić na segmenty relokowalne. Każdy segment jest określony nazwą i deklaracją typu segmentu. Do deklaracji segmentu relokowalnego stosuje się polecenie **SEGMENT**, na przykład:

Nazwa_segmentu **SEGMENT** typ_segmentu

Nazwa segmentu, jeżeli jest związana z tym samym typem segmentu, może powtarzać się w innych modułach łączonych przez linker w jeden wspólny program. Typy segmentów są zdefiniowane w następujący sposób:

- CODE - segment umieszczony w pamięci programu
- DATA - segment umieszczony w wewnętrznej pamięci danych adresowanej bezpośrednio
- IDATA - segment umieszczony w wewnętrznej pamięci danych adresowanej pośrednio
- XDATA - segment umieszczony w zewnętrznej pamięci danych
- BIT - segment umieszczony w wewnętrznej pamięci danych adresowanej bitowo

Polecenie SEGMENT jest tylko deklaracją, że w module występuje segment o podanej nazwie i typie, natomiast właściwy segment relokowalny rozpoczyna się od polecenia **RSEG**:

RSEG Nazwa_segmentu

Linker łączy segmenty w takiej kolejności w jakiej występują polecenia SEGMENT.

W asemblerze występują polecenia **EXTRN** i **PUBLIC**, które umożliwiają wywołanie z jednego modułu podprogramu lub deklaracji zmiennych znajdującego się w innym module lub zbiorze bibliotecznym. Polecenie PUBLIC definiuje związaną z nim nazwę jako nazwę, która może być wywoływana poleceniem EXTRN z innego modułu:

PUBLIC nazwa[, nazwa[....]]

Polecenie EXTRN informuje asembler, że podprogram lub dane związane poprzez nazwę z tym poleceniem będą dołączone przez linker.

EXTRN nazwa [,nazwa[...]]

17.4. Polecenia przypisywania symbolowi wartości

Każdy asembler posiada pewne mechanizmy, operatory, pseudorozkazy ułatwiające pisanie programów źródłowych. Poniżej są omówione podstawowe polecenia i pseudorozkazy stosowane w asemblerze firmy Keil. Niektóre z tych pseudorozkazów są stosowane również w asemblerach innych firm.

EQU

składania:

Nazwa_symbolu **EQU** wyrażenie

Polecenie EQU przypisuje wartość 'wyrażenie' symbolowi 'nazwa_symbolu', np.:

LICZBA **EQU** 0C2h

Nazwie LICZBA została przypisana wartość 0C2h. W module nie wolno drugi raz nadawać symbolowi nowej wartości.

SET

składnia:

Nazwa_symbolu **SET** wyrazenie

Polecenie SET przypisuje wartość 'wyrażenie' symbolowi 'Nazwa symbolu'. W odróżnieniu od polecenia EQU poleceniem SET można wielokrotnie nadać temu samemu symbolowi różne wartości. Na przykład:

```
WSP_A   SET   23
program
WSP_A   SET   45h
```

Nazwie WSP_A została nadana na początku programu wartość 23, a następnie nazwie tej została przypisana wartość 45h.

CODE

składnia:

Nazwa_symbolu **CODE** wyrażenie

Polecenie CODE przypisuje wartość 'wyrażenie', którą jest adres w obszarze pamięci programu, symbolowi 'Nazwa symbolu'. W module nie wolno drugi raz nadawać symbolowi nowej wartości. Zakres wartości wynosi 0 - 65535. Na przykład:

```
START   CODE   0h
RS      CODE   23h
```

Nazwie START został przypisany adres 0, a nazwie RS został, przypisany adres 23h - adres przerwania od portu szeregowego.

DATA

składnia:

Nazwa_symbolu **DATA** wyrażenie

Polecenie DATA przypisuje wartość 'wyrażenie', którą jest adres w obszarze pamięci wewnętrznej RAM adresowanej bezpośrednio, symbolowi 'Nazwa symbolu'. W module nie wolno drugi raz nadać symbolowi nowej wartości. Wartość 'wyrażenie' musi być z przedziału 0 do 255. Na przykład:

```
PORT1   DATA   90h
PORTWE  DATA   0E8h
```

Nazwie PORT1 został przypisany adres 90h, adres portu P1, a nazwie PORTWE - adres 0E8h, adres portu P4.

IDATA

składnia:

Nazwa_symbolu **IDATA** wyrażenie

Polecenie IDATA przypisuje wartość 'wyrażenie' którą jest adres w pamięci wewnętrznej RAM adresowanej pośrednio, symbolowi 'Nazwa symbolu'. W module nie wolno drugi raz nadać symbolowi nowej wartości. Wartość 'wyrażenie' musi być z przedziału 0 do 255. Na przykład:

```
WYNIK1   IDATA   20h
WYNIK2   IDATA   WYNIK1+4
```

Nazwie WYNIK1 został przypisany adres 20h w pamięci wewnętrznej RAM mikrokontrolera, a nazwie WYNIK2 - adres 24h.

XDATA

składnia:

Nazwa_symbolu **XDATA** wyrażenie

Polecenie XDATA przypisuje wartość 'wyrażenie'. którą jest adres w pamięci zewnętrznej RAM, symbolowi 'Nazwa symbolu'. W module nie wolno drugi raz nadać symbolowi nowej wartości. Wartość 'wyrażenie' musi być z przedziału 0 ÷ 65535. Na przykład:

```
TEKST_1   XDATA  10
TEKST_2   XDATA  TEKST_1+5
```

Nazwie TEKST_1 został przypisany adres 10 (dziesiętnie) w pamięci zewnętrznej, a nazwie TEKST_2 - adres 15. Dzięki temu dla zmiennej TEKST_1 pozostawiono 5 bajtów pamięci zewnętrznej RAM.

BIT

składnia:

```
Nazwa_symbolu   BIT   wyrażenie
```

Polecenie BIT przypisuje wartość 'wyrażenie', którą jest adres bitu (w obszarze pamięci wewnętrznej RAM lub w obszarze SFR adresowanych bitowo) symbolowi 'Nazwa symbolu'. W module nie wolno drugi raz nadać symbolowi nowej wartości. Wartość 'wyrażenie' może przyjmować wartości z zakresu 0 - 255. Na przykład:

```
LED   BIT   P1.4
FL1   BIT   40h
FL2   BIT   24h.4
FL3   BIT   FL2+1
```

17.5 Polecenia inicjacji i rezerwacji obszarów pamięci

DB

składnia:

```
[etykieta:]   DB   wyrażenie[,wyrażenie...]
```

Polecenie powoduje umieszczenie w kolejnych bajtach pamięci programu wartości wskazanych przez 'wyrażenie'. Wartość 'wyrażenie' musi być z przedziału -128 do 255. Polecenie dopuszcza umieszczenie zamiast wyrażenia numerycznego łańcucha znaków dowolnej długości umieszczonego pomiędzy apostrofami. W celu umieszczenia w łańcuchu apostrofu należy umieścić dwa apostrofy. Asembler zamienia każdy znak łańcucha na odpowiadającą mu wartość kodu ASCII. Na przykład:

```
TAB:    DB   24, 24H, 1001001B, 255
TEKST:  DB   'To jest apostrof''
```

W obszarze pamięci programu, od adresu określonego etykietą TAB, zostaną umieszczone wartości znajdujące się za poleceniem DB. Natomiast od adresu określonego etykietą TEKST zostaną umieszczone kody ASCII tekstu znajdującego się pomiędzy apostrofami.

DW

składnia:

```
[etykieta:]   DW   wyrażenie[, wyrażenie...]
```

Polecenie to działa podobnie jak polecenie DB, ale powoduje umieszczenie w kolejnych bajtach pamięci programu wartości 16-bitowej wskazanej przez 'wyrażenie'. Wartość wyrażenia musi być z przedziału -32768 do 65535. Można również deklarować łańcuchy znakowe. Na przykład:

```
TAB_W:   DW   1234, 233AH, 'QW'
```

DS

składnia:

```
[etykieta:]   DS   wyrażenie
```

Polecenie rezerwuje w obszarze pamięci RAM, wskazanym typem segmentu, liczbę bajtów określoną przez 'wyrażenie'. Wartość 'wyrażenie' musi być zdefiniowana w chwili wykonania polecenia. Na przykład:

```
DANE:   DS   2
CZAS:   DS   4
```

W pamięci RAM, od adresu określonego etykietą DANE, zostaną zarezerwowane 2 bajty pamięci, a od adresu określonego etykietą CZAS - 4 bajty pamięci RAM.

DBIT

składnia:

```
[etykieta:]   DBIT   wyrażenie
```

Polecenie rezerwuje liczbę bitów w obszarach adresowanych bitowo określoną przez 'wyrażenie'. Wartość 'wyrażenie' musi być zdefiniowana w chwili wykonania polecenia. Na przykład:

```
FLAGA1:   DBIT   1
TEMP:     DBIT   4
```

W obszarze adresowanym bitowo, od adresu określonego etykietą FLAGA1 zostanie zarezerwowany 1 bit, a od adresu określonego etykietą TEMP - 4 bity.

17.6. Linker

Linker wykonuje następujące operacje:

- łączy wielu modułów w jeden moduł,
- łączy relokowalne segmenty o tej samej nazwie w jeden segment,
- optymalizuje obszary pamięci przydzielane dla poszczególnych segmentów,
- łączy wywołania podprogramów zaznaczone jako EXTRN w jednym module z podprogramami o tej samej nazwie oznaczonymi jako PUBLIC w innym module,
- ustala adresy absolutne dla segmentów relokowalnych,
- tworzy zbiór wyjściowy zawierający cały program,
- tworzy zbiór zawierający informacje o procedurach wykonanych przez linker oraz zawierający tablicę użytych symboli i nazw związanych z poleceniami PUBLIC i EXTRN,
- wykrywa błędy występujące w trakcie łączenia programów.

Zbiory wejściowe są przetwarzane przez linker w kolejności podanej w linii wywołującej linker. Jeżeli są dołączane zbiory biblioteczne, to będą z nich wybrane tylko moduły, których nazwy związane z poleceniem PUBLIC będą odpowiadały nazwom związanym z poleceniem EXTRN zbiorów wejściowych.

Wywołanie linkera odbywa się przez podanie jego nazwy, listy nazw wraz z rozszerzeniami zbiorów wejściowych w kolejności w jakiej mają być łączone poszczególne moduły, opcjonalnie nazwy zbioru wyjściowego oraz opcjonalnie listy komend sterujących pracą linkera:

L51 list_zb_wej [**TO** zb_wyj] [komendy]

Przy dużej liczbie zbiorów wejściowych linia wywołania linkera może przekroczyć 128 znaków dopuszczalnych dla poleceń DOS-a. Należy ją w takim przypadku podzielić stawiając w miejscu podziału znak & i naciskając klawisz ENTER. Linker wyświetli w nowej linii znak >> od którego można wprowadzać dalszy ciąg linii wywołania. Znak & może być umieszczony tylko pomiędzy komendami lub parametrami, natomiast nie może być wstawiony w środku komendy lub parametru. Przy długiej linii wywołania linkera wygodniej jest użyć zbioru sterującego pracą linkera. Wywołanie linkera w tym przypadku jest następujące:

L51 @nazwa_zb

gdzie: nazwa_zb jest nazwą zbioru sterującego.

Lista zbiorów wejściowych zawiera nazwy zbiorów, wraz z rozszerzeniami (.obj lub .lib), zawierające moduły, które mają być łączone przez linker oraz nazwy modułów, podane w nawiasach, które znajdują się w zbiorze bibliotecznym. Jeżeli ze zbiorów bibliotecznych są wybierane elementy programów wskazywane przez polecenia PUBLIC i EXTRN, to nazwy modułów są pomijane. Poszczególne elementy listy są oddzielone przecinkiem. Sposób podawania listy zbiorów wejściowych ilustrują poniższe przykłady:

L51 TEST.OBJ, A_C.OBJ, MAT.LIB

W powyższym przykładzie będą łączone moduły zawarte w zbiorach TEST.OBJ i A_C.OBJ oraz elementy ze zbioru bibliotecznego MAT.LIB wybrane poleceniem EXTRN, których nazwy odpowiadają nazwom powiązanym z poleceniem PUBLIC. W wywołaniu

L51 A_C.OBJ, I_O.LIB (LCD, KLAW)

zostaną połączone moduły ze zbioru A_C.OBJ i moduły LCD i KLAW ze zbioru bibliotecznego I_O.LIB.

W obu powyższych przykładach nazwa zbioru wyjściowego będzie nazwą pierwszego zbioru wejściowego bez rozszerzenia. Jeżeli nazwa zbioru wyjściowego ma być zmieniona to należy to zadeklarować w linii wywołania linkera w sposób następujący:

L51 A_C.OBJ, I_O.LIB (LCD, KLAW) **TO** STER.ABS

Zbiór wyjściowy będzie miał nazwę STER z rozszerzeniem ABS.

Oprócz zbioru wyjściowego zawierającego program, linker generuje zbiór z rozszerzeniem .m51, w którym jest podany sposób wywołania linkera, nazwy łączonych modułów oraz nazwy, typy i rozmieszczenie segmentów. Zbiór wyjściowy z linkera może być testowany i sprawdzany narzędziami DSCOPE51.

Jeżeli zbiór z programem ma być zapisany w standardzie INTEL-HEX, to należy użyć konwertera OH51 wywołując go wraz ze zbiorem otrzymanym po linkerze. Wynikiem przetwarzania będzie zbiór z nazwą zbioru wejściowego z rozszerzeniem .hex.

Pytania i problemy

1. Co to jest asembler i makroasembler oraz do czego służą ?

2. Czym się różnią języki niskiego rzędu od języków wysokiego rzędu ?

3. Jakie kroki należy wykonać dla otrzymania poprawnie działającego programu ?

4. Co zawiera zestaw narzędzi programowania firmy Keil ?

5. Jakie są podstawowe zasady pisania programów w języku asemblera ?

6. Co to są symbole predefiniowane ?

7. Co to są etykiety ?

8. Podać do czego służą podstawowe polecenia asemblera.

9. W jaki sposób deklaruje się segmenty ?

10. Jak umieścić program pod ściśle określonym adresem ?

11. Jakie powinny być spełnione warunki by linker mógł połączyć dwa programy ?

12. Jak nazwom przypisuje się wartości i adresy bajtowe i bitowe ?

13. W jaki sposób dokonuje się rezerwacji pamięci dla zmiennych ?

14. Do czego jest potrzebny zbiór „listing" (*.lst) ?

15. Jakie jest zadanie linkera ?

16. Jaka jest zasada umieszczania przez linker zbiorów wejściowych w zbiorze wyjściowym ?

17. W jaki sposób można wywoływać linker ?

18. Do czego służy konwerter OH51 ?

Dodatek A. Alfabetyczna lista instrukcji - gdzie szukać w książce

Dodatek B. Tematyczna lista instrukcji - gdzie szukać w książce

Instrukcje wymiany danych

Instrukcje arytmetyczne

Instrukcje manipulacji bitowych

Instrukcje logiczne

Instrukcje skoków i wywołań podprogramów

Dodatek C. Rejestry i rejestry specjalne SFR układ alfabetyczny

Rejstry wspólne dla mikrokontrolerów rodziny '51:

Rejstry specjalne mikrokontrolera 8x(C)51/52:

Rejstry specjalne mikrokontrolera 8x(C)52:

Rejstry specjalne mikrokontrolera SAB 80(C)515/535:

Dodatek D. Rejestry i rejestry specjalne SFR układ tematyczny

Rejstry ogólnego przeznaczenia

Porty

Przerwania

Port szeregowy

Licznik T0/T1

Licznik T2 w 8052

Licznik T2 w 80515/535

Przetwornik A/C w 80515/535

Redukcja mocy

Watchdog w 80515/535

Literatura:

1. Doliński J.: Mikrokomputer jednoukładowy INTEL 8051. PLJ, Warszawa 1993.

2. Janiczek J., Stępień A.: Systemy Mikroprocesorowe. Mikrokontroler 80(C)51/52. Wydawnictwo Elektronicznych Zakładów Naukowych, Wrocław 1995.

3. Janiczek J., Stępień A.: Systemy Mikroprocesorowe. Mikrokontroler SAB 80(C)515/535. Wydawnictwo Elektronicznych Zakładów Naukowych, Wrocław 1995.

4. Janiczek J., Stępień A.: Systemy Mikroprocesorowe. Laboratorium systemów mikroporcesorowych, cz. I. Wydawnictwo Elektronicznych Zakładów Naukowych, Wrocław 1995.

5. Janiczek J., Stępień A.: Systemy Mikroprocesorowe. Laboratorium systemów mikroporcesorowych, cz. II. Wydawnictwo Centrum Kształcenia Praktycznego, Wrocław 1996.

6. Majewski J., Kardach K.: Mikrokontrolery jednoukładowe 8051. Programowanie w języku C w przykładach. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 1995.

7. Rydzewski A.: Mikrokomputery jednoukładowe rodziny MCS-51. WNT, Warszawa 1992.

8. Katalogi mikrokontrolerów firm (dostępne także w Internecie):
   - → AMD (http://www.amd.com),
   - → ATMEL(http://www.atmel.com),
   - → DALLAS (http://www.dalsemi.com),
   - → INTEL (http://www.intel.com/design/mcs51),
   - → MATRA-HARRIS (http://www.temic.de/static/c51.html),
   - → PHILIPS (http://www.semiconductors.philips.com),
   - → SIEMENS (http://www.sci.siemens.com),