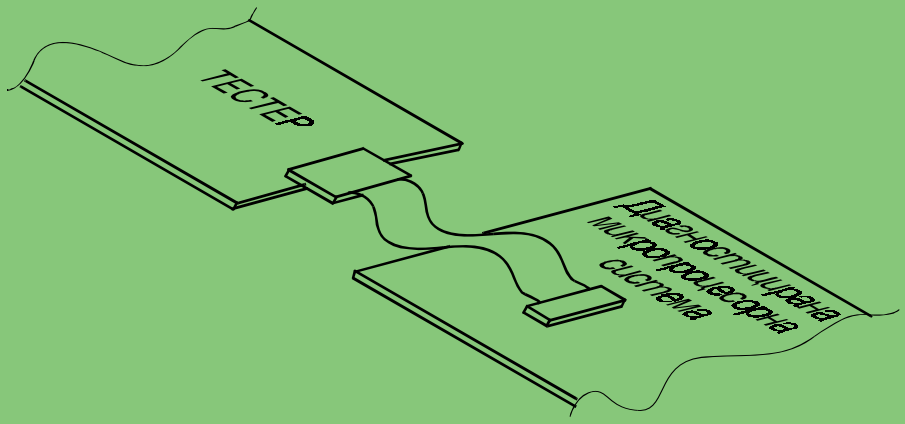
ГЕОРГИ МИХОВ

НАСТРОЙКА И ДИАГНОСТИКА НА МИКРОПРОЦЕСОРНИ СИСТЕМИ



ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ

Доц. д-р инж. **ГЕОРГИ С. МИХОВ**

НАСТРОЙКА И ДИАГНОСТИКА НА МИКРОПРОЦЕСОРНИ СИСТЕМИ

ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ
2003

В учебника са изложени въпроси, свързани с теоретичните и практическите аспекти на настройката и диагностиката на микропроцесорни системи, а също и на използуваните методи и контролно-диагностична апаратура. Разгледани са тестери за статични въздействия, вътрешносхемни микропроцесорни и ROM емулатори, логически и сигнатурни анализатори, програматори, прости контролно-измервателни уреди и др. Внимание е обърнато на развойните средствата за диагностика на микропроцесорните системи в реално време и на съвместната настройка на апаратната и програмната им част. Представени са новите методи за контрол и диагностика на микропроцесорни системи с използуването на гранично сканираща логика. Дадени са конструктивни сведения за онези блокове от контролно-диагностичната апаратура, които определят нейните функции.

При представянето на материала са преследвани две основни цели: получаване на познания за провеждането на настройка и диагностика на микропроцесорни системи; получаване на познания за устройството на електронните диагностични уреди, с цел тяхното ползуване и разработване.

Учебникът е предназначен за студенти от висшите технически учебни заведения. Той може да бъде използуван и от специалисти, работещи в тази област.

1. ВЪВЕДЕНИЕ

Микропроцесорните системи представляват особен клас електронна цифрова апаратура, при който функциите се реализират по програмен път чрез алгоритъма на работа. Веднага с появата им започва разработването на специфична за тях контролно-диагностична апаратура, която непрекъснато се усъвършенствува. Успоредно с нея за диагностиката на микропроцесорни системи продължават да се използуват и традиционните методи и апаратура, характерни за обикновената електронна техника.

Диагностиката на микропроцесорни системи изисква висококвалифициран персонал. При провеждането й е необходимо преди всичко да се познава отлично самото устройство като апаратна част, а също и принципът на функционирането му. Съществена помощ при диагностиката оказват принципната електрическа схема, алгоритъмът на работа, а също и друга документация като например таблиците на разпределение на адресното пространство, листинга с работната програма и др.

Много от проблемите при диагностиката на микропроцесорни системи са характерни за повечето електронни устройства. Най-често това са:

– повреди в захранващия източник;
– лоши електрически съединения;
– замърсени механични контакти;
– неизправни елементи (резистори, кондензатори и др.).

Освен тези проблеми от общ характер, при микропроцесорните системи съществуват редица особености, които са предпоставка за проявление на друг тип неизправности.

1.1. Особености на микропроцесорната диагностика

Апаратните, конструктивните и технологичните особености на микропроцесорните системи силно специфицират диагностиката им. Тя е затруднена от редица причини, по-важните от които са изброени по-долу.

Значителна сложност на изграждащите ги интегрални схеми. Вътрешните елементи в схемите от микропроцесорните фамилии достигат стотици хиляди. Това определя голям брой състояния на интегралната схема. Общият брой на състоянията на една интегрална схема може да се определи като $C=2^k$, където k е броят на вътрешните ключови елементи (всеки ключов елемент може да заема само две състояния – *0* и *1*). Не всички състояния обаче се използуват, т.е. съществуват голям брой невъзможни състояния, които не трябва да се отчитат. Специално за микропроцесорите може да се приложи друга формула, даваща приблизителна представа за общите му състояния, а именно $C=2^{mn}$, където m е разредността на микропроцесора, а n – броят на командите, т.е. всяка команда

се спряга с всяка двоична комбинация на данните.

Големият брой състояния не може да бъде изцяло тестван. Фирмите производителки подлагат на тест само ограничено подмножество състояния и някои неизправности могат да останат неоткрити. Това са най-неприятните неизправности, обусловени от взаимното влияние на отделните елементи, които водят до редки, непредсказуеми откази. Тези откази са свързани с конкретното съчетание на състоянията на вътрешните елементи и действуващите върху тях външни сигнали – проявява се т.нар. ефект на "чувствителност към конкретна ситуация".

Ограничен брой на възможните контролни точки. Потребителят няма физически достъп до вътрешните линии и магистрали в елементите на микропроцесорните системи. Въздействието върху вътрешните елементи и диагностиката на тяхното състояние се извършва косвено, като достъпът е възможен само по програмен път.

Апаратна цялост и неразделност на микропроцесорните системи. Понякога микропроцесорните системи трудно могат да бъдат разделени на функционални възли за отделна проверка на всеки от тях. Често цялата микропроцесорна система или значителни части от нея са конструктивно изработени на една платка, която, разбира се, не може да бъде разделена на части. Освен това тенденция в микроелектрониката е да се съчетават в една интегрална схема различни функционални устройства.

Магистрална организация на линиите и необходимост от едновременно следене на състоянията им. Към системните магистрали на микропроцесорните системи са включени паралелно схемите и блоковете на системата. При откриване на некоректна информация на някоя от линиите е необходимо допълнително да се изясни каква е причината и кой е вносител на некоректността или повредата.

Информацията по магистралите бързо се сменя и зависи от конкретното действие, което микропроцесорната система извършва, т.е. почти липсва повтаряемост на сигналите. Освен това по магистралите невинаги има смислена информация – съществуват преходни процеси, а при определени условия магистралите могат да застават и във високоимпедансно състояние. Получаването на информация за работата на микропроцесорната система от магистралите й обуславя необходимостта от едновременно следене на сигналите по голям брой линии, което изисква използването на специални апаратни средства и персонал с висока квалификация.

Апаратно и програмно единство на микропроцесорните системи. Функциите на тези системи се реализират по програмен път. Затова разделното диагностициране на апаратната и програмната част не могат да гарантират цялостната работоспособност на устройството.

Изброяването на всички тези особености показва, че само с използването

на традиционно оборудване трудно може да се извърши диагностика на микропроцесорни системи. В повечето случаи е необходима специална апаратура - закупена от фирма производител или изработена за конкретен случай от самия проектант или потребител на микропроцесорни системи.

Едновременно с указаните причини, затрудняващи извършването на диагностика, съществуват и някои фактори, които улесняват нейното провеждане. По-важните от тях са следните:

Способност за самодиагностика. Функциите на една микропроцесорна система могат да бъдат изменяни по програмен път. Чрез подходяща програма микропроцесорът може да събира и обработва информация за състоянието на елементите от системата и да информира потребителя за евентуални неизправности.

Стандартна форма на електрическите сигнали. Тази привилегия е присъща на всички електронни цифрови устройства. Тя позволява да се опрости контролът на състоянията на сигналите, свеждайки го до следене на логически състояния *0* и *1*.

Съществуват два основни подхода за проверка на изправността на микропроцесорна система – функционален и структурен.

Функционалният подход се основава на проверка на правилната работа на микропроцесорната система според алгоритъма на нейната работа. Следи се правилното функциониране на системата при решаването на конкретните потребителски задачи. Функционалният подход не може да локализира значителна част от неизправностите и затова се прилага тогава когато липсва информация за причините и характера на неизправностите, а също при голяма сложност на системата или при принизени изисквания към диагностиката.

Структурният подход се основава на данни за структурата на диагностицираната микропроцесорна система, нейната апаратна същност и характера на възможните неизправности. Държи се сметка преди всичко за схемотехничната изправност на системата. Като правило структурният подход осигурява достатъчно пълна проверка на работоспособността на микропроцесорните системи. Същевременно той е по-трудно изпълним поради големия брой елементи и недостатъчна предварителна информация за всички възможни неизправности.

В зависимост от разглеждането на проверяваната микропроцесорна апаратура като цяла или съставена от части се различават системна и модулна диагностика.

При системната диагностика апаратурата се разглежда като единна, за която се използува съответен тестови подход.

При модулната диагностика проверяваната апаратура се разглежда като съвкупност от модули, за всеки от които се използува съответен подход. Тази диагностика се извършва по-лесно и е по-разпространена, но трябва да се има предвид, че работоспособността на отделните модули (възли) не гарантира ра-

ботоспособността на апаратурата като цяло.

1.2. Етапи при микропроцесорната диагностика

От гледна точка на прилагането на диагностиката на микропроцесорните системи могат да се обособят следните основни етапи в провеждането ѝ.

1. Процес на разработка на микропроцесорната система. Диагностиката на този етап (проектантска диагностика) има за цел да осигури настройване и първоначално пускане в действие на системата. Това е най-трудната диагностика, характеризираща се със следните особености:

– наличие на неизправности, свързани с грешка на проектанта;

– голяма вероятност за поява на няколко неизправности едновременно;

– едновременно наличие на неизправности от апаратно и от програмно естество.

2. Процес на производство. Диагностиката на този етап (производствена диагностика) изхожда от презумпцията, че системата по принцип е работоспособна и е необходимо да се отстранят евентуално възникналите неизправности вследствие използуването на дефектни елементи или нарушаване на производствена технология.

3. Процес на експлоатация. Този процес се характеризира с две задачи. Първата е свързана с наблюдаване на работата на микропроцесорната система и непрекъснат контрол на нейната изправност (работна диагностика). Втората задача цели ремонтирането на системата при установяване на повреда (сервизна диагностика). Тази диагностика е най-лека, тъй като се знае, че конкретната система е работоспособна и се търси повреда.

Тестването на микропроцесорните системи се извършва най-често с помощта на специализирана контролно-диагностична апаратура. Тя решава задачи по генериране на входни въздействия, по регистриране на изходни реакции, по анализ на резултатите от проверката и др. При това е необходимо контролно-диагностична апаратура да притежава възможност за работа както с апаратното, така и с програмното осигуряване на микропроцесорните системи.

Конкретният метод за диагностика и съставът на необходимата контролно-диагностична апаратура се определят в зависимост от задачите на диагностиката при всяка отделна микропроцесорна система.

2. СТРУКТУРА НА МИКРОПРОЦЕСОРНИ СИСТЕМИ

Микропроцесорната система представлява универсално цифрово устройство, чиято функция се реализира по апаратно-програмен път, т.е. с апаратни средства, управлявани от програмно осигуряване, което е организиран набор от програми и данни. Използуваните изделия на базата на микропроцесорни системи могат да се разделят на *универсални микрокомпютри* и *специализирани цифрови управляващи устройства* за обработка на информация и управление в реално време – *микроконтролери*. Типичен пример за универсален микрокомпютър представлява персоналният компютър (PC).

Разликата между микрокомпютрите и микроконтролерите е съществена. Терминът *микроконтролер* се използува за означаване на устройство, в което неголяма микропроцесорна система заедно с други необходими елементи, се използува за специализирано управление на специфичен процес или апарат. В това приложение микропроцесорната система заменя сложна схема, състояща се от логически елементи, тригери, аналогово-цифрови и цифрово-аналогови преобразуватели. Именно в този аспект ще бъдат разгледани микропроцесорните системи в настоящата глава. За такъв род приложения съществуват и специализирани микропроцесорни системи, характеризиращи се с това, че техните основни компоненти, обикновено изграждани с допълнителни интегрални схеми, се разполагат на един кристал – те се наричат *едночипови микроконтролери* (*интегрални микроконтролери*). В този смисъл се използува и понятието *едночипов микрокомпютър*.

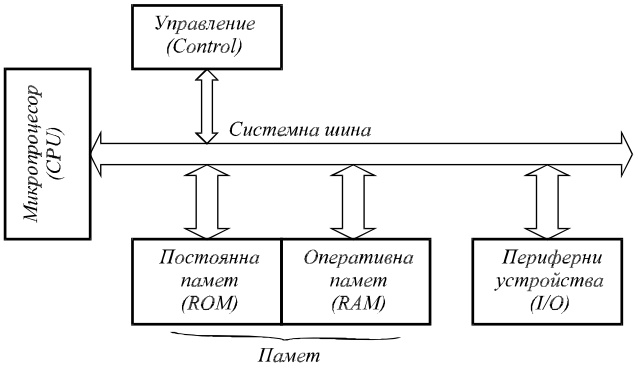
Съществува още едно модерно направление в приложната електроника, където универсални микрокомпютри се допълват с апаратна и програмна част за извършване на конкретна работа по управление на обект – направлението на *микрокомпютърните приложни системи*.

2.1. Състав на микропроцесорна система

Микропроцесорната система се състои от микропроцесор (CPU, Central Processing Unit – централен процесор), памет, периферни устройства, както и от необходимата логика, осигуряваща съвместната им работа – фиг. 2.1.

Микропроцесорът е устройство за обработка на данни и не се използува самостоятелно. За неговата работа е необходима постоянна памет, съхраняваща последователността от командите, които да изпълнява, (програмата). Нужно е тази памет да бъде енергонезависима, т.е. да не губи съдържанието си при изключване на захранването. От нея микропроцесорът само чете. За съхраняване на променливи е необходима още и оперативна памет, във и от която микропроцесорът пише и чете. Обменът на информация с външните системи се извършва с помощта на периферни устройства, които представляват специализирани схеми

за въвеждане и извеждане на информация (портове).



Фиг. 2.1. Състав на микропроцесорна система.

За организация на апаратната връзка между частите на микропроцесорната система и за тяхното взаимодействие са необходими допълнителни устройства, изпълняващи функции за генериране на тактови сигнали, за начално установяване на системата, за избор на устройствата, за буфериране на сигналите, както и друга управляваща логика. Компонентите на микропроцесорната система се свързват помежду си с множество сигнали, наричани най-общо системна шина.

2.2. Системна шина

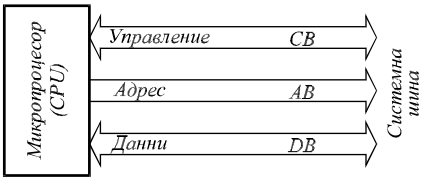
Системната шина се състои от три групи линии (магистрали):

– магистралата за данни *DB* (Data Bus) служи за паралелен обмен на двоична информация между компонентите в микропроцесорната система. Броят на линиите в тази магистрала (ширина на магистралата) определя разредността на системната шина. От нея зависи максималното число, което може да бъде представено в двоична форма. Всеки от компонентите на микропроцесорната система може да предава или приема информация по магистралата за данни, която е двупосочна;

– адресната магистрала *AB* (Address Bus) се използува за предаване на адресите към компонентите в микропроцесорната система. В простите системи адресът може да бъде издаван само от микропроцесора, затова обикновено тази магистрала е еднопосочна;

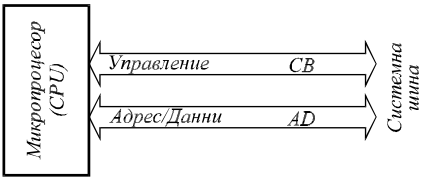
– управляващата магистрала *CB* (Control Bus) съдържа сигнали, реализиращи функциите по управление на работата на системата. Въвеждането на данни в микропроцесора е операция четене (Read), а извеждането – операция запис (Write). Това са най-важните операции, които се извършват в микропроцесорната система. Микропроцесорът определя типа на операцията чрез сигнали от ли-

нията на управление.



Фиг. 2.2. Структура на системната шина на микропроцесорна система.

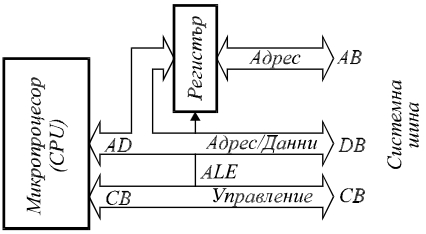
Някои микропроцесори, например тези на Motorola, указват операция четене или запис с една-единствена линия $R/\overline{W}$ – Read/Write. В този случай *1* на тази линия указва, че се извършва четене, а *0* – че се извършва запис. Този сигнал обаче задължително се придружава от втори сигнал, указващ валидност на данните (най-често означаван с *E* – Enable). В други микропроцесори, например тези на Intel, за указване на операцията четене или запис се използуват два разделни сигнала $\overline{RD}$ и $\overline{WR}$ – Read и Write, и двата от които са в неактивно състояние при липса на прехвърляне на данни. В зависимост от вида на операцията се активира само един от тях.



Фиг.2.3. Микропроцесорна системна шина с мултиплексирани адреси и данни.

Не всички линии в управляващата магистрала се изработват от микропроцесора и са изходящи за него. Някои от управляващите сигнали се приемат от микропроцесора, като например сигналите за прекъсване, за заявка за магистралите и др. Структура на системна шина от такъв тип е показана на фиг. 2.2.

В някои микропроцесори, с цел съкращаване на броя на изводите, се прилага съвместна магистрала за адреси и данни (мултиплексирана магистрала) *AD* (Address/Data Bus), по която се предават както адресът (или част от него), така и данните – фиг. 2.3.



Фиг. 2.4. Демултиплексиране на микропроцесорна системна шина.

Етапът на предаване на адресната информация е отделен по време от етапа на предаване на данните и се придружава от специален сигнал (обикновено означаван с *ALE* – Address Latch Enable), който е включен в състава на управляващата магистрала. Под негово управление адресът се запомня (стробира) във външен за микропроцесора демултиплексиращ паралелен регистър. Същес-

твуват системи, при които демултиплексиращият регистър е включен към всеки от компонентите на микропроцесорната система и всички те работят с мултиплексираната системна шина.

Най-често обаче се прилага вариантът, при който се използува един-единствен демултиплексиращ регистър за цялата микропроцесорна система. Такова преобразуване е показано на фиг. 2.4. Веднага след микропроцесора (микроконтролера) демултиплексираната системна шина се разпространява до всички компоненти на микропроцесорната система. Възможен е и смесен вариант, при който част от компонентите в микропроцесорната система ползуват мултиплексираната шина, а други – демултиплексираната.

2.3. Адресно пространство

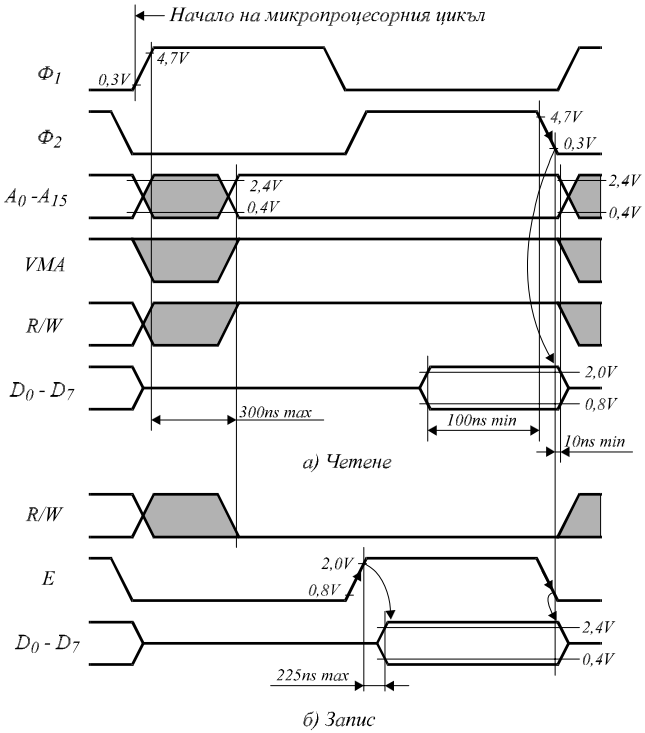
Повечето микропроцесори, например тези на Motorola, имат общо адресно пространство за паметта и за входно-изходните устройства. Четенето от входно-изходно устройство и записът в него се осъществяват точно така, както към клетка от паметта – със същите команди. Недостатък на това решение е необходимостта от отделянето на адресно пространство за входно-изходните устройства от общото.

Алтернативното решение, което се прилага от някои фирми, например Intel, е формирането на отделни адресни пространства за паметта и за входно-изходните устройства. То позволява периферията да се разполага отделно от паметта. В управляващата магистрала са включени допълнителни сигнали за указване на обръщение към различните пространства. Освен това в системата команди са предвидени специални команди за операции с входно-изходните устройства, които по принцип притежават по-малки възможности в сравнение с командите за обръщение към паметта. Необходимо е обаче да се отбележи, че при микропроцесори, имащи отделно входно-изходно пространство, не е задължително входно-изходните устройства да се разполагат в него. Всичките или част от тях могат да бъдат разположени в адресното пространство на паметта и да ползуват инструкциите за обръщение към нея.

В някои микропроцесори (микроконтролери) съществува разделение между адресно пространство за програмата (програмна памет) и за данните (даннова памет). То задължително изисква отделни управляващи сигнали за достъп до всяко от тях.

2.4. Цикли на обмен на информация

Операциите по прехвърляне на информация между микропроцесора и останалите компоненти на микропроцесорната система протичат в определена последователност. Между събитията се формират съответните интервали с разчет до началото на прехвърлянето на данните те да придобият устойчиво статично състояние.



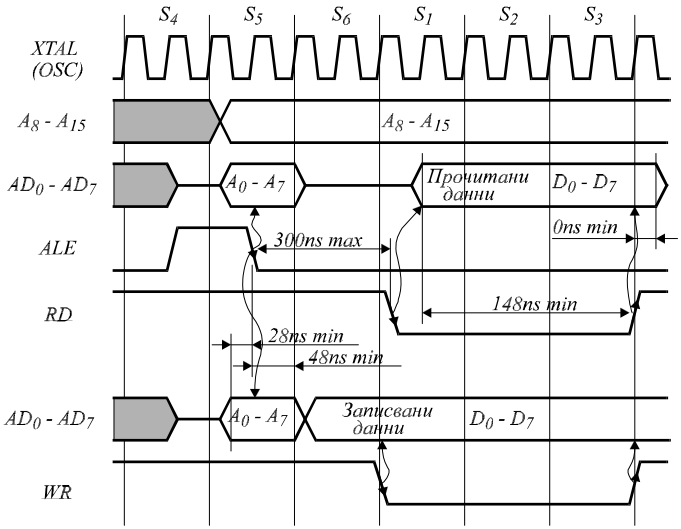
Фиг. 2.5. Обмен на данни в немултиплексирана системна шина – времедиаграми на циклите четене и запис на микропроцесора MC6800.

Типичен пример за обмен на данни по немултиплексирана системна шина е показан на фиг. 2.5, където са дадени конкретните времедиаграми на операциите четене и запис на микропроцесора MC6800. Микропроцесорът ползува две външни тактови поредици – Φ_1 и Φ_2 (с номинална честота 1 *MHz*), като обръщението към паметта трае един цикъл на тактовите поредици. Управлението на обмена се извършва с една линия $R / \overline{W}$, указваща посоката, и с придружаващия я сигнал *E*, указващ разрешение на магистралата за данни. Макар че микропроцесорът MC6800 притежава единно адресно поле, той включва в управляващата магистрала сигнал *VMA* за указване на валиден адрес за паметта.

При операцията четене, в началото на цикъла, микропроцесорът издава адреса, към който се обръща. Заедно с него се формира сигналът *VMA*, указващ действителен адрес на паметта, и $R / \overline{W}$, указващ режим на четене. Всичко това се извършва не по-късно от 300 *ns* след началото на микропроцесорния цикъл. По време на сигнала *E*, указващ разрешение на данните, микропроцесорът пое-

ма данните и ако последните са се установили до 100 *ns* преди края на цикъла, те ще бъдат правилно прочетени. Адресите и управляващите сигнали остават валидни поне още 10 *ns* след микропроцесорния цикъл.

По подобен начин протича операцията запис в микропроцесорната система. Тя започва както операцията четене, с тази разлика, че сигналът $R / \overline{W}$ указва режим на запис. По времето на сигнала *E* микропроцесорът разрешава изходните си буфери за данни и издава данните за запис, които се установяват на магистралата не по-късно от 225 *ns* след началото на *E*. Както и при четенето, адресите, управляващите сигнали, а сега и данните остават валидни поне още 10 *ns* след микропроцесорния цикъл.



Фиг. 2.6. Обмен на данни в мултиплексирана сестемна шина – времедиаграми на циклите четене и запис на микропроконтролера i8031.

Типичен пример за обмен на данни по мултиплексирана системна шина е показан на фиг. 2.6, където са дадени конкретните времедиаграми на операциите четене и запис в данновата памет на едночиповия микрокомпютър i8031 на фирмата Intel. Той ползува един тактов сигнал *OSC* (номинално 12 *MHz*), като един цикъл на обръщение към данновата памет трае 12 такта. Както при всички процесори на Intel, i8031 ползува два управляващи сигнала $\overline{RD}$ и $\overline{WR}$ за указване на операциите четене и запис и понеже те се използуват само при обръщение към данновата памет, друг управляващ сигнал при това обръщение не е предвиден. Мултиплексирана е информационната магистрала, която е 8-разредна, с младшите 8 линии на адресната магистрала.

В началото на цикъла четене от данновата памет се изработва сигнал *ALE*. Не по-късно от 28 *ns* преди спадащия фронт на *ALE* се извеждат старшите адресни линии, а по мултиплексираната *AD* магистрала се извеждат младшите адресни линии. Последните остават валидни поне още 48 *ns* след спадащия фронт на *ALE* (фронтът, който стробира младшите адреси в демултиплексиращия тригер). До 300 *ns* след *ALE* се активира сигналът $\overline{RD}$, чието активно състояние продължава до края на цикъла – максимално до 123 *ns* преди следващия *ALE* сигнал. По време на нарастващия фронт на $\overline{RD}$ едночиповият микрокомпютър стробира в себе си прочетените данни, които е трябвало да се появят на мултиплексираната *AD* магистрала поне 148 *ns* преди дезактивирането на $\overline{RD}$.

Времедиаграмите при цикъл запис в данновата памет са същите, както и при четене, с тази разлика, че се активира сигналът $\overline{WR}$, указващ операция запис и едночиповият микрокомпютър извежда данните за запис по мултиплексираната магистрала преди активирането на $\overline{WR}$ – веднага след младшия адрес.

2.5. Микроконтролер с MC6800

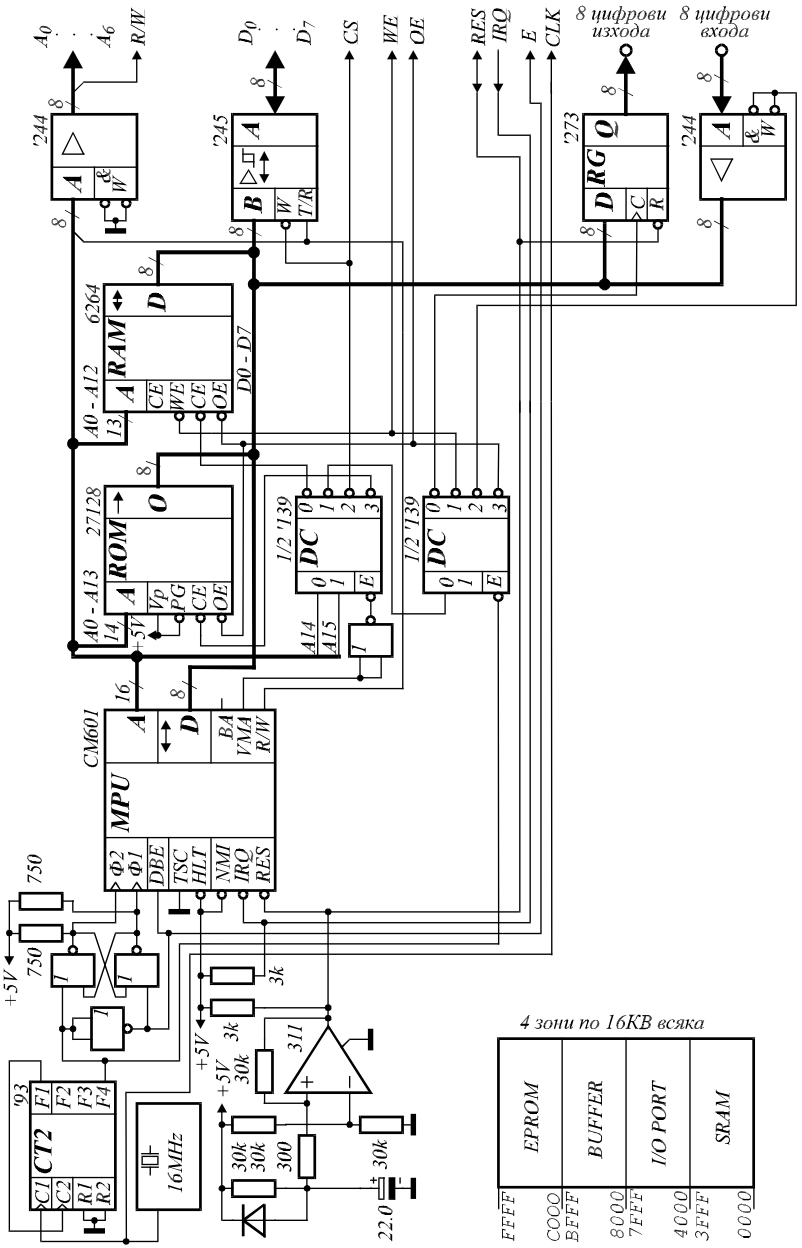
На фиг. 2.7 е показана схема на микропроцесорна система с MC6800, изградена на буферирана системна платка. Микропроцесорната система съдържа: тактов генератор, схема за автоматично начално установяване, постоянна и оперативна памет, паралелни портове за вход и изход, буфери за системната шина и допълнителна помощна логика.

Тактовият генератор е изграден с използването на синхронен брояч ‘161 като делител на 16. Основната честота 16 *MHz* и системният такт *E* (1 *MHz*) са изведени извън платката за разширяване на системата. Използването на синхронен брояч за делител е по-подходящо, тъй като тези две честоти остават в синхрон и от тях по-лесно могат да се синтезират допълнителни сигнали.

В схемата за начално установяване е използуван компараторът LM311, притежаващ изходен транзистор с отворен колектор. В неговия колектор се обединява $\overline{RESET}$ сигнал от системната шина по схема “жично ИЛИ”.

Паметта на микропроцесорната система съдържа 8 KB SRAM 6264 и 16 KB EPROM 27128. В системата са включени 8-разредни входен и изходен порта, изградени съответно с буфер и паралелен регистър. За избор и на двата порта е използуван един и същ сигнал $\overline{CS}$, така че те отговарят на един и същ адрес при непълно дешифриране, но непосредственият избор на портовете е разделен според операцията четене или запис.

В системата е приложен първичен дешифратор, изграден с интегралната схема 1/2 ‘139, който разделя адресното поле на 4 зони с обем по 16 KB. Втората половина на ИС ‘139 е използувана за синтезиране на сигнали за четене от паметта, за запис в паметта, за четене от паралелния порт и за запис в паралелния порт.



Фиг. 2.7. Микроконтролер с MC6800.

Три от зоните се използуват в основната платка на микропроцесорната система – за адресиране на оперативната памет, за адресиране на паралелните портове и за адресиране на постоянната памет. Една от зоните е предоставена за избор на модули извън основната платка. Само когато се активира сигналът за избор на тази зона, се разрешава работата на буфера за информационната магистрала, с което се избягва конфликтът на шината.

От адресните сигнали към буферираната системна шина са подадени само младшите 7, като по този начин достъпните области, с които могат да разполагат външните модули, са съкратени до обем от по 128 B. За целите на микроконтролерите това е напълно достатъчно.

2.6. Разширяване на интегралния микроконтролер i8031

Изграждането на разширена микропроцесорна система с i8031 е показано на фиг. 2.8. Забраната на вътрешната програмна памет и освобождаването на това пространство за външната програмна памет се извършва чрез подаване на *0* на входа $\overline{EA}$. Необходимо е това да се направи при разширяване на системата, за да се адресират векторите във външната програмна памет.

Вградената схема на тактов генератор в i8031 е разчетена за работа с кварцов резонатор в диапазона от 3,5 до 12 *MHz*. Схемата на автоматично начално установяване трябва да осигури високо ниво на входа *RST* в продължение на не по-малко от 24 периода на *OSC*. Микроконтролерът i8031 има режим на съхраняване на записаната във вътрешната му даннова памет информация при изключено захранване. За осигуряване на такъв режим е необходимо на входа *RTS* да се подава акумулаторно захранване +5 V, докато основното захранване е изключено.

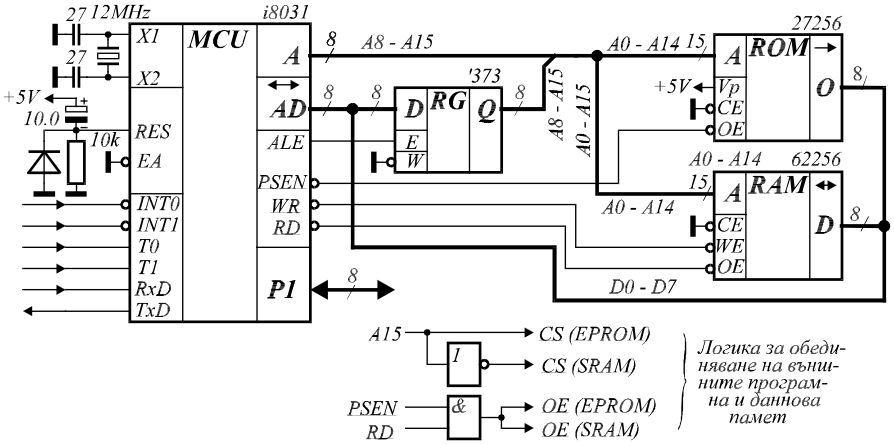
Показаната микропроцесорна система съдържа допълнителен паралелен регистър ‘373 (от типа “прозрачен фиксатор”), който извършва разделянето на младшата част на адреса от данните под управлението на спадащия фронт на сигнала *ALE*. Сигналите $\overline{PSEN}$, $\overline{RD}$ и $\overline{WR}$ определят типа на достъпа. При четене на програмната памет се активира сигналът $\overline{PSEN}$ и данните се стробират в микроконтролера по нарастващия (отминаващия) фронт на $\overline{PSEN}$. Цикъл на външно обръщение към програмната памет се извършва винаги когато генерираният адрес е извън пределите на вътрешната програмна памет или при $\overline{EA} = 0$. При четене на външната даннова памет се активира сигналът $\overline{RD}$, като данните се стробират в микроконтролера по нарастващия (отминаващия) фронт на $\overline{RD}$. При запис във външната даннова памет се активира сигналът $\overline{WR}$, като данните са валидни през цялото време, докато $\overline{WR}$ е с ниво *0*.

Изходите на портовете *P1* ÷ *P3* имат товароспособност 1 нормален TTL товар, а товароспособността на порт *P0* и на сигнала $\overline{PSEN}$ е 2 нормални TTL товара. В разширен режим, когато порт *P0* извежда *AD* магистралата, не е необхо-

димо поставянето на външни товарни резистори в него.

В пространството на външната даннова памет може да се включи оперативна памет с обем до 64 KB. В примера е показано включването на SRAM 62256 с обем 32 KB.

В пространството на външната програмна памет може да се включи постоянна памет с обем до 64 KB. В примера е показано включването на EPROM 27256 с обем 32 KB.



Фиг. 2.8. Разширена микропроцесорна система с i8031.

За осигуряването на максимално бързодействие SRAM и EPROM са постоянно избрани (на входовете им $\overline{CS}$ са подадени *0*), а изходните буфери на EPROM се отварят при активиран сигнал $\overline{PSEN}$, отговарящ за операцията четене от програмната памет. Записът и четенето от SRAM се определят от сигналите $\overline{RD}$ и $\overline{WR}$.

Понякога се налага съвместяването на външната програмна и даннова памет, най-често когато при настройка на потребителското програмно осигуряване то трябва да бъде зареждано в RAM. Това може да стане с допълнителна логика. Старшата адресна линия *A15* разделя обединеното външно пространство на две части, като в долната половина се разполага EPROM, а в горната – SRAM (EPROM се избира при *A15 = 0*, а SRAM – при *A15 = 1*). Четенето от общото пространство става чрез обединяване на сигналите $\overline{PSEN}$ и $\overline{RD}$.

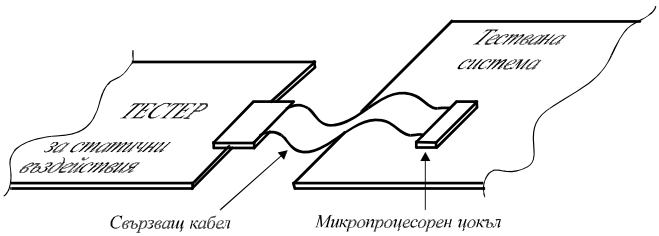
3. ДИАГНОСТИКА СЪС СТИМУЛИРАЩИ ВЪЗДЕЙСТВИЯ

Един от основните методи при диагностиката на произволна система се състои в това да се подават стимулиращи сигнали (въздействия) и да се наблюдава реакцията на системата, за да се установи дали тя функционира правилно или неправилно.

При диагностиката с микропроцесорни въздействия на тестваната система се подават стимулиращи сигнали през микропроцесорния й цокъл. Тези въздействия в по-малка или в по-голяма степен имитират поведението на реалния микропроцесор. Голяма част от реакциите на микропроцесорната система се получават също от микропроцесорния й цокъл.

3.1. Метод на статичните въздействия

Работата на микропроцесорните системи има динамичен характер, като сигналите непрекъснато се променят и най-често не може да се определи някаква периодичност. Тази непрекъсната промяна силно затруднява тяхното проследяване. Методът на статичните въздействия има за цел да елиминира фактора "време", като даде възможност за неограничено постоянно действие на сигналите и откриване на съществуващи неизправности в статичен режим.

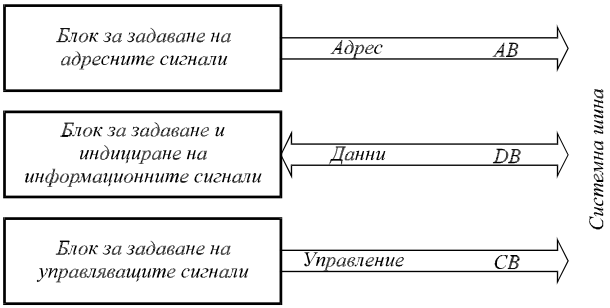


Фиг. 3.1. Включване на тестер за статични въздействия към тествана микропроцесорна система.

Всяко едно електронно устройство притежава определена скорост на работа (бързодействие), имаща горна и долна граница. Превишаването от страна на потребителя на горната граница обикновено довежда до пълен отказ на цялото устройство. Нарушаването на долната граница в повечето случаи не влияе върху работоспособността на устройството или на голяма част от блоковете му. Възможно е скоростта на работа на една микропроцесорна система да бъде намалена и практически сведена до нула. Това дава основание да се изследва последователността на събитията в една микропроцесорна система чрез заместване на микропроцесорните сигнали със статични. Целта е да се установи, че последователността на събитията в системата е правилна.

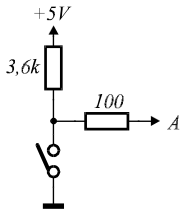
При метода на статичните въздействия микропроцесорът на проверяваната система се изважда и чрез съединител в неговия цокъл се включва кабел, свързващ проверяваната система с тестер за статични въздействия. Свързването е показано на фиг. 3.1.

Тестерът за статични въздействия трябва да осигури всички сигнали, необходими за изпълнение на операциите "четене" и "запис" в тестваната микропроцесорна система. Формират се въздействия за адресната, за информационната и за управляващата магистрала, които през микропроцесорния цокъл се подават към системната шина на тестваната система. Основните блокове на тестер за статични сигнали са показани на фиг. 3.2.



Фиг. 3.2. Блокова схема на тестер за статични въздействия.

Адресната магистрала е изходяща за тестера и входяща за тестваната система. Тъй като управлението е статично, сигналите могат да бъдат генерирани с помощта на механични превключватели. В едното положение на превключвателя съответният сигнал е логическа *1*, а в другото – логическа *0*.



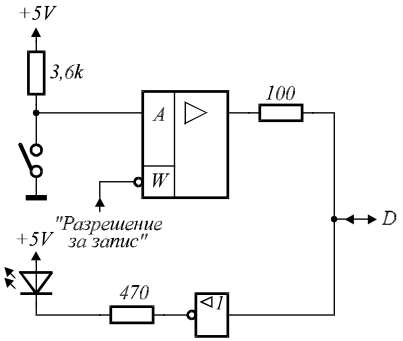
Фиг. 3.3. Формиране на адресна линия в тестер за статични въздействия.

На фиг. 3.3 е показан начин на формиране на една адресна линия за осигуряване на статично въздействие. Блокът за задаване на адресните сигнали трябва да съдържа толкова формирователя, колкото е разредността на адресната магистрала. Повечето от 8-битовите микропроцесори притежават 16-разредна адресна магистрала и за тях ще са необходими 16 такива линии.

Блокът за задаване и индициране на информационните сигнали изпълнява две функции – генерира информационни сигнали и осигурява индикацията им. Тези функции са аналогични на

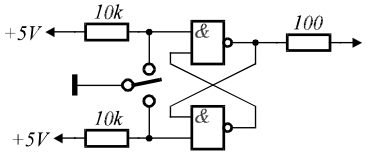
функциите запис и четене, изпълнявани от микропроцесора.

Примерна схема за статично задаване и индициране на една информационна линия в тестера е показана на фиг. 3.4. Информацията за запис се формира от механичен превключвател, след което преминава през буфер с високоимпедансно състояние, управляван от сигнал "разрешение за запис". Буферът е забранен по време на четене на данните. Прочитаната информация се приема от буфер с високо входно съпротивление и се визуализира със светодиод. Тестерът трябва да притежава толкова вериги за задаване и индициране на информационни сигнали, колкото е разредността на микропроцесорната система. За 8-битовите микропроцесори веригите трябва да бъдат 8, а за 16-битовите – 16.



Фиг. 3.4. Задаване и индициране на информационна линия в тестер за статични въздействия.

С най-голяма индивидуалност при различните микропроцесори се характеризират сигналите за управляващата магистрала. При всички положения тестерът трябва да бъде съобразен с тези индивидуалности. Някои от управляващите сигнали трябва да бъдат синхронизирани с тактов сигнал на микропроцесора. По принцип управляващите сигнали са изходящи от микропроцесора и съответно от тестера. Те се формират по подобие на формирането на адресните сигнали. Някои микропроцесори притежават и допълнителни входящи управляващи сигнали, за които трябва да бъде осигурено индицирането им в тестера, подобно на индицирането на информационните сигнали.



Фиг. 3.5. Задаване на информационен сигнал в тестер за статични въздействия с елиминиране на смущения от механични вибрации.

Когато изходящите управляващи сигнали се формират от механични превключватели, трябва да се вземат мерки за елиминиране на смущенията от механичните вибрации на контактите. Например на фиг. 3.5 е показано формирането на управляващ сигнал с механичен превключвател с последващо отстраняване на смущенията чрез *SR*-тригер.

Всички изходящи от тестера сигнали се буферират (най-често с резистори със стойност около 100 Ω), за да се имитира по-пълно ограничената товароспособност на реалния микропроцесор.

По отношение на микропроцесорните операции "запис" и "четене" управляващите сигнали се делят на основни и второстепенни. Основни са онези управляващи сигнали, които участвуват непосредствено в операциите “запис” и “четене” и променят състоянието си през тяхното протичане. Управлението на тези сигнали трябва да бъде осигурено от страна на тестера. Второстепенните управляващи сигнали не участват непосредствено в операциите "запис" и "четене", но те имат определено статично състояние за правилното провеждане на операциите. Например при статичен тестер за микропроцесорни системи с MC6800 основни управляващи сигнали са *VMA* (действителен адрес на паметта) и $R / \overline{W}$ (четене/запис), а второстепенен управляващ сигнал е *BA* (достъпни магистрали), който при извършване на операциите трябва да е в състояние на логическа *0*. Второстепенните управляващи сигнали получават твърдо логическо ниво (неуправляемо от тестера) според спецификата на микропроцесора, който се замества. Много от микропроцесорите и едночиповите микрокомпютри не притежават второстепенни управляващи сигнали (например i8031).

3.2. Тестер за статични въздействия

Чрез тестера за статични въздействия се извършва проверка на правилността на действието на операциите "запис" и "четене" на микропроцесора. Това са основните операции, които той извършва в системата, и ако те са правилни, с голяма вероятност може да се счита, че системата е изправна. Едновременно с тях методът на статичните въздействия служи за проверка и на адресирането в тестваната система.

Проверката на системата чрез тестер за статични въздействия се подчинява на определени правила. Започва се с операция “запис”, като се спазва следната последователност от действия:

1. Първоначално управляващите сигнали се поставят в неактивно състояние.
2. С адресните превключватели на тестера се задава адресът за запис.
3. Чрез превключвателите за информационната магистрала се задават избраните данни за запис.
4. Управляващият сигнал, указващ операцията “запис”, се поставя в активно състояние (активна фаза на операцията “запис”).
5. Управляващите сигнали се връщат в неактивно състояние.

Следващата стъпка е проверка на операцията “четене”, която се извършва по подобен начин със следната последователност от действия:

1. Първоначално управляващите сигнали се поставят в неактивно състояние.
2. С адресните превключватели на тестера се задава адресът, от който ще се чете.
3. Управляващият сигнал, указващ операцията “четене”, се поставя в активно състояние (активна фаза на операцията “четене”).
4. Управляващите сигнали се връщат в неактивно състояние.

След тази процедура индикатори за данни на тестера трябва да показват прочетените данни.

Ако проверката на резултатите от операциите “запис” и “четене” не дадат правилен резултат, след всяко действие съответната операция може да се спре в активната си фаза и да се анализира състоянието на системата. Най-просто това се извършва с логически пробник.

На фиг. 3.6 е показана принципна схема на тестер за статични въздействия за проверка на микропроцесорни системи, изградени с микропроцесора MC6800. Схемата се подчинява на изложените общи съображения. Формираните управляващи сигнали *VMA* и $R / \overline{W}$ допълнително са синхронизирани със спадащия фронт на сигнала *DBE*, което осигурява промяната им само в началото на микропроцесорния цикъл. Приеманите данни също се синхронизират със спадащия фронт на *DBE*. Вместо обикновени светодиоди за прочитане на информационната магистрала в показаната схема са използувани седемсегментни индикатори с цел изобразяване в шестнадесетичен код.

Информацията от магистралата за данни постъпва за запомняне в *D*-тригери с високо входно съпротивление ‘HCT374, по спадащия фронт на сигнала *DBE* (Разрешение на шини данни) – т.е. тогава, когато според времедиаграмите на работа на микропроцесора свършва цикълът на обмен.

От изходите на D-тригерите данните се разделят на две тетради. Всяка от тях се третира като число в шестнадесетична бройна система и се преобразува в код за седемсегментен индикатор от кодов преобразувател. Избран е кодовият преобразувател F9368 на фирмата Fairchild, тъй като той осигурява на изхода си код за индициране на всички шестнадесетични цифри – `0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b, C, d, E, F`.

Вместо единични превключватели за задаване на адресните сигнали и на данните за запис е по-удобно да се използуват превключватели, които директно кодират по тетради информацията в шестнадесетичен код.

За да се извърши операцията "запис" в паметта чрез показания тестер за статични въздействия, е необходимо да се изпълни следната последователност от действия:

1. Първоначално сигналът *VMA* се поставя в логическа *0*, което определя невалиден адрес на паметта.

2. С адресните превключватели на тестера се задава желаният адрес.

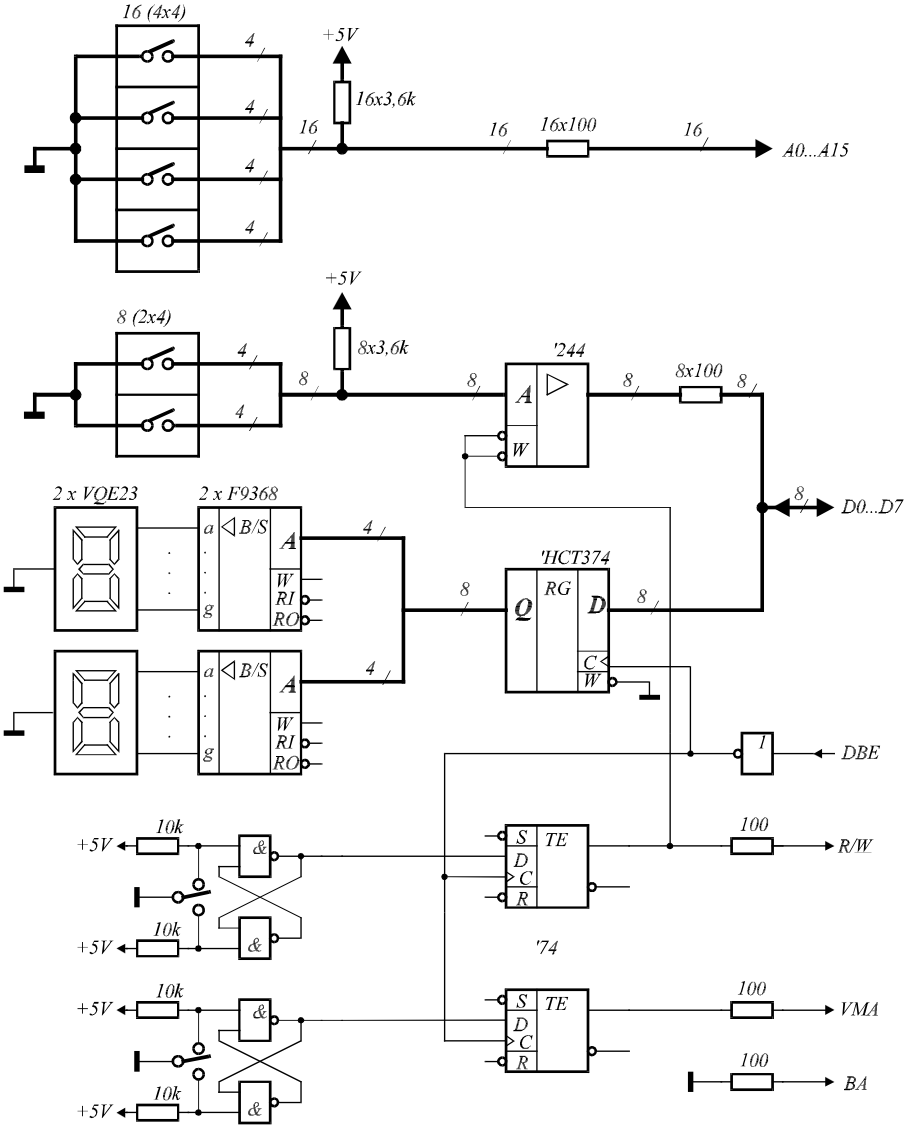
3. Сигналът $R / \overline{W}$ се поставя в положение на логическа *0*, указвайки, че ще се извършва операция “запис”.

4. Чрез превключвателите за информационната магистрала се задават избраните данни за запис.

5. Сигналът *VMA* се поставя в логическа *1*, след което отново се връща в състояние на логическа *0*.

Сигналът *DBE* не се формира от тестера, а непрекъснато се генерира от так-

товия генератор на микропроцесора. Затова при единична операция с тестера се извършват толкова записа, колкото цикъла на сигнала *DBE* протекат по време на валидния адрес на паметта.



Фиг. 3.6. Тестер за статични въздействия за микропроцесорни системи с MC6800.

По подобен начин се извършва и операцията "четене" от паметта с тестера за статични въздействия. Спазва се следната последователност от действия:

1. Първоначално сигналът *VMA* се поставя в логическа *0*.

2. Задава се адресът на паметта, от който ще се чете.

3. Сигналът $R / \overline{W}$ се поставя в логическа *1*, за да укаже на системата, че операцията е четене.

4. Сигналът *VMA* се поставя в логическа *1*, след което отново се връща в състояние на логическа *0*.

След тази процедура светодиодните индикатори за данни на тестера трябва да показват прочетените данни. По времето когато *VMA* е логическа *1*, се извършва по едно четене на всеки цикъл на сигнала *DBE*.

Подобен тестер за статични въздействия за микропроцесорни системи с i8031 е показан на фиг. 3.7. Тъй като i8031 притежава мултиплексирана магистрала, младшите адреси от *A0* до *A7* се генерират от едни и същи превключватели с информационните сигнали *D0* до *D7*. Разлика има и в броя на управляващите сигнали.

Тъй като i8031 притежава отделна даннова и програмна памет, е необходимо допълнително да се провери операцията "четене от програмната памет". Тя се изпълнява в следната последователност:

1. Първоначално сигналите $\overline{PSEN}$, $\overline{RD}$ и $\overline{WR}$ се поставят в пасивно състояние – логическа *1*, а *ALE* – в *0*.

2. С адресните превключватели се задава адресът на програмната памет, от който ще се чете.

3. Сигналът *ALE* се активира в *1*, след което се връща в *0*.

4 Сигналът PSEN се поставя в логическа *0*, след което отново се връща в състояние на логическа *1*. По времето когато $\overline{PSEN}$ е в активно състояние (логическа *0*), индикаторите за данни трябва да показват прочетената информация от зададения адрес.

За да се извърши операцията "запис в данновата памет", е необходимо да се изпълни следната последователност от действия:

1. Първоначално сигналите $\overline{PSEN}$, $\overline{RD}$ и $\overline{WR}$ се поставят в пасивно състояние – логическа *1*, а *ALE* – в *0*.

2. С адресните превключватели на тестера се задава адресът на данновата памет, в който ще се записва.

3. Сигналът *ALE* се активира в *1*, след което се връща в *0*.

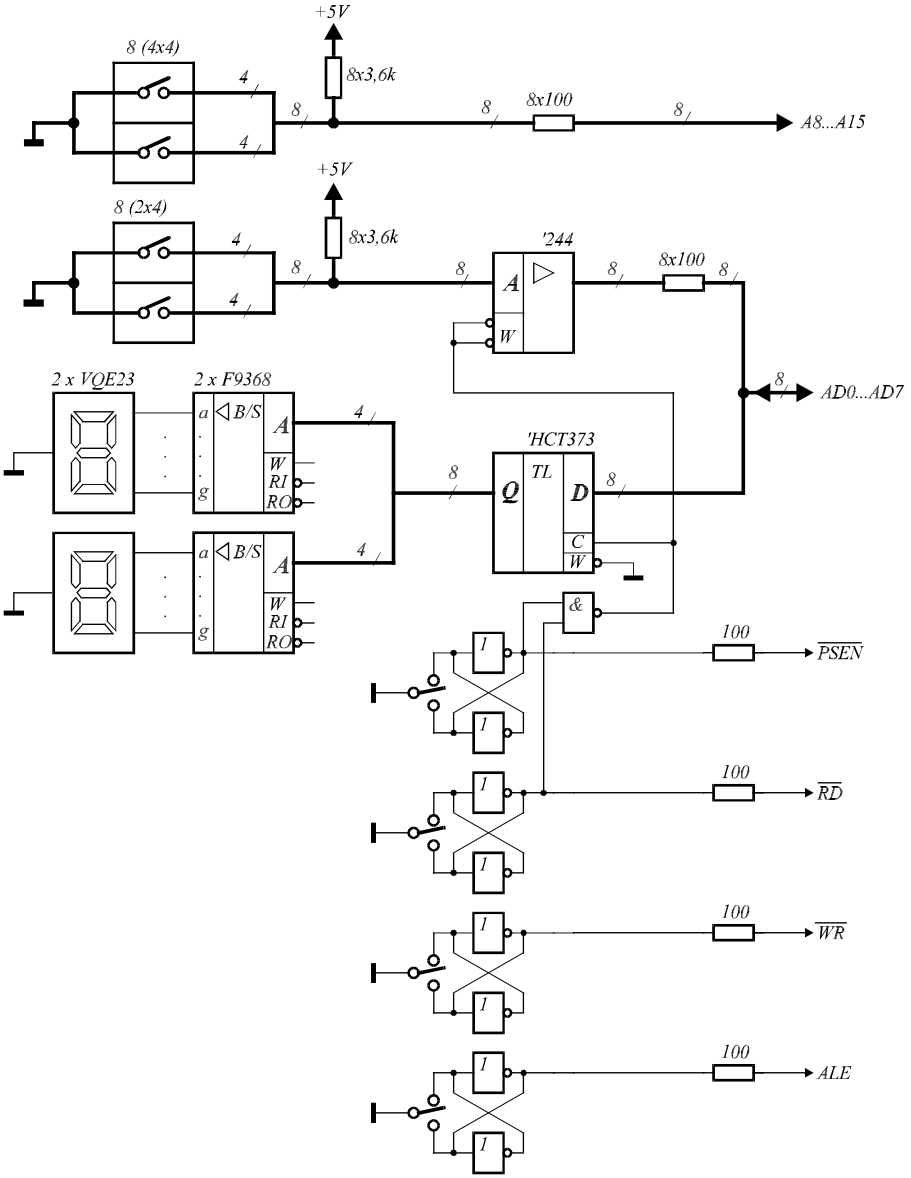
4. Чрез превключвателите от *AD0* до *AD7* се задават данните за запис.

5 Сигналът $\overline{WR}$ се поставя в логическа *0*, след което отново се връща в състояние на логическа *1*.

Проверката на операцията "четене от данновата памет" се осъществява в следната последователност:

1. Първоначално сигналите $\overline{PSEN}$, $\overline{RD}$ и $\overline{WR}$ се поставят в пасивно състоя-

ние – логическа *1*, а *ALE* – в *0*.



Фиг. 3.7. Тестер за статични въздействия за микропроцесорни системи с i8031.

2. С адресните превключватели се задава адресът на програмната памет, от който ще се чете.

3. Сигналът *ALE* се активира в *1*, след което се връща в 0.

4 Сигналът $\overline{RD}$ се поставя в логическа *0*, след което отново се връща в състояние на логическа *1*. По времето когато $\overline{RD}$ е в активно състояние (логическа *0*), индикаторите за данни трябва да показват прочетената информация от зададения адрес.

Методът на диагностика чрез статични въздействия е приложим в много случаи, когато други методи не дават добри резултати. Той може да се използува дори когато нито един блок на проверяваното устройство не е работоспособен. Недостатък на метода е, че някои неизправности могат да бъдат неоткрити. Ако една микропроцесорна система работи правилно в динамичен режим, тя ще бъде определена като изправна чрез метода на статичните въздействия. Обратното твърдение не винаги е вярно. Ако една микропроцесорна система е определена като неизправна по метода на статичните въздействия, тя със сигурност няма да работи и в динамичен режим.

Специално разглеждане изисква тестерът за статични въздействия, заместващ микропроцесора i8088 (i8086). Този микропроцесор освен че притежава мултиплексирана магистрала за младшите адреси и данните, може да работи и в два режима – минимален и максимален. В минимален режим на работа микропроцесорът сам изработва управляващите сигнали. Основни управляващи сигнали за циклите “запис” и “четене” са:

– $IO/\overline{M}$ (Избор на входно-изходно устройство или памет);

– $DT/\overline{R}$ (Данни за изпращане или получаване);

– *ALE* (Строб за адреса) с пасивно състояние *0*;

– $\overline{DEN}$ (Разрешение за данните) с пасивно състояние *1*;

– $\overline{RD}$ (Четене) с пасивно състояние *1*;

– $\overline{WR}$ (Запис) с пасивно състояние *1*. Допълнителни управляващи сигнали за циклите “запис” и “четене” са:

– $\overline{INTA}$ (Потвърждение на прекъсването) със статично пасивно състояние *1*;

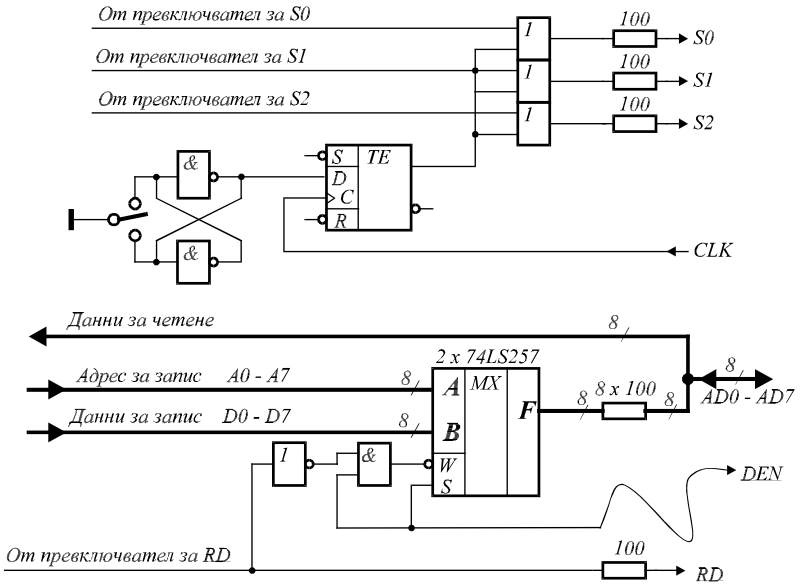
– *HLDA* (Потвърждение за заявката за магистралите) със статично пасивно състояние 0.

В тестера за статични въздействия, заместващ i8088 в минимален режим на работа, няма нищо особено и той е подобен на този за i8031.

По този начин се изграждат тестери за статични въздействия, заместващи микропроцесори с мултиплексирана адресна и информационна магистрала, като i8085, MC146805E2 и др.

При работа на микропроцесора i8088 в максимален режим той изработва управляващи сигнали *S0*, *S1* и *S2*, които в кодиран вид съдържат информация за предстоящата операция. Тези сигнали се декодират от системен контролер, който изработва необходимите конкретни управляващи сигнали за микропроце-

сорната система. За системния контролер пасивно състояние е когато и трите сигнала *S0*, *S1* и *S2* са в логическа *1*. Всяка промяна се възприема като начало на микропроцесорен цикъл. Тъй като операторът не може едновременно да зададе необходимите състояния на тези линии, за да не се допусне някое междинно преходно състояние да се възприеме като начало на операция, е необходимо състоянията на линиите *S0*, *S1* и *S2* да се формират предварително и едновременно да се подадат навън, и то синхронизирано с такта *CLK*. Това става под управлението на допълнителен превключвател "старт-стоп". Външният системен контролер ще изработи необходимите управляващи сигнали, включително и демултиплексиращите сигнали *ALE* и $\overline{DEN}$. Операторът няма възможност бързо да смени адресите с данни по мултиплексираната магистрала. Затова в тестера се извършва разделно задаване на адресните и информационните сигнали, а след това през мултиплексори те се подават към общата магистрала. Управлението на мултиплексорите се извършва от сигнала $\overline{DEN}$. Тъй като в максимален режим микропроцесорът i8088 не генерира такъв сигнал (следователно не се изработва и от тестера), той се взима с допълнителна единична сонда от микропроцесорната система. Възможен е и вариант, при който в тестера се поставя системен контролер специално за генерирането на сигнала $\overline{DEN}$.



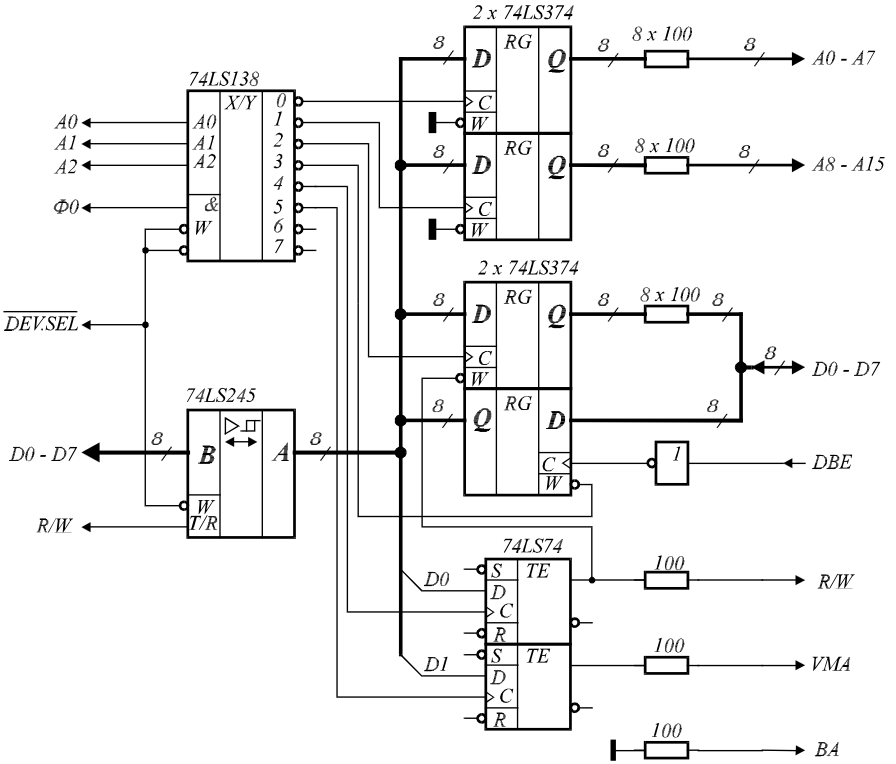
Фиг. 3.8. Тестер за статични въздействия за микропроцесор i8088, работещ в разширен режим.

На фиг. 3.8 е показана схема за формиране на управляващи сигнали и сиг-

нали за мултиплексираната адресна и информационна магистрала за тестер на статични въздействия, заместващ микропроцесора i8088 в максимален режим на работа.

3.3. Диагностика с псевдодинамични въздействия

Една разновидност на метода на статичните въздействия представлява методът на псевдодинамичните въздействия. При него достъпът до проверяваната система също се извършва през нейния микропроцесорен цокъл. Чрез кабел тя се свързва с тестер за псевдодинамични въздействия, в който, за разлика от тестера за статични въздействия, всички сигнали към тестваната система се формират не ръчно от оператора, а автоматично от електронно управляващо устройство.



Фиг. 3.9. Тестер за псевдодинамични въздействия за MC6800.

Най-често тестерът за псевдодинамични въздействия се оформя като пери-

ферен модул за персонален компютър. Под неговото програмно управление се задава последователността на сигналите за извършване на операциите "четене" и "запис" в проверяваната система. Тъй като тестерът не може да възпроизведе реалната динамика на процесорните сигнали, методът на тестване се нарича псевдодинамичен.

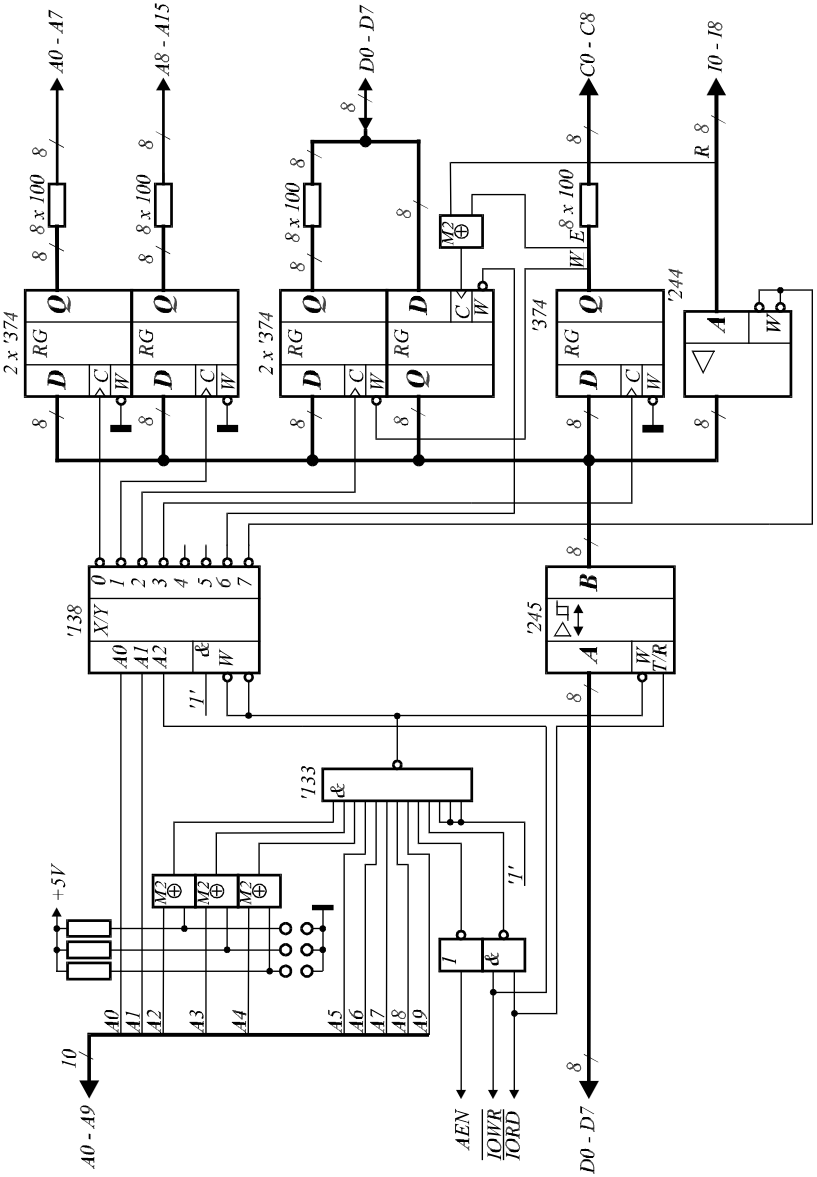
Както проблемите, така и възможностите на метода на псевдодинамичните въздействия са същите, както и при метода на статичните въздействия. На фиг. 3.9. е показана схема на тестер за псевдодинамични въздействия за микропроцесорна система с MC6800, изграден като периферен модул за персоналния компютър Apple II.

Модулът заема изцяло шестнадесетте адреса, определени за всеки слот и дефинирани от сигнала $\overline{DEVICE\_SELECT}$. Допълнително чрез дешифратора 74LS138, разрешаван само по време на *Ф0*, се изработват сигналите за избор на вътрешните регистри в модула. Два 8-битови регистъра 74LS374 служат за извеждане съвместно на младшите и на старшите адреси. Други два регистъра са свързани противопосочно, като единият служи за извеждане на данните за запис, а другият – за съхраняване на прочитаните данни, докато компютърът ги прочете. Такова междинно съхраняване на данните се налага, тъй като по принцип тестваната и тестващата система работят асинхронно една спрямо друга.

На фиг. 3.10 е показан универсален тестер за псевдодинамични въздействия за микропроцесорни системи, изградени с 8-битови микропроцесори и микрокомпютри. Тестерът е изграден като периферен модул за персонален компютър тип IBM PC. Модулът заема 4 адреса във входно-изходното пространство на компютъра, като с помощта на превключватели може да бъде преадресиран в адресното пространство от `3E0` до `3FF`.

Първите два адреса са само за запис и чрез тях се извеждат 16 адресни сигнала. Третият адрес е достъпен за запис и четене и е предназначен за извеждане и въвеждане на 8 информационни сигнала. Четвъртият адрес също е достъпен за запис и четене. При запис на този адрес се извеждат до 8 управляващи сигнала, които се формират според процесора, който се имитира. Два от извежданите управляващи сигнала са върнати обратно в схемата на тестера. Единият (означен с $\overline{W}$) е предназначен с активно ниво *0* да дава "разрешение за запис" на изходния регистър за информационните сигнали, а вторият (означен с *E*) е предназначен да извършва запис на прочитаните данни във входния регистър за информационните сигнали. При четене от четвъртия адрес в тестера могат да бъдат възприети за индициране състоянията на 8 сигнала. Това могат да бъдат управляващи сигнали, сигнали за арбитриране на шините, за управление на прекъсванията и др. от тестваната система. Един от тези сигнали (означен с *R*) е предназначен също да извършва запис на прочитаните данни във входния регистър за информационните сигнали. Всъщност този запис може да се осъществява както под управление на тестера чрез сигнала *E*, така и под управление на

тестваната система чрез сигнала *R*.



Фиг. 3.10. Универсален тестер за статични въздействия за 8-битови системи.

Конфигурирането на тестера за работа с различни процесорни системи се извършва чрез смяна на интерфейсния кабел към микропроцесорния цокъл на тестваната система и чрез подходящо за конкретния случай програмно осигуряване.

Псевдодинамичният тестер може автоматично да изпълни и тества операциите "четене" и "запис". С цел улесняване на диагностиката задължително се предвижда възможност за спиране на операциите в активната им фаза и съответното продължаване по указание на оператора. Всичко това се извършва по програмен път.

Псевдодинамичните тестери обикновено се изграждат с възможност за тестване на системи с различни микропроцесори, като реконфигурирането става по апаратно-програмен път. Стремежът е операциите “четене” и “запис” да се извършват колкото се може по-бързо. Тестер, който може да извърши тези операции с динамиката на реалния микропроцесор, се нарича имитатор, а методът – метод на имитацията.

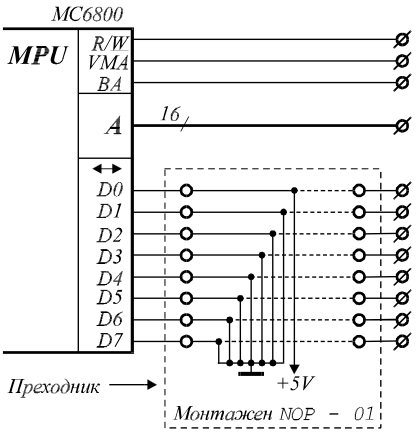
3.4. Броячен режим на микропроцесор

Методите на статичните и на псевдодинамичните въздействия, а както ще се види по-нататък и използването на микропроцесорен вътрешносхемен емулатор, тестват самата микропроцесорна система без микропроцесора, тъй като той се изважда и отстранява.

Прост, но ефективен способ, с помощта на който може да се провери един микропроцесор, а и не само той, е т.нар. броячен режим на микропроцесора или режим на свободно броене. При този метод информационните линии на микропроцесора се откачват от системата и на тях се подава кодът на празната операция NOP (фиг. 3.11). Това се осъществява, като между микропроцесора и цокъла му в системата се постави преходник, в който е извършено отделянето на информационните линии и на тях по монтажен начин е подадена операция NOP (01 за MC6800 и MC68HC11, 00 за Z80, A5 за i8031, EA за 6502 и т.н.). Всички други връзки на микропроцесора със системата се запазват.

След начално установяване микропроцесорът ще се обърне към паметта за първата команда и за такава ще прочете NOP. Той ще увеличи адреса си с 1 и отново ще прочете команда NOP и т.н. Прочитайки непрекъснато NOP микропроцесорът ще адресира последователно всички клетки от адресното си пространство и непрекъснато ще реализира операция четене.

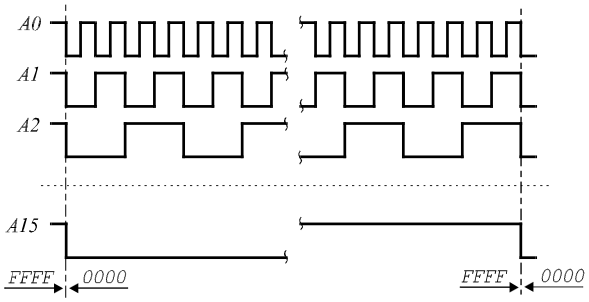
Например микропроцесорът MC6800 след $\overline{RESET}$ прочита векторите на начално установяване от адреси FFFE и FFFF. Там той намира 01 и 01. Съставя си от тях адрес 0101, който го интерпретира като начало на работна програма и прочита кода на първата команда 01 (NOP). Увеличава адреса си с 1 и от 0102 прочита следващата команда – пак 01. Това продължава докато стигне края на адресното пространство и от адрес FFFF прочете код на операция 01.



Фиг. 3.11. Броячен режим на микропроцесор MC6800.

В следващия момент микропроцесорът отново нараства адреса си с 1, неговият брояч на командите се нулира и адресът, от който той търси следващата инструкция, е вече `0000`. Следващият адрес е `0001` и т.н. Получава се непрекъснато циклично обхождане на адресното пространство.

При режима на свободното броене адресната магистрала на микропроцесора има поведението на 16-разреден двоичен брояч. Всяка адресна линия генерира импулси с коефициент на запълване 1/2. С увеличаване на номера на адресната линия, започвайки от *A0*, генерираната честота се намалява два пъти при всяка следваща линия, както е показано на фиг. 3.12. Тази характерна картина, наблюдавана с осцилоскоп, може да послужи за следното:



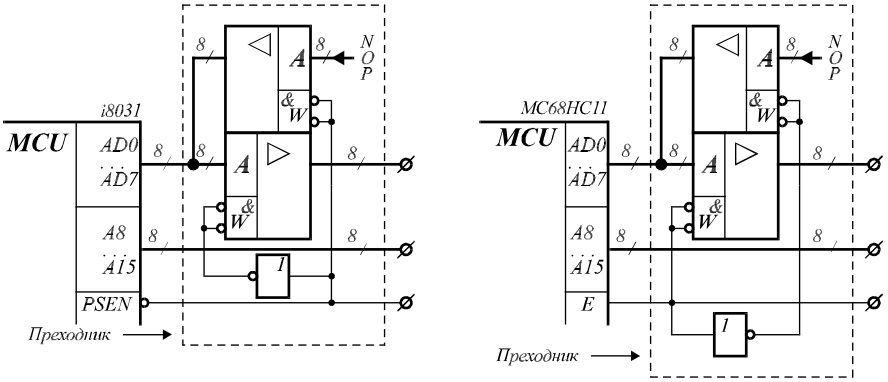
Фиг. 3.12. Поведение на адресните сигнали при броячен режим на микропроцесор.

1. С голяма вероятност може да се направи заключението, че микропроцесорът е работоспособен.

2. Свидетелствува за изправна адресна магистрала, когато картината се наблюдава при адресните входове на компонентите от микропроцесорната система. Всяка адресна линия може да се провери за прекъсване, за късо към маса или захранване и за късо с друг сигнал.

3. Служи за активиране на дешифратора на адресното поле на микропроцесорната система, което може да се използува за определяне чрез допълнителни

апаратни средства на неговата изправност.



Фиг. 3.13. Броячен режим на едночиповите микрокомпютри i8031 и MC68HC11.

При някои микропроцесори и микрокомпютри, имащи мултиплексирана адресна и информационна магистрала (например i8031, MC68HC11 и др.), се налага известно усложняване на схемата за непрекъснато подаване на операция `NOP`. Фиг. 3.13 илюстрира поставянето на микрокомпютрите i8031 и MC68HC11 в режим на свободно броене. Еднопосочен буфер обезпечава подаването на `NOP` по мултиплексираната магистрала към микрокомпютъра само когато тя се използува за получаване на данни (код на операцията).

4. ВЪТРЕШНОСХЕМЕН МИКРОПРОЦЕСОРЕН ЕМУЛАТОР

Емулирането е процес, при който една система се използува за копиране действията на друга система. В зависимост от степента на копиране съществуват различни нива на емулиране. Най-ниското ниво на емулиране е когато работната среда на една система се използува за програмно моделиране на работата на друга система, без двете системи да имат връзка помежду си. Най-високо ниво на емулиране има когато се изгради точен прототип (копие) на емулираната система и върху него се извършат експерименти за изучаването на системата.

Понятието *вътрешносхемна емулация* се използува за означаване на подмяната на определена схема със система, която копира поведението й. Буквалното значение на думата *емулация* или *емулиране* е подражаване. Това понятие е въведено от фирмата Intel, която първа е използувала в свои системи метода на вътрешносхемната емулация като работно средство за проектиране на микропроцесорни системи. Системата, в която е извършена подмяната, се нарича *емулирана система (прототип)*, а системата, която извършва подмяната – *емулираща система* или само *емулатор*.

В една микропроцесорна система могат да бъдат емулирани различни схеми. Когато се заменя и емулира памет, емулиращите системи носят наименованието съответно *ROM-емулатори* или *RAM-емулатори*.

Когато се извършва замяна и емулиране на микропроцесора, емулиращата система се нарича *вътрешносхемен микропроцесорен емулатор*. Микропроцесорът събира в себе си жизненоважните сигнали за системата, той има достъп до всички нейни блокове и възли, притежава цялата информация за поведението й. Затова чрез система, заместваща и копираща поведението на микропроцесора, може да се добие най-пълна представа за състоянието на емулираната система, за нейната работоспособност, както и да се извърши пълното й апаратно и програмно настройване, поправка и др.

4.1. Структура на микропроцесорен емулатор

Съвременните вътрешносхемни микропроцесорни емулатори представляват високоинтелигентни микропроцесорни системи. От появата си те търпят непрекъснато развитие в посока на усъвършенствуване и обогатяване на възможностите им. Стремежът е колкото се може по-пълно и по-точно да бъдат копирани програмните и апаратните възможности на заместваният микропроцесор. Емулираната система не трябва да “усети” подмяната на истинския микропроцесор с емулатор.

Подобно на методите на стимулиращите въздействия, микропроцесорът на проверяваната система се изважда и в неговия цокъл се включва вътрешнос-

хемният микропроцесорен емулатор. Той осигурява на проверяваната система всички необходими микропроцесорни сигнали, заедно с тяхната динамика, съхранявайки времевите им параметри. Създава се възможност за проверка на функциите на диагностицираното устройство в реално време. От гледна точка на диагностиката микропроцесорният емулатор изпълнява следните задачи:

– копира апаратните функции на микропроцесора (извършва обмен на информация, обработва прекъсванията и арбитрира магистралите);

– задава начални състояния на регистрите в микропроцесора и стартира потребителски програми;

– осигурява информация за хода на потребителската програма;

– управлява адресното пространство на микропроцесорната система;

– управлява ресурсите на потребителя и позволява емулирането и на някои други елементи от тестваната система – оперативна памет, постоянна памет, периферни устройства.

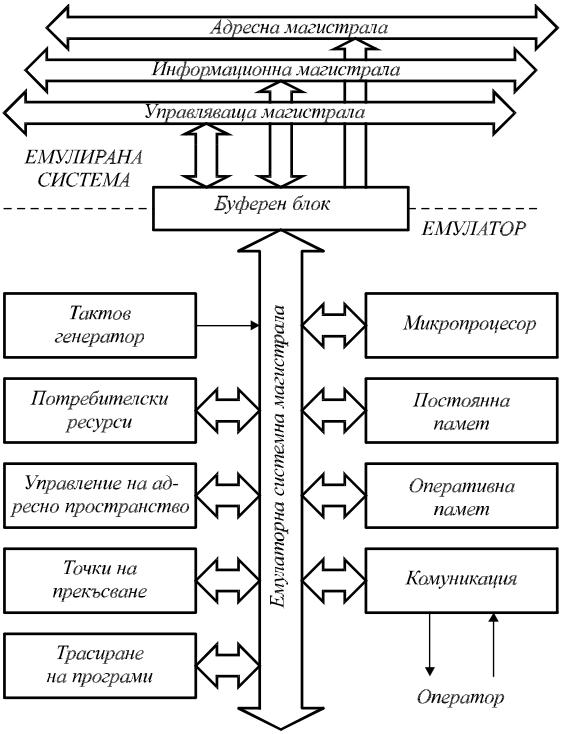
Тези задачи емулаторът изпълнява под управлението на намираща се в него програма, наричана *мониторна програма* или само *монитор*. На първо място, тази програма съдържа драйвер за входно-изходно устройство, чрез което операторът общува с емулатора. На второ място, тя притежава команден интерпретатор, позволяващ идентифицирането на командите, подадени от оператора, и тяхното изпълнение. На трето място, това са команди, позволяващи настройка и диагностика на системата *(debugger)*. Количеството команди, които може да възприема и изпълнява даден емулатор, както и удобната работа с тях, определят богатството на мониторната му програма. Елементарните команди, които трябва да изпълнява един микропроцесорен емулатор, са: обмен на информация с паметта и входно-изходните устройства, зареждане на потребителски програми и тяхното стартиране, управляемо прекъсване на потребителски програми и покомандното им изпълнение и др.

На фиг. 4.1 е показана общата блокова схема на вътрешносхемен микропроцесорен емулатор. През микропроцесорния цокъл на емулираната система се осигурява свързване на трите магистрали (адресна, информационна и управляваща) с тези на емулиращата система. Всички сигнали преминават през буферен блок, осигуряващ взаимодействието на двете системи. Чрез него се осъществява апаратно съгласуване, а също и електрическо развързване на емулиращата от проверяваната система. Последното е необходимо, за да се прекъсне разпространяването на неизправност от проверяваната система към емулатора и да се запази неговата работоспособност при всички случаи.

Обикновено, между емулатора и тестваната система, сигналите преминават през свързващ кабел с определена дължина. Ако дължината на кабела е значителна и за процесорните сигнали има опасност той да се прояви като дълга линия, неговото изпълнение е от проводници с разпределени параметри. В този случай и в двата му края се поставят буферни блокове за съгласуване на сигна-

лите с параметрите на дългата линия.

Микропроцесорът, постоянната памет, оперативната памет и входно-изходното устройство за връзка с оператора са онези атрибути, които оформят емулатора като самостоятелна микропроцесорна система.



Фиг. 4.1. Структура на вътрешносхемен микропроцесорен емулатор.

Управляващият микропроцесор в емулатора може да се изгради по един от следните три начина:

– чрез микропроцесора, за чието физическо копиране е предназначен емулаторът;

– чрез различен микропроцесор от този, който се емулира;

– чрез други схеми.

Първият начин е най-разпространен, тъй като при него се постига най-пълно съвместяване между емулатора и емулираната система. Той се прилага за почти всички микропроцесори, имащи изведени навън адресна, информационна и управляваща магистрала.

Вторият начин се използува при онези микропроцесори, върху които не може да се извършва пълен контрол отвън. Такива са например онези едночипови микрокомпютри, които нямат изведени навън адресни, управляващи и информационни сигнали и вътрешната им магистрала не е достъпна. За емулирането им се използуват други микропроцесори, близки по функции до тях. Някои фамилии едночипови компютри включват в състава си специални (диагностични) варианти с изведена навън микропроцесорна магистрала и тези варианти са подходящи за изграждане на емулатори.

Третият начин практически е загубил самостоятелно приложение при микропроцесорни системи. Той се използува в комбинация с другите два, там където се налага изграждане на устройства, дублиращи работата на част от микропроцесора или едночиповия микрокомпютър. Например при емулирането на MC68HC11 емулаторът може да се изгради със същия едночипов микрокомпютър, работещ в разширен режим. Тогава портовете B и C не могат да се използуват като такива, тъй като те се трансформират в системни линии. В този случай се налага външно изграждане на схеми, копиращи действията на тези портове. Такава е интегралната схема MC68HC24.

Постоянната памет съдържа мониторната програма на емулатора. Възможна е и конфигурация, при която мониторната програма се зарежда отвън в оперативната памет. Тогава постоянната памет съдържа зареждащата програма на монитора. Чрез комуникационен блок се осъществява връзка с оператора, а също така евентуално и с други компютърни системи.

Блокът на потребителските ресурси съдържа апаратни елементи от микропроцесорна система, които се предоставят на потребителя. Тези елементи се включват от потребителя на мястото на аналогични елементи от тестваната система. Така настройката на нейната работна програма може да започне преди да е изцяло апаратно завършена тестваната система. На първо място в тези ресурси се включва определено количество оперативна памет, в която потребителят зарежда работната програма на тестваната система и извършва нейното настройване. На второ място към ресурсите може да бъде прибавено и определено количество цокли за постоянна памет, в които потребителят може да постави запрограмирани вече настроени части от работната програма. По-старите емулатори притежават като потребителски ресурс и някакво количество стандартни интерфейсни схеми, като например PIA, ACIA, PTM и др. Но тъй като разнообразието на входно-изходни устройства е голямо и не може да се обхване само със стандартни интерфейсни схеми, все по-рядко такива елементи се включват в блока на потребителските ресурси. Управлението на блока на потребителските ресурси се извършва от оператора чрез декодера на емулатора и от блока за управление на адресната карта на паметта.

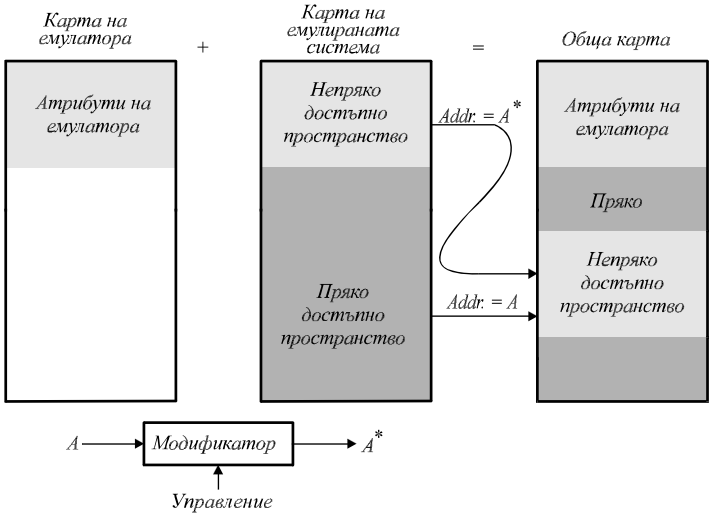
Тактовият генератор на емулатора осигурява необходимите синхронизиращи сигнали за работа на емулиращата микропроцесорна система. Той обаче се

използува само когато тактовият генератор на емулираната схема апаратно не е завършен или при самостоятелна работа на емулатора. Обикновено той се настройва да работи на максимална тактова честота на микропроцесора. По принцип емулаторът получава тактова честота от емулираната система, тъй като по този начин се осигурява тестване в реалните времеви условия.

Останалите три блока определят спецификата на микропроцесорния емулатор и затова е необходимо да се обърне повече внимание както за използването им, така и за евентуалното им проектиране. Функциите на някои от блоковете могат да бъдат изпълнени изцяло по програмен път от мониторната програма, затова такива блокове могат да не присъстват апаратно в емулатора.

4.2. Управление на адресното пространство

Блокът за управление на адресното пространство (картата на паметта) осигурява решаване на адресния конфликт между емулатора и тестваната микропроцесорна система. Същевременно той разрешава и разпределя потребителските ресурси в адресното пространство. Адресният конфликт възниква поради това, че двете свързани системи (емулаторна и тествана), се управляват от един микропроцесор. Всяка от системите си има свое адресно пространство, а от двете пространства трябва да се направи едно общо, което е във възможностите на общия микропроцесор.

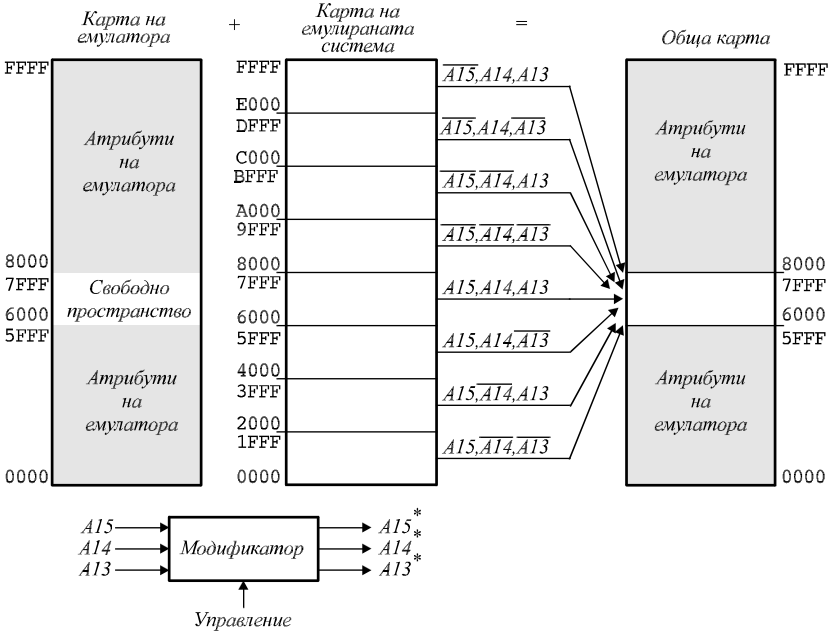


Фиг. 4.2. Модифициране на адресното пространство на емулираната система.

Най-простият начин за разрешаване на адресния конфликт е чрез модифи-

цириране на адресното пространство на тестваната микропроцесорна система. Подаваните от емулатора адреси към тестваната система преминават през управляем модификатор. Начинът е илюстриран на фиг. 4.2. Адресното пространство на емулатора има определена празна (незаета) част и част, заета от неговите атрибути (ROM, RAM, I/O). От двете адресни пространства се прави едно общо, което включва заетата от атрибутите на емулатора част от адресното му пространство и частта от пространството на тестваната система, незатъмнена от атрибутите на емулатора (съответстваща на свободната му част).

Когато микропроцесорът издаде адрес от областта, заета от емулаторните атрибути, обръщението се извършва към емулаторното пространство. Когато микропроцесорът издаде адрес от свободната зона, обръщението се извършва към пространството на емулираната система (обръщението трябва да бъде съпроводено със съответно апаратно управление на буферния блок). При немодифициран от модификаторния блок адрес емулаторът има пряк достъп до частта от адресното пространство на тестваната система, съответстваща на свободното пространство на емулатора.



Фиг. 4.3. Модифициране на три адресни линии.

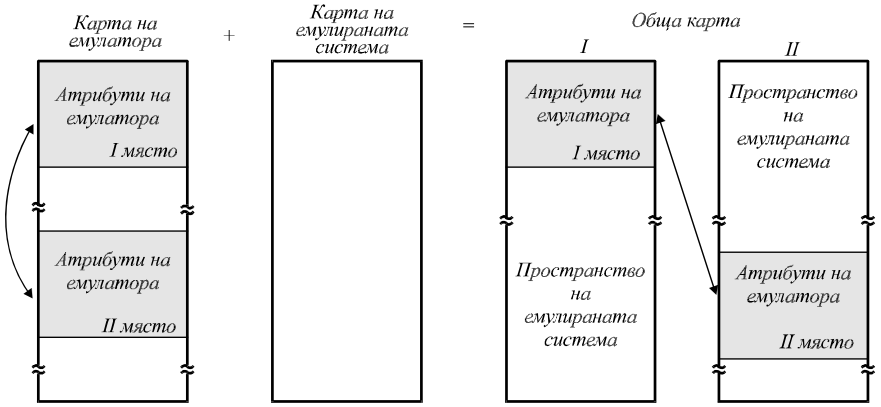
Когато потребителят желае да получи достъп до частта от адресното прост-

ранство на емулираната система, което е затъмнено от емулаторните атрибути, се подава команда към адресния модификатор, който променя адресите към тестваната система. Адресите се модифицират така, че затъмнената част да се премести в свободното пространство на емулатора.

Когато атрибутите на емулатора заемат повече от половината от адресното пространство на процесора, достъпът до цялото адресно пространство на тестваната система се извършва чрез повече от едно модифициране.

Примерът от фиг. 4.3 илюстрира приложението на метода при емулатор, чиито атрибути заемат повече от половината от адресното пространство на микропроцесора. В показания случай на модифициране са подложени старшите три адресни линии *A15*, *A14* и *A13*, като модифицирането се състои в управляемото им инвертиране. Тук емулаторът има достъп през адресната врата от `6000` до `7FFF` към пространството на тестваната система чрез 7 допълнителни модифицирания на адреса.

Втори начин за разрешаване на адресния конфликт е чрез преадресиране на атрибутите на емулатора (смяна на адресната карта). При този начин достъпът до цялото адресно пространство на тестваната система също се извършва на няколко пъти. Както и при предишния начин, от двете адресни пространства се прави едно общо, което включва заетите от атрибутите на емулатора част от адресното му пространство и допълващата го част от пространството на тестваната система.



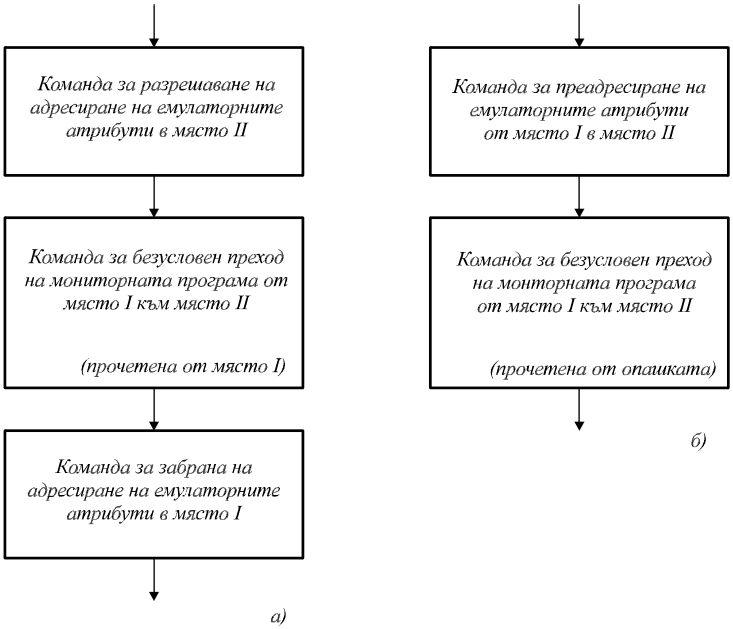
Фиг. 4.4. Смяна на адресната карта на емулатора.

Микропроцесорът има пряк достъп до допълващата част от пространството на тестваната система. Достъпът до други части на адресното пространство на тестваната система се осъществява след преадресиране (преместване в адресното пространство) на атрибутите на емулатора. След това преместване се съставя

общо адресно пространство, съдържащо атрибутите на емулатора и допълващата го част от пространството на тестваната система. Начинът е илюстриран на фиг. 4.4.

Предимството на този начин спрямо предишния е, че достъпът на микропроцесора до тестваната система е винаги пряк, което дава възможност в нея да се изпълняват потребителски програми.

Преадресирането на атрибутите на емулатора се извършва по програмен път с команда от оператора. Алгоритъмът на преадресирането е показан на фиг. 4.5. Необходимо е, в определен момент преди и след скока на програмата на монитора на новото място, постоянната памет с мониторната програма да бъдат адресирани едновременно и на старото, и на новото място (фиг. 4.5.а). Това е за да може да се прочете инструкцията за безусловен преход от старото място.



Фиг. 4.5. Последователност за преадресирането на емулаторните атрибути.

При микропроцесори, които притежават вътрешна опашка на инструкциите, е възможно програмата да бъде така разчетена, че инструкцията за безусловен преход да е влязла в опашката, когато се извършва апаратното преместване на мониторната програма и тя се взима от там (фиг. 4.5.б). Преадресирането на останалите атрибути на емулатора (RAM, I/O) може да стане отделно или заедно с преадресирането на мониторната програма.

Предимството на този начин има спрямо първия е, че достъпът на микропроцесора до тестваната система е винаги пряк.

Трети начин за решаване на адресния конфликт е изграждането на две отделни карти на паметта (dual map) (за емулатора – системна карта, и за тестваната система – потребителска карта). Всяка една от картите разполага с цялото адресно пространство на микропроцесора. Разпределението на действията в двете адресни карти се извършва от специално изграден апаратно-програмен арбитър.

При този случай се изхожда от предпоставката, че всяка микропроцесорна команда притежава изпълнителна фаза. Ако по време на изпълнителната фаза се забрани обръщението към системното адресно пространство и се разреши това към потребителското, то предписаното действие от командата ще се извърши в потребителското пространство.

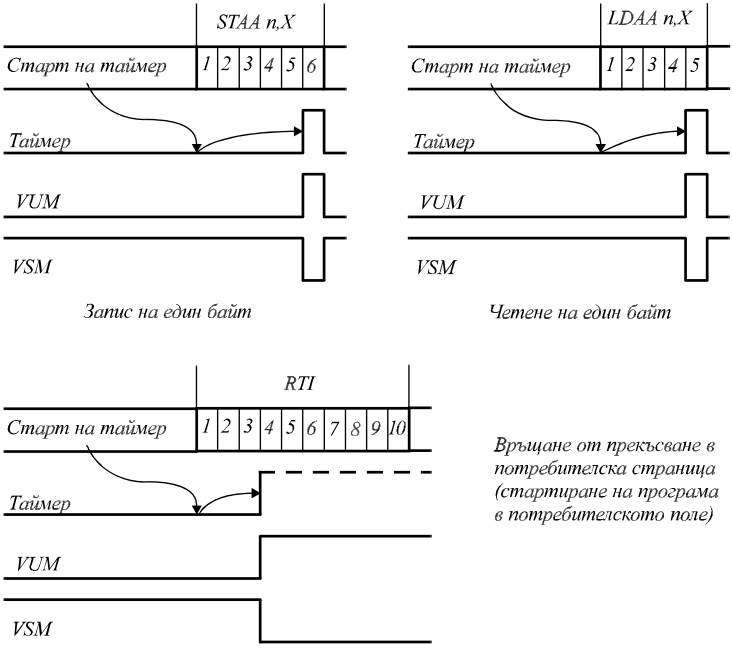
Арбитърът е този, който в определени моменти разрешава и забранява потребителската и системната карта. Неговата програмна част се съдържа в мониторната програма. Тя следи кога операторът иска работа в потребителското пространство, какъв тип е тази работа и инициализира по подходящ начин апаратната част на арбитъра. Работата на оператора в потребителското пространство е ограничена до определен брой функции.

Минималният набор от функции зависи от конкретния микропроцесор. За всяка функция се използува строго определена микропроцесорна команда и се извършва специфично спрямо нея инициализиране на апаратната част на арбитъра. Последната представлява генератор на времеви интервали, които се изработват по време на изпълнителните фази на определените микропроцесорни команди. Генерираният времеви интервал забранява работата в системното адресно пространство и разрешава потребителското адресно пространство чрез допълнително формираните сигнали *VSM* (валидна системна карта) и *VUM* (валидна потребителска карта).

Начинът е илюстриран на фиг. 4.6 с реализиране на обръщения към потребителско пространство при емулиране на система с микропроцесора MC6800. Минималеният набор от функции за този микропроцесор (както и за повечето 8-разредни микропроцесори и микрокомпютри на Motorola) включва операциите: четене на байт, запис на байт, предаване на управлението в потребителското пространство.

За четене и за запис на байт в потребителската страница са определени съответно инструкциите, имащи мнемоника `LDAA n,X` с изпълнителна фаза в петия такт и `STAA n,X` с изпълнителна фаза в шестия такт. Адресът на обръщение в потребителското пространство трябва да бъде предварително зададен в индексния регистър, а информацията за четене или запис се получава или задава предварително с акумулатора A. Арбитърът трябва да бъде инициализиран да генерира единичен импулс (с продължителност, равна на продължителността на

един микропроцесорен такт), който да съвпада по време съответно с петия или шестия такт на горните инструкции.



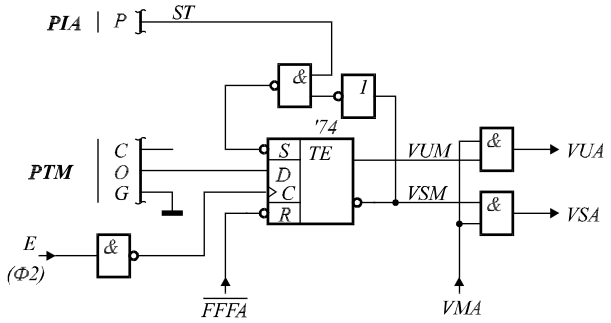
Фиг. 4.6. Организиране на двойна карта на паметта за микропроцесор MC6800.

По-особена е ситуацията при стартиране на потребителска програма. Тази функция може да се извърши, като чрез команда за запис в потребителското пространство в мястото на стека се запише съдържанието на потребителските микропроцесорни регистри, след което се изпълни инструкция за връщане от прекъсване.

За да може микропроцесорните регистри да се заредят от потребителската страница и да се извърши връщането там, е необходимо непосредствено след третия такт от инструкцията `RTI` да се забрани системното пространство и да се разреши потребителското. Тогава микропроцесорните регистри ще бъдат прочетени от потребителското пространство и изпълнението на програмата ще продължи там. В този случай арбитърът трябва да генерира непрекъсван импулс (ниво) за смяна на пространствата.

Апаратната част на арбитъра се управлява от значително програмно осигуряване, включено в мониторната програма. Непосредствено преди изпълнението на определените инструкции за обръщение към потребителското адресно

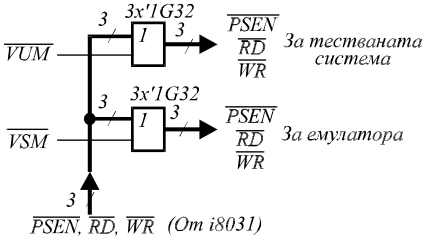
пространство микропроцесорът изпълнява команда, предизвикваща стартиране на работата на арбитъра за генериране на импулс.



Фиг. 4.7. Апаратна част на арбитъра на двойната карта на паметта.

На фиг. 4.7. е показана апаратната част на арбитър, използуващ изход на една секция от програмируем таймер MC6840, тактуван вътрешно по сигнала *E*. С допълнителен сигнал *ST* (генериран от потенциален изход в емулатора – например от PIA) се указват режим на единичен импулс за четене и запис на байт и режим на ниво – за продължаване на работата в потребителското пространство след връщане от прекъсване. При *ST = 0* тригерът ‘74 само синхронизира единичния импулс от PTM със спадащия фронт на *E*. При *ST = 1*, след преминаване на тригера в състояние 1, той се "заключва" в това състояние поради действащата обратна връзка от изхода *Q* към входа $\overline{S}$ и може да бъде нулиран само при подаването на сигнал $\overline{FFFA} = 0$ на входа му $\overline{R}$.

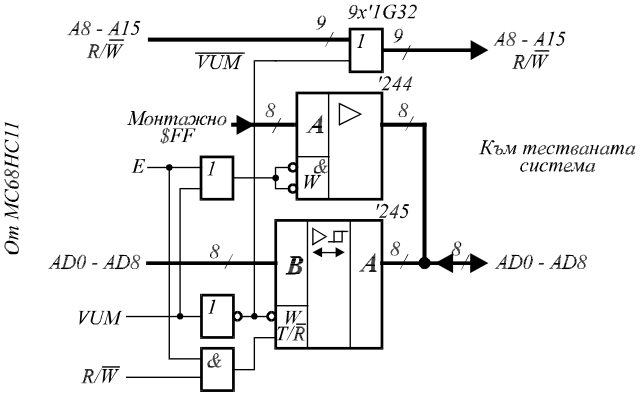
Физическото разрешаване и забраняване на адресните пространства се извършва, като чрез допълнителна логика се модифицира сигналът *VMA* – вместо него към емулатора се подвежда сигналът *VSA* (валиден системен адрес), а към емулираната система – *VUA* (валиден потребителски адрес).



Фиг. 4.8. Арбитриране на управляващи сигнали за двойна карта при i8031.

Всеки микропроцесор има подобен сигнал или начин, чрез който указва невалидно обръщение към адресното пространство, можещ да се използува за целите на разрешаване или забраняване на обръщение към паметта. Например i8031 не активира $\overline{PSEN}$, $\overline{RD}$ или $\overline{WR}$. Като пример на фиг. 4.8 е показана допълнителна логика за физическото разрешаване или забраняване на адресните пространст-

ва при i8031. По-сложно е поведението на MC68HC11, който в момент на невалидно обръщение към паметта извършва четене от адрес FFFF. При него допълнителната логика за разрешаване на двете адресни пространства трябва да се съчетае с подходящо буфериране на сигналите. На фиг. 4.9 е показано буферирането на сигналите към тестваната система.



Фиг. 4.9. Буфериране на сигнали при 68HC11 за работа с двойна карта на паметта.

Връщането на управлението на микропроцесора от потребителското пространство (при стартирана потребителска програма) в мониторната програма на емулатора се извършва чрез използуване на някое прекъсване. За целта в емулатора се изгражда апаратна система, която следи възникването на определеното прекъсване и забранява работата в потребителското пространство, разрешавайки емулаторното.

При емулатор с MC6800, MC68HC11 и др. може да се изгради адресен компаратор, който следи адресирането на младшия адрес на вектора на програмното прекъсване FFFA и връща управлението в системното пространство. Използуваните прекъсвания за връщане в системното пространство преминават през проверка, която определя дали те са генерирани за връщане или са потребителски прекъсвания. Във втория случай микропроцесорът се препраща към потребителските вектори на прекъсване и се прехвърля отново в потребителската страница. Тези проверки и манипулации забавят изпълнението на потребителските прекъсвания (ако същите прекъсвания са използуваните и за връщане в системната страница) и те се обслужват извън реално време.

Предимство на третия начин за разрешаване на адресния конфликт е, че потребителят може да ползува пряко цялото адресно пространство на емулираната система и да стартира там произволна потребителска програма. Недостатъкът е значителното му апаратно и програмно обезпечаване.

4.3. Диагностично прекъсване на програми

Диагностичното прекъсване на програма (блок за поставяне точки на прекъсване) осигурява управляемо прекъсване на изпълнението на потребителска програма при достигане на указани адреси и връщане на управлението на мониторната програма. Това позволява анализиране на работата на отделни части от потребителската програма.

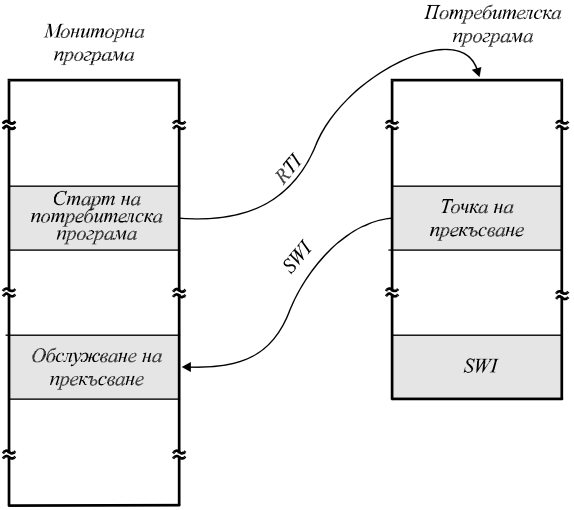
Функцията може да се реализира от мониторната програма изцяло по програмен път (фиг. 4.10). В този случай се използува някое от програмните прекъсвания на микропроцесора (за MC6800, MC68HC11 – `SWI = 3F`).

Реализирането на функцията протича в следния порядък:

1. Мониторната програма запомня адресите, на които е поставила точки на прекъсване, и истинските инструкции, които трябва да бъдат там.

2. Непосредствено преди стартирането на потребителската програма на желаните адреси, където тя трябва да прекъсне своето изпълнение, се извършва подмяна на истинските кодове на инструкциите с кода на инструкцията за програмно прекъсване. Това са т.нар. точки на прекъсване.

3. Извършва се стартиране на потребителската програма.



Фиг. 4.10. Точка на прекъсване в потребителска програма с програмно прекъсване.

След като при изпълнението на потребителската програма се достигне до инструкция за програмно прекъсване, микропроцесорът започва да изпълнява съответната процедура по обслужването му, намираща се в мониторната програма.

4. Процедурата за обслужване на програмното прекъсване проверява дали прекъсването е генерирано вследствие на достигане на точка на прекъсване (като сравнява адреса на прекъсването с адресите на запомнените точки на прекъсване). Ако програмното прекъсването не е генерирано от точка на прекъсване, управлението на програмата се предава на потребителската процедура за обслужване на програмното прекъсване.

5. Ако програмното прекъсване е генерирано от точка на прекъсване, заменените от програмното прекъсване истински инструкции се възстановяват и управлението на програмата се предава на командния интерпретатор на мониторната програма.

С последното се дава възможност на оператора, използувайки командите на мониторната програма, да анализира изпълнената до точката на прекъсване част от потребителската програма.

Чрез програмния начин на реализиране на диагностично прекъсване на програми могат да се поставят едновременно няколко точки на прекъсване. Той има и предимството, че не се нуждае от никакво апаратно осигуряване. Като недостатък обаче се сочат следните елементи: необходимост от значително програмно осигуряване; забавено реагиране на потребителско програмно прекъсване и задължително поставяне на потребителската програма в оперативна памет.

Апаратното реализиране на функцията диагностично прекъсване на потребителска програма използува апаратно прекъсване на микропроцесора (най-често – немаскируемо прекъсване). Към емулатора се изгражда адресен компаратор, следящ за поява на указания адрес или адреси. Реализирането на функцията протича в следния порядък:

1. Мониторната програма запомня адресите, на които е поставила точки на прекъсване.

2. Непосредствено преди стартирането на потребителската програма в компаратора (компараторите) се зареждат желаните адреси, където потребителската програма трябва да прекъсне своето изпълнение.

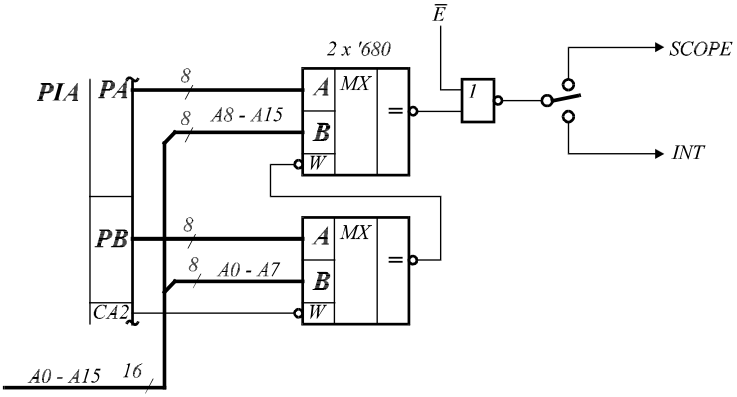
3. Извършва се стартиране на потребителската програма.

След като при изпълнението на потребителската програма се достигне до заредените адреси, адресният компаратор сработва и генерира апаратно прекъсване към микропроцесора и той започва да изпълнява съответната процедура по обслужването на апаратното прекъсване, намираща се в мониторната програма.

4. Процедурата за обслужване на апаратното прекъсване проверява дали прекъсването е генерирано вследствие на достигане на точка на прекъсване (като сравнява адреса на прекъсването с адресите на запомнените точки на прекъсване). Ако програмното прекъсването не е генерирано от точка на прекъсване, управлението на програмата се предава на потребителската процедура за обслужване на апаратното прекъсване.

5. Ако апаратното прекъсването е генерирано от точка на прекъсване, управлението на програмата се предава на командния интерпретатор на мониторната програма.

Най-простата схема на адресен компаратор за адресно пространство с 16 адресни линии е показана на фиг. 4.11. Тя включва 16-разреден компаратор на адресните линии, изграден с ‘680. Адресът на сравнение се задава от страни A и B на два 8-разредни порта (например PIA) с разрешение от допълнителен портов извод (в случая CA2 на PIA). В зависимост от положението на превключвателя резултатът от сравняването се подава като стробиращ сигнал *SCOPE* за синхронизиране на осцилоскоп или като сигнал *INT* за формиране на прекъсване за микропроцесора. Допълнително във формирането на изходящия сигнал се включва разрешение $\overline{E}$, което определя сравняването само при валиден и стабилен адрес, за да се избегнат грешни сработвания на компаратора от преходни процеси в адреса.



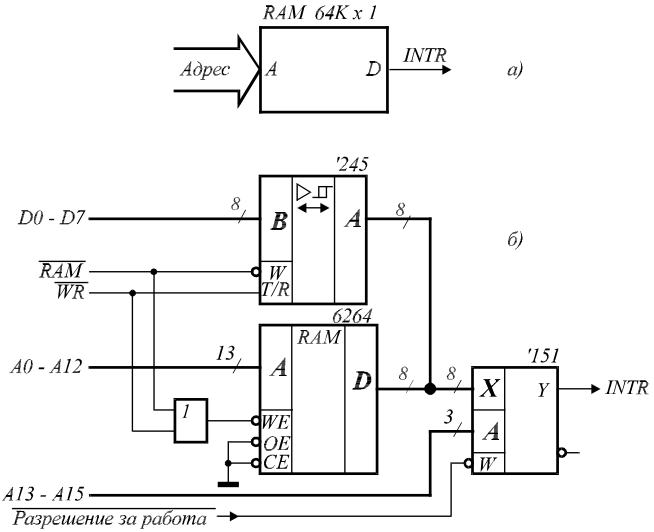
Фиг. 4.11. Адресен компаратор.

Тази схема позволява поставянето на една-единствена точка на прекъсване. Поставянето на неограничен брой точки на прекъсване може да се реализира чрез използуването на еднобитова оперативна памет, като на всеки адрес от адресното пространство съответства по един бит от паметта. В тези битове мониторната програма записва `1` или `0`, ако на съответстващите им адреси трябва или не трябва да се постави точка на прекъсване. След стартиране на потребителската програма при нейното изпълнение се извършва и едновременно адресиране на еднобитовата памет. В зависимост от съдържанието й при издаването на определени адреси ще се генерират сигнали за прекъсване на микропроцесора. Идейната схема на използуване на еднобитова памет е показана на фиг. 4.12.а, а на 4.12.б е показано организирането на еднобитова памет 64K×1 от па-

мет с организация 8K×8.

Апаратният начин за диагностично прекъсване на програми има предимството, че позволява поставяне на точки на прекъсване и за програми, записани в постоянна памет. Като недостатък обаче може да се посочат следните елементи: необходимост от апаратно осигуряване; забавено реагиране на потребителско апаратно прекъсване.

Като особеност трябва да се има предвид, че при използуването на апаратно прекъсване потребителската програма спира след изпълнение на инструкцията, намираща се в точката на прекъсване, а при използуване на програмно прекъсване – преди изпълнение на инструкцията.

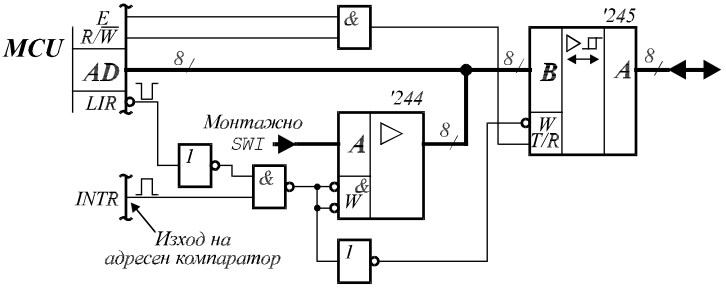


Фиг. 4.12. Организиране на еднобитова памет за апаратно поставяне на точки на прекъсване.

В някои емулатори се прилага комбиниран програмно-апаратен начин за диагностично прекъсване на програми. При него също се използува адресен компаратор. Изходният му сигнал обаче не подава апаратно прекъсване към микропроцесора, а забранява буферите за данни между микропроцесора на емулатора и останалата част от информационната магистрала, и разрешава еднопосочен буфер, подаващ на микропроцесора командата за програмно прекъсване (формирана по монтажен начин) вместо текущата команда.

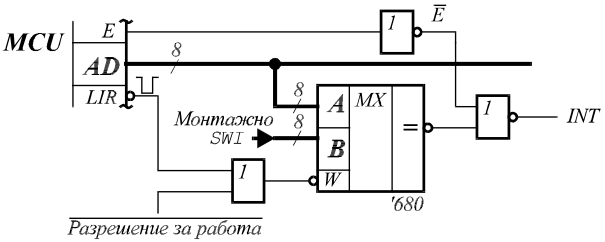
Този комбиниран начин се прилага при микропроцесори, които не притежават вход за апаратно прекъсване, или използуването на апаратното прекъсване е нежелателно. Начинът също позволява поставяне на точки на прекъсване в програма, записана в постоянна памет.

Схемата от фиг. 4.13 илюстрира този комбиниран начин (за същата цел се използува програмното прекъсване на микропроцесора). Ако съществува възможност, сигналът от изхода на адресния компаратор се разрешава допълнително от сигнал на микропроцесора, указващ прочитане на код на операция. С това се избягва фалшиво сработване на системата за поставяне на точки на прекъсване, когато микропроцесорът се обръща за обикновено четене или запис към маркирани адреси. Такъв е например сигналът $\overline{LIR}$ (Latch Instruction) при едночиповия микрокомпютър MC68HC11, *LI* при MC146805E2 и т.н.



Фиг. 4.13. Комбиниран програмно-апаратен начин за реализиране на точки на прекъсване.

Понякога се реализира и друг комбиниран апаратно-програмен начин за диагностично прекъсване на програми. При него чрез компаратор на магистралата за данни се открива инструкцията за програмно прекъсване и се подава сигнал за апаратно прекъсване към микропроцесора. Фиг. 4.14 илюстрира този комбиниран начин. Той се прилага тогава, когато използуването на програмното прекъсване е нежелателно. Начинът не позволява поставяне на точки на прекъсване в програма, записана в постоянна памет.

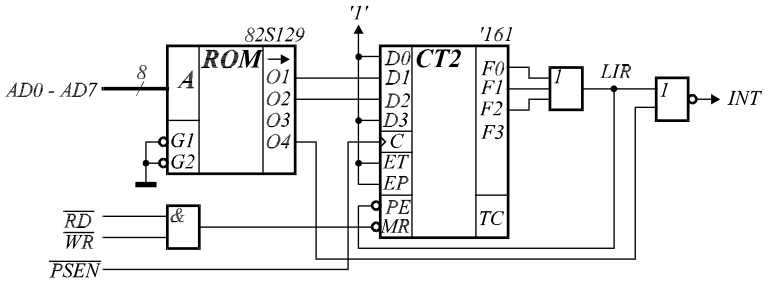


Фиг. 4.14. Комбиниран апаратно-програмен начин за реализиране точки на прекъсване.

Съществуват микропроцесори, които не притежават инструкция за програмно прекъсване. Тогава друга някоя инструкция може да бъде набелязана да из-

пълнява тази роля. Инструкцията трябва да бъде с минималната за процесора дължина, да се изпълнява за минималния брой инструкционни цикли и да има строго дефинирано действие. Последното е необходимо, за да може след прихващането на инструкцията и генерирането на апаратно прекъсване, въздействието ѝ да бъде отменено. Такъв е случаят с едночиповия микрокомпютър i8031. Там може да бъде използувана инструкцията `INC DPTR` (Increment Data Pointer), траеща 2 процесорни цикъла, код `A3`, с действие DPTR ← DPTR+1 (увеличава с 1 показалеца на данните). След влизане в прекъсване показалецът на данните трябва да бъде намален с 1.

Неудобство при някои процесори е липсата на сигнал, който да указва, че се прочита валиден код на инструкция ($\overline{LIR}$). Такъв сигнал може да бъде синтезиран допълнително. Фиг. 4.15 дава примерно синтезиране на сигнал за прочитане на валидна инструкция за микрокомпютъра i8031.



Фиг. 4.15. Синтезиране на сигнал за извличане на инструкция при i8031.

С постоянна памет 82S129 е изграден компаратор, който активира изхода си при откриване на определена инструкция (изход *O4*). Същевременно същата памет генерира в изходите си *O1* и *O2* число, което съответства на броя инструкционни цикли на постъпващата на входа инструкция. Това число се зарежда в брояч ‘161, който, тактуван от $\overline{PSEN}$, отброява числото и след препълване се нулира. Числото е оразмерено така, че препълването на брояча да става в последния инструкционен цикъл на текущата инструкция. Естественото нулиране на брояча съвпада с първия инструкционен цикъл на следващата инструкция, което е нейното извличане. Броячът принудително се нулира при процесорен цикъл на четене и запис в данновото пространство, тъй като след това задължително следва цикъл на извличане на инструкция.

4.4. Трасиране на програми

Блокът за трасиране на програми осигурява покомандно изпълнение на участъци от потребителската програма. След изпълнението на всяка инструкция

потребителската програма се преустановява и се дава възможност за проверка на резултата от изпълнението й, а също и за преглед на микропроцесорните регистри, някои клетки от паметта, входно-изходните устройства и др. Така след всяка инструкция се добива представа за цялостното състояние на тестваната система. Изпълнението на всяка следваща инструкция се извършва по указание на оператора. Този режим на изпълнение на програма се нарича още изпълнение на единични инструкции или изпълнение стъпка по стъпка. Функцията може да се реализира по програмен и по апаратен начин.

При програмния начин се използува програмно прекъсване на микропроцесора. Функцията протича в следния порядък:

1. Преди единичното изпълнение на дадена инструкция мониторната програма изчислява нейната дължина (по кода на инструкцията), за да се намери адресът на следващата инструкция.

2. Кодът на следващата инструкция се подменя с кода на инструкцията за програмно прекъсване.

3. Мониторната програма вдига вътрешен програмен флаг за работа в режим на трасиране.

4. Извършва се стартиране на потребителска програма от адреса на инструкцията, която ще се изпълни поединично (нейният адрес се зарежда в мястото на стека, съответстващо на програмния брояч, и се изпълнява инструкцията за връщане от прекъсване).

След стартирането микропроцесорът изпълнява набелязаната инструкция, след нея попада на поставената инструкция за програмно прекъсване и тръгва да изпълнява съответната процедура по обслужването му, намираща се в мониторната програма.

5. Процедурата за обслужване на програмното прекъсване проверява дали прекъсването е генерирано вследствие на изпълнение на единична инструкция (като проверява вътрешния програмен флаг за работа в режим на трасиране). Ако програмното прекъсването не е генерирано при работа в този режим, управлението на програмата се предава на потребителската процедура за обслужване на програмното прекъсване.

6. Ако програмното прекъсването е генерирано в режим на трасиране, се възстановява заменената от програмното прекъсване истинска инструкция и управлението на програмата се предава на командния интерпретатор на мониторната програма.

С последното се дава възможност на оператора, използувайки командите на мониторната програма, да анализира изпълнението на единичната инструкция.

Начинът не изисква апаратно осигуряване, но може да се прилага само към програми, записани в оперативна памет. Освен това е необходимо доста сложно програмно осигуряване, особено когато се трасират инструкции за условен преход, чието действие трябва предварително да се проиграе.

Апаратният начин за осигуряване на трасиране на потребителски програми използува някое от апаратните прекъсвания на микропроцесора (най-често – немаскируемо прекъсване). Той изисква организирането на програмно управляем генератор на времеви интервал (таймер), изработващ сигнала за апаратното прекъсване. Този начин е илюстриран със схемата и времедиаграмата на фиг. 4.16, приложими за микропроцесори на Motorola (например MC6800, MC68HC11 и др.).

Реализирането на функцията протича в следния порядък:

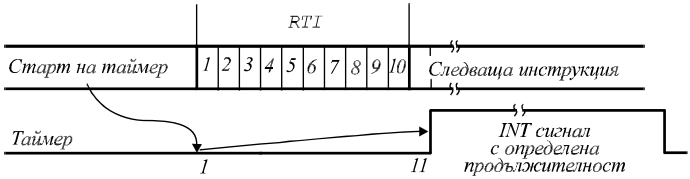
1. Непосредствено преди изпълнението на единичната инструкция мониторната програма записва адреса на инструкцията в мястото от стека, съответстващо на програмния брояч и инициализира таймера за генериране на единичен импулс.

2. Мониторната програма вдига вътрешен програмен флаг за работа в режим на трасиране.

3. Мониторната програма стартира таймера, след което изпълнява командата "връщане от прекъсване" (`RTI`) за стартиране на потребителската програма.

Таймерът предварително е така инициализиран, че отработва 11 периода на тактовия сигнал *E* и генерира сигнал за апаратно прекъсване. През времето на 11-те периода на *E* процесорът е изпълнил инструкцията `RTI`, траеща 10 периода, и е изпълнил първия такт от единично изпълняваната инструкцията. При получаване на апаратно прекъсване микропроцесорът завършва изпълнението на започнатата инструкция и влиза в процедура за обслужване на прекъсването, намираща се в мониторната програма.

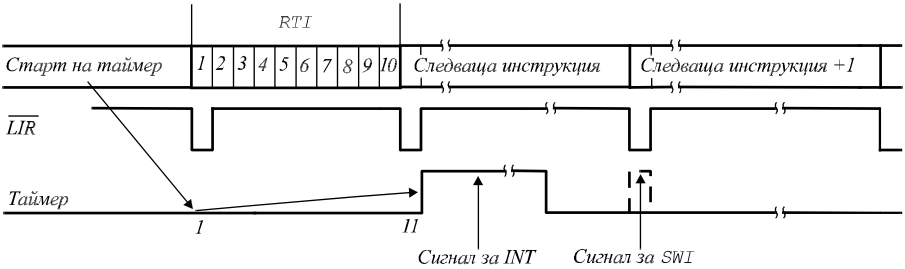
4. Процедурата за обслужване на апаратното прекъсване проверява дали прекъсването е генерирано вследствие на изпълнение на единична инструкция (като проверява вътрешния програмен флаг за работа в режим на трасиране). Ако програмното прекъсването не е генерирано при работа в този режим, управлението на програмата се предава на потребителската процедура за обслужване на апаратното прекъсване.



Фиг. 4.16. Апаратен способ за трасиране на програми.

5. Ако апаратното прекъсването е генерирано в режим на трасиране, управлението на програмата се предава на командния интерпретатор на мониторната програма.

При микропроцесори които притежават апаратен сигнал, указващ четене на код на инструкцията $\overline{LIR}$, инструкцията за връщане от прекъсване се пропуска и след прочитане на кода на единично изпълняваната инструкция се генерира прекъсване към микропроцесора. На фиг. 4.17 е показано генерирането на сигнал за немаскируемо прекъсване при единично изпълнение на инструкции.



Фиг. 4.17. Изпълнение на единична инструкция с използуване на сигнал за извличане на кода на операцията.

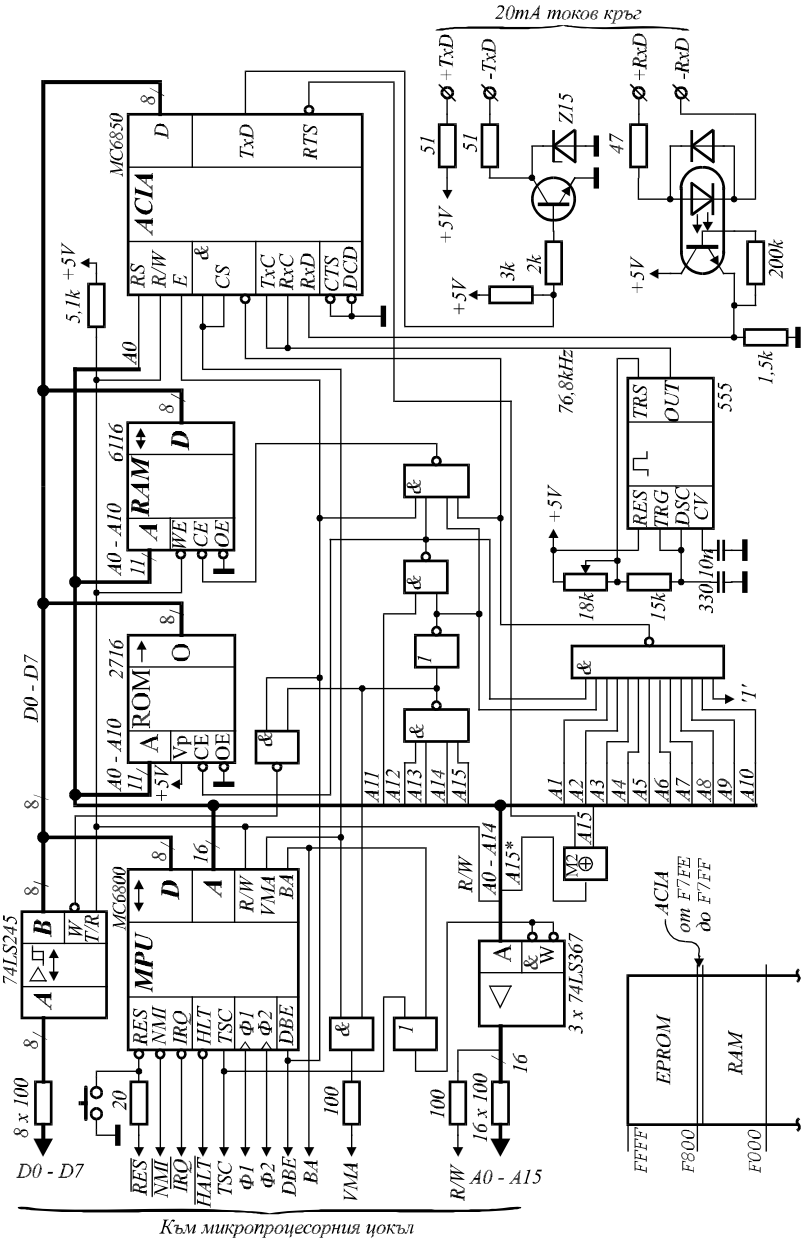
Апаратният начин също има разновидност, при която вместо подаване на апаратно прекъсване информационните линии се прекъсват към микропроцесора и на него се подава кодът на програмното прекъсване. В този случай сигнал за неговото превключване се изработва след единично изпълняваната инструкция (показан с прекъсната линия на фиг. 4.17).

Апаратният начин за трасиране на програми има предимство, че могат да се трасират и програми, записани в постоянна памет, но изисква допълнително апаратно осигуряване.

Необходимо е да се отбележи, че повечето 16-разредни микропроцесори притежават вградена възможност за трасиране на програми. Те имат специален флаг за трасиране на програми в регистъра на състоянията. При вдигнат флаг за трасиране след всяка изпълнена инструкция микропроцесорът отново влиза вътрешно в прекъсване. Например при микропроцесора i8086 това е флагът TF в думата на състоянието PSW. Същата роля играе и бит T в регистъра на състоянието на микропроцесора MC68000.

4.5. Схемотехника на микропроцесорен емулатор

На фиг. 4.18 е показана прост вътрешносхемен микропроцесорен емулатор за системи, изградени на базата на микропроцесора СМ601. Схемата притежава, макар и в неявен вид, всички основни блокове от фиг. 3.1. Връзката с оператора се извършва чрез външен терминал, който се съединява с емулатора посредством сериен асинхронен канал "20 *mA* токов кръг". Скоростта на обмена е 4800 bit/s, като тактовата честота за обмена се генерира от таймер 555 (76,8 kHz).



Фиг.4.18. Вътрешносхемен микропроцесорен емулатор с СМ601.

Заеманото от емулатора адресно пространство е съсредоточено в зоната от F000 до FFFF с обем 4K byte. Върхът на оперативната памет – F7FE и F7FF е блокиран и там е разположен ACIA.

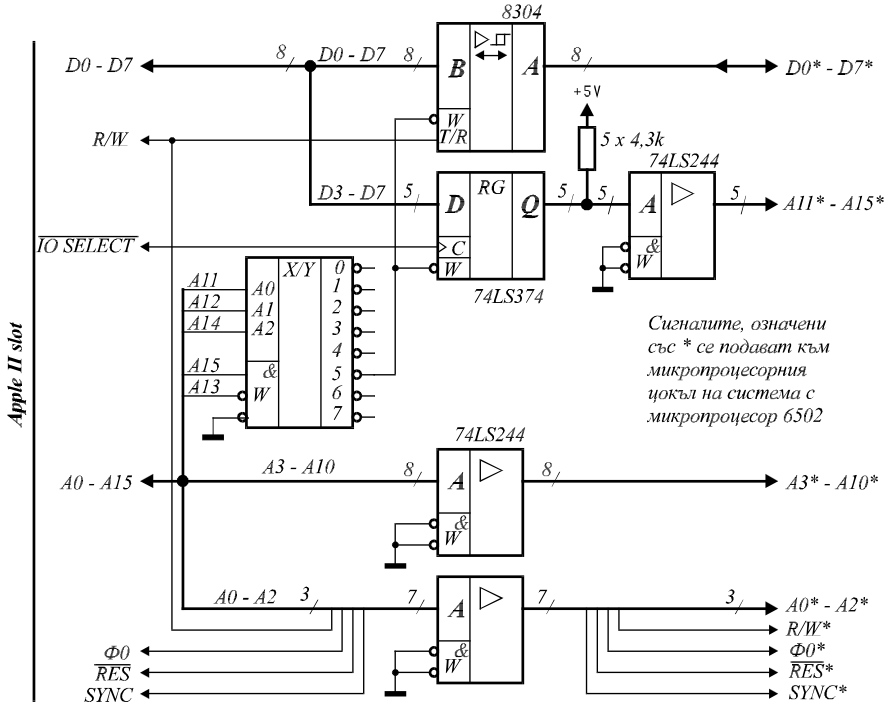
Общата карта на адресното пространство се образува от зоната от 0000 до EFFF на проверяваната система и зоната от F000 до FFFF на емулатора. Когато микропроцесорът се обръща към адреси от зоната на емулатора, буферите за данни 74LS245 се забраняват и сигналът VMA към емулираната система се поставя в логическа *0*. Адресният конфликт е разрешен чрез модифициране на адресното пространство на тестваната система. Емулаторът има пряк достъп до зоната от 0000 до EFFF и непряк – до зоната от F000 до FFFF на проверяваната система. Достъпът се извършва на два пъти. По команда от оператора под управление от изхода $\overline{RTS}$ на ACIA се инвертира изпращаната към емулираната система адресна линия *A15*. Управляемото инвертиране се реализира чрез елемент EOR. Адресната зона от F000 до FFFF на проверяваната система се трансформира в зона от 7000 до 7FFF, към която емулаторът може да осъществи достъп. Мониторната програма е заредена в EPROM и ползува за системни цели оперативната памет от F700 до F7FD. Оперативната памет от F000 до F6FF е на разположение на потребителя.

Трасирането на програми в показания емулатор се осъществява по програмен път. Тези действия могат да се прилагат само върху програми, поставени в оперативна памет. Схемата на емулатора е компактна и той може да бъде изграден така, че да се свързва с микропроцесорния цокъл на тестваната система без съединителен кабел, като се "забожда" непосредствено в него.

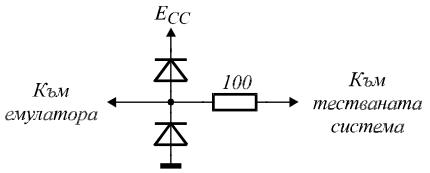
Вътрешносхемният микропроцесорен емулатор може да бъде изграден като периферен модул за персонален компютър (комплексен емулатор). На фиг. 4.19 е показана принципната схема на вътрешносхемен емулатор за микропроцесора 6502, конструктивно оформен като стандартна периферна платка за компютъра Apple II. Достъпът до адресното пространство на проверяваната система става на 32 пъти по 2K byte през обикновено свободното компютърно пространство от C800 до CFFF. Сигналите от адресните линии от *A0* до *A10*, а също и управляващите сигнали $R/\overline{W}$, $\overline{RES}$, *Ф0* и *SYNC*, постъпват направо от емулиращата система през буфери 74LS244. Старшите адресни линии от *A11* до *A15* не се генерират от адресната магистрала на Apple II, а по програмен начин се записват техни модифицирани стойности в регистъра 74LS374. Записът се осъществява от сигнала $\overline{I/O\_SELECT}$. Така се реализира модифицирането на адресното пространство на тестваната система.

Стремежът при проектиране на микропроцесорен емулатор е колкото се може по-точно да се копира апаратното поведение на заместваният микропроцесор. Емулираната микропроцесорна система не трябва да прави разлика между управление от микропроцесор или от негов емулатор. Затова се вземат мерки входящите в емулатора сигнали да изпитват същото натоварване, каквото би

оказал реалният микропроцесор.



Фиг. 4.19. Комплексен емулатор за Apple II.



Фиг. 4.20. Защита на извод на емулатор.

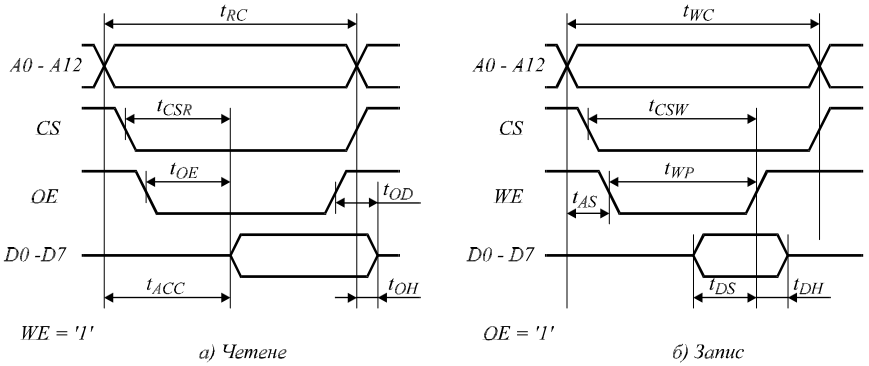
Същевременно на изходящите от емулатора сигнали се осигурява същата товароспособност, както и на реалните микропроцесорни изходи. По пътя на изходящи сигнали, които се генерират от схеми с по-голяма товароспособност, се поставят резистори. Изводите на емулатора трябва да бъдат защитени и от подаването на некоректни напрежения. Такава проста защита е показана на фиг. 4.20.

Основното приложение на вътрешносхемните микропроцесорни емулатори е в етапа на разработване и настройване на изделия. Те се използуват самостоятелно или по-често заедно с развойни системи.

5. ЛОГИЧЕСКИ АНАЛИЗ

В микропроцесорните системи информацията за дадена операция съществува за кратко време върху системните шини. Например, ако се прегледат циклите на четене и запис за статична оперативна памет в една микропроцесорна система (фиг. 5.1 за SRAM 6264-12), ще се види, че те продължават части от микросекундата.

В цикъла за четене (фиг. 5.1.а) времето на избор е t_{ACC}, което е интервала от установяването на адреса до появата на правилни данни и максималната му стойност е 120 *ns*. Другите по-важни времеви величини означават следното: t_{CSR} – максимално време от избора на чипа до прочитането на правилни данни; t_{OE} – максимално време от активирането на сигнала $\overline{OE}$ до прочитането на правилни данни; t_{OD} – минимално време от дезактивирането на $\overline{OE}$ до затварянето на изходните буфери; t_{OH} – минимално време на задържане на прочитаните данни след промяна на адреса; t_{RC} – продължителност на цикъла четене (минимално време, необходимо за еднократно протичане на данните). Всички те се измерват в наносекунди.



Фиг. 5.1. Времедиаграми на циклите четене и запис на SRAM – 6264-12.

В цикъла на запис (фиг. 5.1.б) минималното време от избора на паметта до края на записа е t_{CSW} , което в дадения случай е 100 *ns*. Останалите по-важни времеви параметри, чиято продължителност също се измерва с наносекунди, са: t_{WP} – минимална продължителност на активното състояние на сигнала $\overline{WE}$; t_{DS} – минимално необходимо време на установени данни преди дезактивирането на $\overline{WE}$; t_{AS} – минимално необходим интервал от време от установяването на адре-

са до активирането на сигнала $\overline{WE}$ (в дадения случай може да бъде равно на 0, с други думи, ако даже едновременно с изменението на адреса сигналът $\overline{WE}$ се активира, записът на данните се осъществява по правилен адрес); t_{DH} – минимално време за задържане на записваните данни след дезактивирането на $\overline{WE}$ (в дадения случай то също може да бъде равно на 0); t_{WC} – минимално време, необходимо за осъществяването на еднократен запис.

Изучаването на фиг. 5.1 показва, че цялата информация около процесите на четене и запис в оперативна памет е стабилна по време на отминаващите фронтове съответно на $\overline{OE}$ и $\overline{WE}$. Следователно, ако по тези фронтове се стробира и запомни състоянието на адресните линии, линиите за данни и на управляващите сигнали, ще бъде получена цялата информация, необходима за интерпретиране на изпълняваната операция. За да се постигне това в една микропроцесорна система, изградена на базата на типичен 8-разреден микропроцесор, ще е необходимо да се запомнят състоянията на 16 адресни линии, 8 линии за данни и управляващите сигнали за четене и запис. Получават се общо около 27 до 30 линии, от които информацията трябва да се стробира и запомни. При това не са важни реалните форми на сигналите. Важни са само техните логически състояния в момента на стробирането и запомнянето им.

5.1. Структура на логически анализатор

Уредите, които извършват запомняне във времето на логическите състояния от значителен брой точки на цифрови системи и извършващи последваща обработка и визуализация на тези състояния, се наричат *логически анализатори*, а методът на такава работа – *логически анализ*. По същество логическият анализ е метод за наблюдаване на работата на апаратната част на електронна цифрова апаратура. Използването му се определя от особеностите при диагностика на сложни електронни цифрови и микропроцесорни системи и предполага реализирането на следните възможности:

– едновременно регистриране на логическите състояния в голям брой точки от схемата в продължителен времеви интервал;

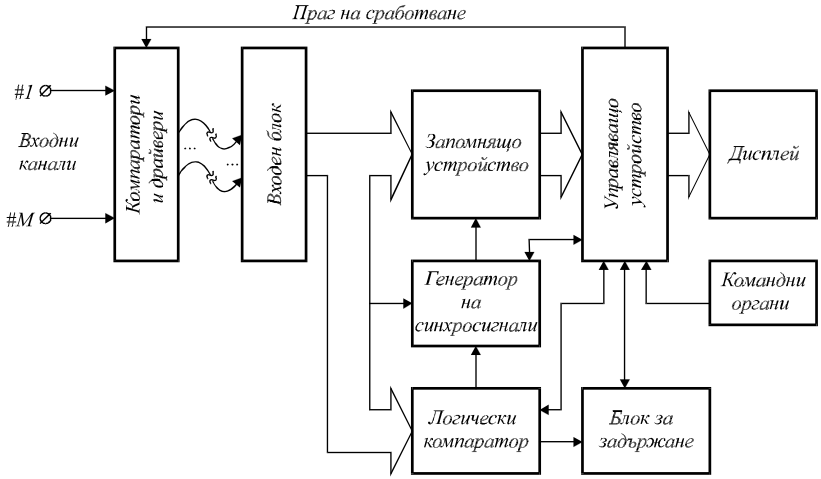
– регистриране на последователности от логически състояния при появата на редки, еднократни събития;

– регистриране на състоянията на контролираните точки за определен интервал от време, предшествуващ избрано от потребителя събитие;

– анализиране на резултатите и представянето им в удобен за оператора вид.

Логическият анализатор представлява контролно-измервателен уред, предназначен за събирането на данни от електронни цифрови устройства, обработката на тези данни и представянето им в удобен за оператора вид. Той работи независимо от изследваното устройство, без да се намесва в работата му. Абсолютно прозрачен е от гледна точка на потребителското устройство, което не

"усеща" присъствието му. Обобщената блокова схема на логически анализатор е представена на фиг. 5.2. Тя включва канални компаратори и драйвери, входен блок, оперативно запомнящо устройство, логически компаратор, генератор на синхросигнали, блок за задържане, управляващо устройство, дисплей и командни органи.



Фиг. 5.2. Обобщена блокова схема на логически анализатор

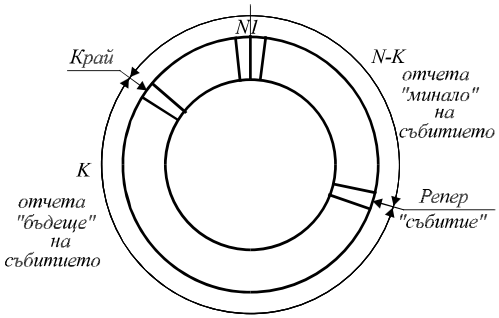
Каналните вериги на логическите анализатори, съдържащи компаратори и драйвери, не трябва да влияят върху работата на изследваното устройство. Затова те притежават високо входно съпротивление (около 1 $M\Omega$) и малък входен капацитет (от порядъка на 10 ÷ 25 pF). Входните компаратори извършват квантоване на сигналите по две нива – логическа *0* и логическа *1*, според зададения им праг на сработване. За намаляване на входния капацитет и индуктивност те се изнасят извън анализаторите и се оформят в отделен блок, разположен близо до изследваното устройство. В блока се включват драйвери за изпращане на дискретизираните сигнали към входния блок на логическия анализатор. Логическите анализатори имат от 8 до над 100 паралелни входни канала, обикновено разделени по 8 в група. Праговото ниво на сработване на входните компаратори се задава по групи от оператора. Като правило, когато се диагностицира микропроцесорна система, за анализ се вземат сигналите от микропроцесора, които в концентриран вид съдържат и отразяват работата на системата. Затова и повечето логически анализатори притежават специализирани сонди-клипсове, ориентирани към конкретни микропроцесори.

Входният блок поема сигналите от блока с каналните компаратори и

драйвери, като съгласува входовете си с параметрите на линиите, по които получава цифровата информация от драйверите. Този блок конфигурира и редуцира информацията за запис, като в него се определят подлежащите на запис в паметта входни канали.

Запомнящото устройство на логическия анализатор съхранява в себе си определен брой стойности от всеки канал – цифрови отчети. Неговата разредност, обем и бързодействие определят техническите характеристики на логическия анализатор – брой на информационните канали, дължина на логическите последователности и максимална тактова честота. Разредността на клетките на паметта е равна на броя на входните канали. При едно стробиране на *M* входни канала се получават *M* логически стойности, които се записват паралелно в *M*-разредна клетка от паметта.

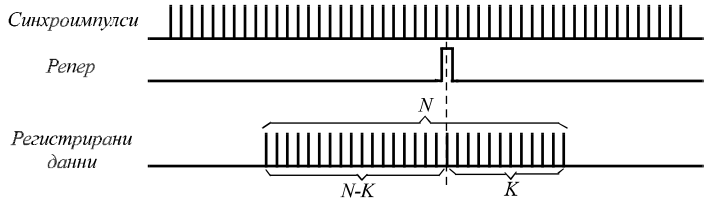
Броят на клетките в запомнящото устройство дефинира т.нар. "прозорец" на логическия анализатор и съответно големината на изследвания участък. Целенасоченото търсене на неизправност в електронно-цифрово устройство с помощта на логически анализатор е свързано с преглед и анализ на отделни участъци от целия информационен поток. Поставянето на анализаторния прозорец върху даден участък се извършва след като операторът дефинира участъка. За целта се избира характерно за този участък логическо събитие, наречено репер (някъде тригер). Това събитие се задава в логическия компаратор. Същевременно операторът задава в блока за задържане отстоянието на репера от края на участъка. След завършването на тази процедура логическият анализатор започва въвеждане на входната информация в запомнящото устройство. В случай на запълване на цялото запомнящо устройство записът продължава циклично от нулев адрес, а най-старите данни се изтриват.



Фиг. 5.3. Организацията на кръговата памет на логически анализатор.

По същество запомнящото устройство действа като кръгова памет. Тя е броячно адресируема, като след всеки запис на входната информация адресиращият брояч увеличава състоянието си с 1. Започвайки от адрес 0, адресирането продължава до последния адрес, след което броячът се препълва, нулира се и

адресирането продължава наново от нулев адрес. Щом настъпи зададеното логическо събитие (репера), логическият анализатор извършва още толкова записа, колкото е указано в блока за задържане, и спира въвеждането на входната информация.



Фиг. 5.4. Прозорец на логически анализатор.

Организацията на кръговата памет на един логически анализатор е показана на фиг. 5.3, а на фиг. 5.4 е показано състоянието на прозореца на логическия анализатор. Ако броят на клетките в запомнящото устройство е N, а в блока за задържане е указано, че записът трябва да спре след K синхронизиращи цикли, то прозорецът ще съдържа K отчета след репера и $N - K$ отчета преди репера. По такъв начин за анализиране е достъпно както миналото, така и бъдещето на събитието, избрано от оператора.

5.2. Синхронен и асинхронен режим на работа

Генераторът на синхросигналите на логическия анализатор изработва стробиращи импулси за запис на входната информация в запомнящото устройство. Тактовете за запис могат да постъпват от наблюдаваното устройство, а могат да бъдат изработвани и от самия логически анализатор. Режимът на стробиране, при който тактовите импулси постъпват от изследваното устройство, се нарича *синхронен режим* на работа, а логическият анализатор, работещ в този режим, се нарича *анализатор на логически състояния*.

Синхронният режим на работа е особено удачен при изследване на микропроцесорни системи, където интерес представлява състоянието на системата в определени моменти, докато преходите на сигналите не са от значение. Постъпващата информация трябва да се възприема от логическия анализатор по същия начин, по който я възприема и изследваната система, т.е. по същия синхросигнал или синхросигнали. Например при диагностика на процесорна система, изградена с микропроцесора СМ601, е удобно входните сигнали за логическия анализатор (адреси, данни и управляващи сигнали) да бъдат синхронизирани по спадащия фронт на системния такт *Ф2*. Подобно за микропроцесорни системи с MC68HC11, като синхронизиращ сигнал е удобно да се използува системният такт *E* (спадащия фронт). Малко по-различно е положението при микропроце-

сорни системи с едночиповия микрокомпютър i8031, където синхронизирането на сигналите трябва да става едновременно по отминаващите фронтове на сигналите $\overline{RD}$, $\overline{WR}$ и $\overline{PSEN}$.

В анализаторите на логически състояния е предвидена възможност за задаване на активния фронт на синхросигнала, по който се извършва стробирането на каналната информация. Важни параметри, които трябва да се имат предвид при избора на синхросигнала, са времето за установяване и времето на задържане на информацията. Първият параметър е времето, в което информацията трябва да е установена на каналите преди активния записващ фронт на тактовия сигнал, а вторият определя минималното време, през което информацията трябва да остане неизменна след постъпването на записващия фронт.

В някои случаи е необходимо стробирането да не се извършва по всеки синхронизиращ импулс. За указване на тези синхроимпулси, по които се извършва стробиране на каналната информация, се използува допълнителен входен сигнал, наречен квалификатор на такта. По желание на потребителя на него може да се задава активно ниво *0*, *1* или *X* (безразлично, т.е. не се използува). Като такъв квалификатор на такта може да се използува сигналът *VMA* при анализ на работата на микропроцесора СМ601. Ако потребителят не желае да извършва запис на информация при невалиден адрес, той може да включи сигнала *VMA* да изпълнява ролята на квалификатор на такта с активно ниво *1*.

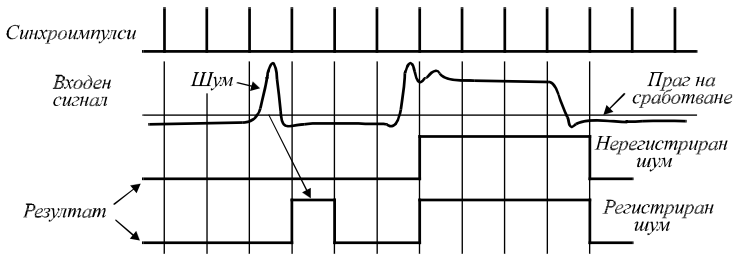
Разновидност на синхронния режим на работа на логическия анализатор е режимът на *многофазова синхронизация*. При този режим различните входни сигнали се стробират по няколко различни синхросигнала. Типичен пример за необходима многофазова синхронизация е процесът на анализиране състоянията на микропроцесорите, имащи мултиплексирани магистрали, като едни и същи линии служат за предаване на адреси и данни, а понякога и за управляващи сигнали. Така например, при логически анализ на работата на микропроцесора MC68HC11 е необходимо адресните сигнали да се стробират по спадащия фронт на сигнала за адресен строб *AS*, а информационните и управляващите сигнали – по спадащия фронт на сигнала за разрешение *E*.

С оглед улесняване на потребителя при диагностициране на електронни цифрови устройства с микропроцесорно управление производителите на логически анализатори предлагат специализирани адаптери за всеки микропроцесор. Адаптерите свързват изследвания микропроцесор с логическия анализатор. Вътре в тях твърдо е определено нивото на компариране на входните сигнали, извършено е разпределение на каналите на логическия анализатор по адреси, данни и управляващи сигнали, зададен е режимът на синхронизация и синхронизиращият сигнал (сигнали).

Режимът на стробиране в логическия анализатор, при който тактовите импулси се изработват вътре в самия логически анализатор, се нарича *асинхронен режим* на работа, а логическият анализатор, работещ в този режим се нарича

анализатор на времеви съотношения. Асинхронният режим на работа на логическите анализатори обикновено се използува при изследването на несинхронизирани сигнали в наблюдавания обект, при преходни процеси и др. Необходимо условие за правилния анализ на наблюдаваните явления е честотата на синхронизиране на логическия анализатор да бъде достатъчно по-висока от честотата на наблюдаваните сигнали. В такъв режим могат да работят само ограничен брой от входните канали – неповече от 16, като честотата на стробиране достига и надминава 200 *MHz*. Запомнената в асинхронен режим информация не представлява аналогов образ на анализираните сигнали. Както и при синхронния режим на работа, информацията се запомня и индицира във вид на идеализирани правоъгълни сигнали.

Анализаторите на времеви съотношения обикновено притежават две модификации на този режим на работа. Първата модификация е *функционален времеви анализ*, а втората – *параметричен времеви анализ*. Функционалният времеви анализ се прилага там, където е необходимо да се схване корелацията между някакви събития и сигнали, без да се търсят конкретните времеви измерения на тези събития. Например, когато се търси възникнало смущение в цифрова система, преди всичко е необходимо да се открие кога се получава то, без да се проявява интерес към неговата продължителност. Чрез този модифициран режим се откриват краткотрайни лъжливи сигнали, които възникват между два синхроимпулса и не могат да бъдат регистрирани в запомнящото устройство. Тази модификация се реализира чрез специални схемотехнични методи, които откриват нееднократни изменения на сигнала вътре в синхронизиращия период и възпроизвеждат импулса в следващия период. Модификацията е илюстрирана чрез показаните на фиг. 5.5 графики.



Фиг. 5.5. Регистриране на шум в логически анализатор.

При модификацията за параметричен времеви анализ е необходимо да се знаят точните времеви параметри на анализираните събития. В този случай честотата на вътрешния синхросигнал трябва да бъде значително по-голяма, отколкото при функционалния времеви анализ.

При микропроцесорните системи анализаторите на времеви съотношения се

използуват главно при изследване на входно-изходни канали за обмен на информация и на всички процеси, които са асинхронни по отношение на микропроцесорните операции.

5.3. Определяне на анализаторен прозорец

Логическият компаратор на анализатора има за задача да изработи репера при пристигане на предписаната му информация. Начините на определянето на репера представляват важна характеристика и отличителна особеност на логическия анализатор. Най-простият вид репер, използуван при всички логически анализатори, представлява комбинация от логическите състояния на входните сигнали (кодова дума). Всеки разред на кодовата дума може да приема три значения – нулево, единично и безразлично. Безразлично значение означава, че състоянието на този канал не се взема предвид при формиране на кодовата дума. Например при диагностика на микропроцесорна система в качество на кодова дума за репер може да бъде избран издаван от микропроцесора адрес, появата на определени данни или наличието на определени данни при зададен адрес и т.н.

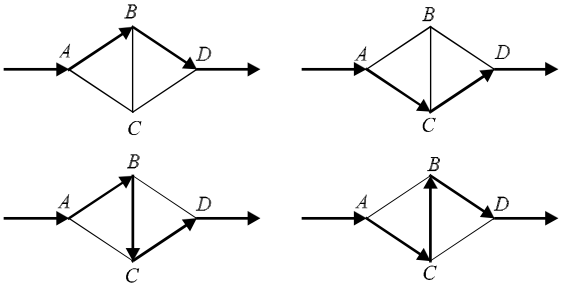
При формирането на репера могат да участвуват и други сигнали, които с активно ниво указват валидност или невалидност на кодовата дума. Такива сигнали се наричат квалификатори на репера и те не се стробират и запомнят в анализаторната памет. Например при анализ на микропроцесора СМ601 като квалификатор на репера може да се използува сигналът за достъпни шини – *ВА* с активно ниво *0*, като по този начин допълнително се определя, че репер ще се изработва само когато микропроцесорът е в активна работа (*ВА = 0*).

Разновидност на начина на изработване на репер по кодова дума е изготвянето на репер по *n*-кратно повторение на зададена кодова дума. Такъв репер с успех се използува при изследване на цикли в микропроцесорни програми, където е възможно да се анализира ситуацията след *n*-тото изпълнение на цикъла.

Значително по-съвършен начин на изработване на репер представлява т.нар. репер по комбинация от кодови думи. Във формирането на репера участвуват няколко кодови думи с указани логически релации между тях. Релациите биват "И", "ИЛИ", "СЛЕДВА" и др., които са оформени в стъпки за последователното им изпълнение. При този случай репер се изработва след появата на последната кодова дума, ако всички кодови думи са пристигнали в зададения релационен ред.

Този начин е подходящ при наличието на сложни, разклонени програми. Например при показания на фиг. 5.6 случай е невъзможно да се различат отделните разклонения на програмата при задаването само на една кодова дума за репер. Такова разделяне може да се осъществи, като се използува последователност от кодовите думи A→B→D, A→C→D, A→B→C→D и A→C→B→D. В релациите може да участва и неколкократна поява на кодова дума или комби-

нация от кодови думи, което улеснява анализирането на програми с цикли.



Фиг. 5.6. Различни възможности за преход между две състояния (A и D) на програма.

Най-сложен вид репер представлява т.нар. репер при несъвпадение. Този начин е предназначен за откриване на случайни смущения, които по традиционните начини не могат да бъдат идентифицирани. За неговото реализиране е необходимо логическият анализатор да притежава допълнително запомнящо устройство.

Първоначално, използувайки репер по кодова дума или комбинации от кодови думи, логическият анализатор записва еталонен анализаторен прозорец от данни в допълнителното запомнящо устройство. След това, преминавайки в режим на проверка по несъвпадение, анализаторът очаква данните според зададения репер и ги сравнява с данните, записани в допълнителното запомнящо устройство. Ако данните са различни, при първата констатирана разлика работата спира и се дава възможност на оператора да анализира тази разлика. Ако разлика няма, логическият анализатор наново се пуска сам за заснемане на нов прозорец и т.н.

От съществуващите начини за изработване на репер зависи удобството при работа с логическия анализатор и възможностите му за откриване на неизправности. В някои анализатори изработваният репер може да се използува за друг анализатор при съвместната им работа.

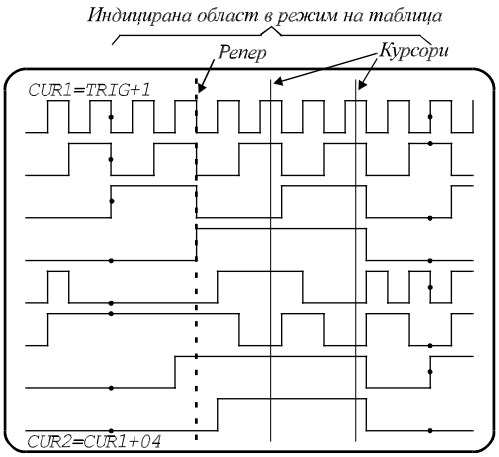
Блокът в логическият анализатор, който дава възможност около желаното събитие да се анализира неговото минало и бъдеще, е блокът за задържане. Той формира времето след получаването на репера, през което запомнящото устройство ще продължи да запомня каналната информация, преди да спре. Най-често това време на задържане се задава в брой тактове на синхронизиращия сигнал. В някои логически анализатори то може да се задава и в текущо параметрично време.

По време на работа логическият анализатор се намира в един от трите основни режима – настройка, регистрация и анализ-индикация. В режим на настройка операторът програмира за работа логическия анализатор. На първо място

се извършва подвеждането на сигналите за регистрация към входните канали на логическия анализатор и се задават тактовите сигнали и необходимите квалификатори. Специално за анализ на микропроцесорни системи съвременните логически анализатори се снабдяват с готови конфигурации за подвеждане на сигналите и автоматично задаване на тактове и квалификатори само с указване типа на микропроцесора. Настройката продължава със задаване на репера и задържането на записа след репера. В режим на регистрация анализаторът заснема данните според определения анализаторен прозорец.

5.4. Анализ и индикация

След като събере данните логическият анализатор преминава в режим на анализ и индикация. Изобразяването на заснетата и анализираната цифрова информация се извършва върху електронно-лъчева тръба. По-често управлението на тръбата е растерно, а по-рядко – векторно. Освен заснетата информация на екрана задължително се указва мястото на репера. Най-често указването е с TRIG.



Фиг. 5.7. Графично изобразяване на събрана от логически анализатор информация.

За увеличаване на функционалните възможности, за повишаване на ефективността на анализа на резултатите и за подобряване на нагледността на изобразяването, на екрана на логическия анализатор се извежда и много служебна информация. Извежда се информация за режима на регистрация, за формирането на репера, за закъснението след репера, за форматите на данните, а също и заглавни надписи на таблици, времеви диаграми и др. С помощта на електронни показалци (курсори) може да се определи взаимното положение на данните в логическите последователности, да се измерят времеви интервали и да се отбе-

лежат най-важните данни. Взаимното положение на курсорите и на репера се индицира на екрана в цифров вид като време в единици на периода на тактовия сигнал.

Една от формите на изобразяване на събраните данни представляват времевите диаграми. Едновременно на екрана се изобразява в графичен вид информация от няколко канала. Пример за такова изобразяване е даден на фиг. 5.7. Обикновено моментът на репера се означава на екрана с вертикална прекъсната линия. Операторът има възможност да изменя мащаба на времевата ос, броя на изобразяваните сигнали, да премества изобразявания участък и др. Този начин на изобразяване се използува най-вече при параметричен и при функционален времеви анализ.

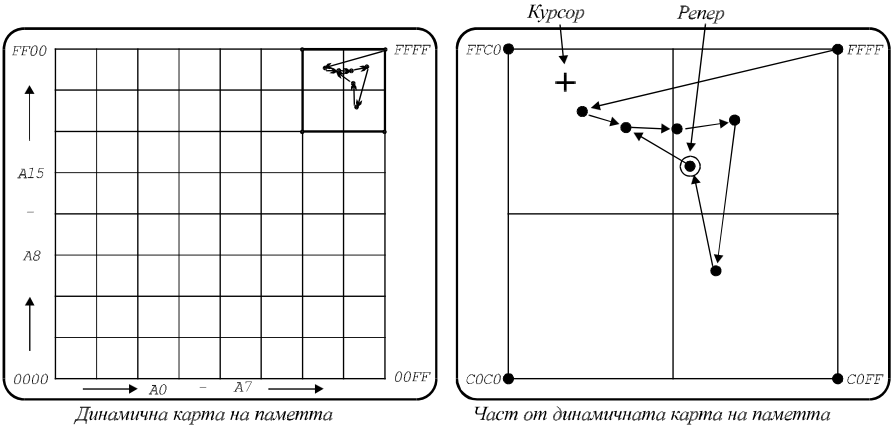
	BIN	HEX	ASCII
-4	0010 0100	24	$
-3	1010 0100	A4	.
-2	0110 0100	64	d
-1	1110 0110	F6	.
TRIG	0001 0110	16	SYN
+1	1001 1110	9E	.
+2	0101 1011	5B	[
+3	1101 1011	DB	.
+4	0011 1111	3B	;
+5	1011 0111	B7	.
+6	0111 0011	73	s
+7	1111 0011	F3	.
+8	0000 1100	0C	FF
+9	1000 0100	84	4
+10	0100 1000	48	H

Фиг. 5.8. Таблично изобразяване на събрана от логически анализатор информация.

Друг вид представяне на информацията, който се използува при логическите анализатори, е табличният. Последователността на данните, записани в запомнящото устройство, се изобразява на екрана чрез нули и единици. Последователно се изобразява съдържанието на клетките от паметта, като мястото на репера задължително е указано. За по-лесно възприемане на информацията двоичните цифри се групират по три или четири в хоризонтална посока. Данните също се групират по три или четири реда във вертикална посока. Понякога е по-удобно за оператора при представянето на данните да се използуват двоични, осмични и шестнадесетични числа или комбинация от тях. Например при диагностика на микропроцесорна система е удобно адресните и информационните сигнали да се представят на екрана в шестнадесетичен вид, а управляващите сигнали – в двоичен. Примерът от фиг. 5.8 илюстрира табличния начин на представяне на данните.

Съществуват логически анализатори, които възпроизвеждат данните във вид на карта на състоянията (графи на преходите). На екрана се възпроизвеждат 2^M точки, като всяка точка съответства на една комбинация на M входни сигнали,

т.е. всяка получена дума на тези M входни сигнали се индицира като светеща точка. Точките са свързани по такъв начин, че може да се наблюдава последователността на смяната на входящата в анализатора информация. Линията, съединяваща две точки, става по-ярка при приближаването си към новата точка, показвайки по такъв начин посоката на промяна на данните.



Фиг. 5.9. Изобразяване на динамичната карта на паметта на микропроцесорна система върху екрана на логически анализатор.

Съществен въпрос при съставянето на картата на състоянията е въпросът за избора на M канали, определящи картата. При проверка на микропроцесорни системи е удобно и препоръчително това да бъдат сигналите от адресните линии. На екрана ще се изобрази съвкупността от адресите, към които микропроцесорът се обръща при изпълнение на програмата. Тази съвкупност се нарича динамична карта на паметта на микропроцесора. При микропроцесор с 16 адресни линии състоянието на първите 8 определя абсцисната координата на светещата точка, а състоянието на старшите 8 – ординатата. Адрес `0000` се изобразява като точка в долния ляв ъгъл на екрана, а адрес `FFFF` – като точка в горния десен ъгъл. Тъй като цялото това поле е твърде голямо, в някои анализатори е предвидена възможност за отделяне на една част от него и "разгъването" му върху целия екран. Този метод на индициране дава възможност да се открие неправилно изпълнение на програмата от микропроцесора чрез сравняване с "образцова" динамична карта на паметта. Изобразяването е илюстрирано на фиг. 5.9.

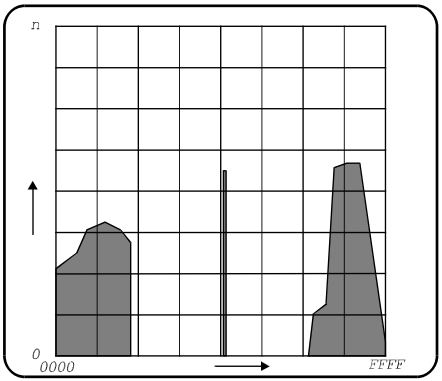
За логическия анализ на микропроцесорни системи, най-съвършен начин на изобразяване на събраната информация е дисасемблерският формат. Тъй като всеки микропроцесор притежава различна мнемоника на командите, в анализа-

тора трябва да бъде заложен дисасемблер за конкретния микропроцесор. Производителите на логически анализатори изработват специални програмни модули за индивидуална диагностика на устройства, изградени с даден микропроцесор. По потока на влизащата в логическия анализатор информация програмният модул реасемблира работната програма на наблюдавания микропроцесор и я изобразява на екрана в мнемонични кодове. Пример за такова изобразяване е показан на фиг. 5.10.

#	ADDR	DATA	VMA	R/W	DISASM
-6	6023	86	1	1	LDAA #$78
-5	6024	78	1	1	
-4	6025	B7	1	1	STAA $2002
-3	6026	20	1	1	
-2	6027	02	1	1	
-1	2002	FF	0	1	
TRIG	2002	78	1	0	
+1	6028	CE	1	1	LDX #$2042
+2	6029	20	1	1	
+3	602A	42	1	1	
+4	602B	20	1	1	BRA *-13
+5	602C	F3	1	1	
+6	602D	FF	0	1	
+7	6020	FF	0	1	

Фиг. 5.10. Реасемблирането на работна програма при анализ на събрана от микропроцесорна система информация в логически анализатор.

Някои логически анализатори имат възможност да възпроизведат данните във вид на честота на появяванията на събитията (статистика на постъпващите думи). При този начин абсцисната ос на екрана изобразява събитията (кода на постъпващите думи), а ординатната ос – броя на появяванията на всяка дума.



Фиг. 5.11. Изобразяване на статистика на постъпващи думи върху екрана на логически анализатор.

Съществен въпрос при определянето на думите е въпросът за избора им. При проверка на микропроцесорни системи удобно и препоръчително е това да

бъде активният адрес. При микропроцесор с 16 адресни линии адрес `0000` ще съответствува на най-лявата точка на абсцисната ос, а адрес `FFFF` – на най-дясната. Изобразяването е илюстрирано на фиг. 5.11. При това изобразяване би могло да се идентифицират обръщенията на микропроцесора към програмата, към данните и към периферията.

В някои логически анализатори има заложени и други начини на изобразяване на резултатите от анализа – например преброяване на събития, използуване на формати със специални кодове и др., но тяхното използуване е силно ограничено.

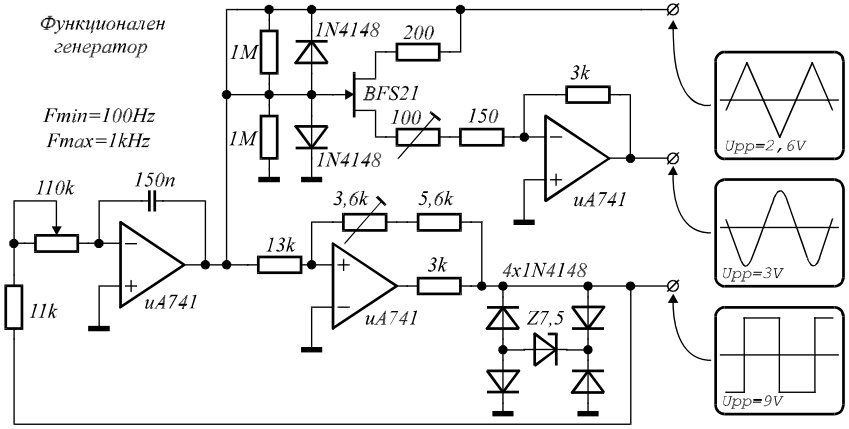
Наличието на значителни функционални възможности при съвременните логически анализатори обуславя сложно управление и необходимост от висококвалифициран оператор. Работата с логическите анализатори се извършва с клавиатура, а операторът получава обратна информация, наставления, списък на командите и друга информация чрез дисплея на анализатора. Задължително управлението на логическите анализатори вече се осъществява от микропроцесор, осигуряващ функциите на самоконтрол, изчисления, анализ и изобразяване на данните.

Съществуват логически анализатори, които са изградени да поемат информацията не от изводите на микропроцесора, а от системната магистрала на изследваната система. Те са снабдени със специален преходник за включването им към стандартен куплунг на системната магистрала. Такива анализатори се изграждат най-вече за работа по стандартизирани системни магистрали на многопроцесорни системи, като например VME-BUS, MULTIBUS и др., и се наричат *магистрални анализатори*.

В последно време се появиха специализирани апаратни модули, включвани към стандартни персонални микрокомпютри, които с помощта на подходящо програмно осигуряване превръщат персоналния микрокомпютър в логически анализатор (комплексни анализатори).

6. СИГНАТУРЕН АНАЛИЗ

Естеството на аналоговите системи позволява те да бъдат проверявани чрез подаването на подходящи тестови въздействия и наблюдаване на сигналите с осцилоскоп в определени контролни точки. За сравняване се използуват принципни схеми с ясно отразени стойности на напреженията и осцилограми в контролните точки. Една такава аналогова схема е показана на фиг. 6.1. Сравнявайки реално получените сигнали с осцилограмите, нанесени в схемата, специалистът може да определи неизправния блок в системата, дори и да не е запознат подробно с нейната работа.



Фиг. 6.1. Използуване на осцилограми при аналогови схеми.

Този метод за сравняване на осцилограми обаче не може да се използува при цифрови системи, тъй като всички двоични сигнали на екрана на осцилоскопа изглеждат почти еднакво. Потоците от цифрова информация трябва да бъдат регистрирани и представени на оператора в подходящ за възприемане вид. За тази именно цел е създаден т.нар. сигнатурен анализатор. Подобно на осцилоскопа, той възприема логическите сигнали в точки от цифровата система, но вместо да възпроизведе формата им, изработва ключови кодове – *сигнатури*, съответствуващи на информационните потоци.

6.1. Сигнатура

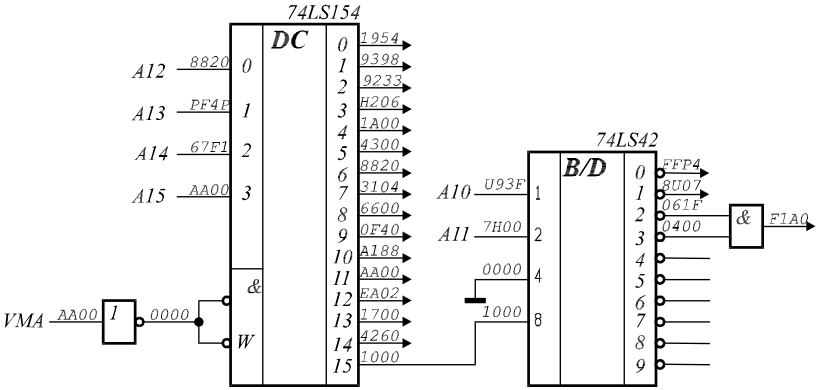
Сигнатурата представлява свита информация за логическите сигнали в определена точка от цифровата система и е характерен "подпис" на тази точка. Методът, при който се извършва сравняването на сигнатурите в контролните

точки на две системи, едната от които е изправна (образцова) с цел диагностика на другата система, се нарича *сигнатурен анализ*. При него се изхожда от предпоставката, че ако входните сигнатури на един блок са правилни, а изходните – не, мястото на неизправността е локализирано в този блок. Като критерии за правилни или неправилни сигнатури служат сигнатурите в аналогичните точки от изправната електронна цифрова система.

При снемането на сигнатурите се спазват двата основни принципа на сигнатурния анализ:

– снемането се извършва за определен отрязък от времето (времеви прозорец), еднакъв за диагностицираната и еталонната система;

– вътре във времевия прозорец снемането се извършва през определени интервали от време (отчети), еднакви и за двете системи.



Фиг. 6.2. Нанасяне на сигнатурите при изводите на цифровите схеми и каталогизирането им.

Обикновено сигналите в контролните точки на еталонната система са снети предварително при специални условия и се наричат *образцови сигнатури*. За удобство при работата те се нанасят на принципните схеми на цифровите системи, както примерно това е показано на фиг. 6.2. Снетите образцови сигнатури се записват в сервизен каталог заедно с условията, при които за заснети. Последното се изисква от принципите на сигнатурния анализ.

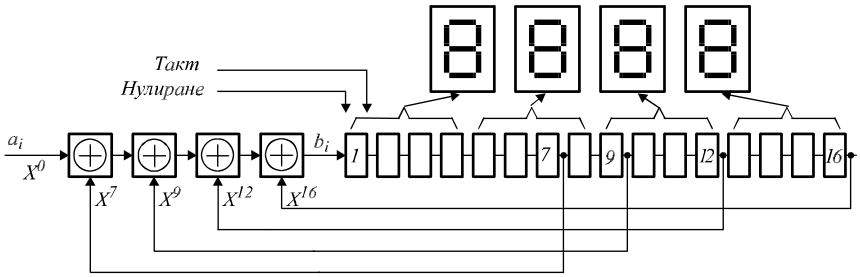
Отчетите, снети във времевия прозорец от дадена контролна точка на системата, представляват последователност от логически нули и единици, т.е. поток от битове. Началният и крайният бит от потока съответстват на началото и на края на измервателния интервал. Потокът от битовете се въвежда в сигнатурния анализатор в последователността, в която се формира. Дължината на потока може да бъде от няколко бита до няколко милиона бита. Поради тази причина в сигнатурните анализатори се извършва свиване на постъпващата диаг-

ностична информация.

Съществуват много начини за свиване на последователност от двоични данни, като преброяването на логическите преходи, преобразуване броя на логическите единици за определяне на контролни суми и дори на ентропията като мяра на информацията. Сравнителният анализ на различните методи е показал, че най-ефективен е методът, основаващ се на преобразуване чрез преместващ регистър с линейни обратни връзки. Схемите, прилагащи този метод, принадлежат към т.нар. генератори на псевдослучайни последователности.

При използването на 16-разреден преместващ регистър съществуват 2048 различни начина за реализирането на обратните връзки, удовлетворяващи определени критерии. За целите на тестването на схеми и устройства се предпочита такъв начин на взимането на обратните връзки, при който грешките се разпределят максимално. По тази причина не се препоръчва да се взимат обратни връзки през 4 или 8 разреда, тъй като те съответстват на разредности на микропроцесори. Фирмата Hewlett-Packard е предложила обратните връзки да се формират по полиноминалния израз $L(X) = X^0 + X^7 + X^9 + X^{12} + X^{16}$, който съответствува на характеристичен (пораждащ) полином на генератора $G(X) = X^{16} + X^9 + X^7 + X^4 + 1$.

Принципът на действието на такова свиване е показано на фиг. 6.3. Сигнатурата се формира от схема, съдържаща 16-разреден преместващ регистър, суматор по modulo 2, изграден с елементи EXOR (изключващо ИЛИ), и верига за обратна връзка от някои тригери на преместващия регистър към суматора по modulo 2. Това е линейна последователностна схема, тъй като суматорът по modulo 2 придава една и съща тежест на всеки входящ бит.



Фиг. 6.3. Свиване на входната последователност в 16-разреден преместващ регистър за получаване на сигнатура.

В преместващия регистър допълнително влизат тактов сигнал и сигнал за нулиране. Те се изработват от управляващо устройство. Благодарение на обратната връзка, независимо от дължината на входната информация, се получава само 16-битово число, което се представя чрез четири шестнадесетични цифри.

Това число представлява действителната сигнатура на изследваната точка.

Използуваният от сигнатурните анализатори шестнадесетичен код е различен от обикновения. Вместо числата **0**, **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **A**, **B**, **C**, **D**, **E** и **F** се използуват **0**, **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **A**, **C**, **F**, **H**, **P** и **U**. Този начин на изобразяване на числата е въведен за пръв път от Hewlett-Packard и има за цел облекчаване на четливостта при използуването на седемсегментни индикатори. Получаването на сигнатурата се извършва съгласно дадената по-долу формула.

Нека входната последователност съдържа *n* бита: a_1 , a_2 , a_3 , . . . , a_{n-1} , a_n (a_i може да заема числови стойности само *0* и *1*). На тази последователност съответствува друга последователност от *n* бита – b_1 , b_2 , b_3 , . . . , b_{n-1} , b_n след суматора по modulo 2. Според прокараните обратни връзки първите седем бита на двете последователности ще са еднакви, т.е. $b_1 = a_1$, $b_2 = a_2$, $b_3 = a_3$, $b_4 = a_4$, $b_5 = a_5$, $b_6 = a_6$ и $b_7 = a_7$. Всички останали битове b_i ще се определят от формулата

$$b_i = \begin{cases} a_i & при \quad b_{i-16} \oplus b_{i-12} \oplus b_{i-9} \oplus b_{i-7} = 0 \\ \\ \overline{a_i} & при \quad b_{i-16} \oplus b_{i-12} \oplus b_{i-9} \oplus b_{i-7} = 1 \end{cases} , \qquad (6.1)$$

като $8 < i < n$. Всички битове b_{i-k} , за които в индекса се получава отрицателно число или нула ($i < k$), се премахват. Двоичният код на сигнатурата представлява последните 16 бита от последователността b_1 , b_2 , b_3 , . . . , b_{n-1} , b_n .

От математична гледна точка получаването на сигнатура в преместващ регистър с линейни обратни връзки представлява формирането на остатъка от деленето на входния двоичен полином $A(X)$ на пораждащия полином $G(X)$. При деленето на тези два полинома се получава частно $Q(X)$ и остатък $R(X)$, както е показано в следващото уравнение:

$$\frac{A(X)}{G(X)} = Q(X) + \frac{R(X)}{G(X)} \qquad . \qquad (6.2)$$

Същото уравнение може да се представи и във вида:

$$\frac{A(X) - R(X)}{G(X)} = Q(X) \qquad . \qquad (6.3)$$

Входният полином $A(X)$ представлява полиноминалното представяне на входната двоична поредица. Например двоичната поредица `11001001` може да се представи във вида $1.2^7 + 1.2^6 + 0.2^5 + 0.2^4 + 1.2^3 + 0.2^2 + 0.2^1 + 1.2^0$. Когато конкретната основа *2* се замести с обобщената *X*, изразът се опростява на $A(X) = X^7 + X^6 + X^3 + 1$.

Характеристичният полином на генератора $G(X)$ се нарича още пораждащ полином, а фактическите точки, откъдето се взимат обратните връзки, се изчисляват от неговата инверсия. Инверсният (обратният) израз на неговата стойност

се определя чрез изваждане на всеки член в характеристичния полином на генератора от X^{16}, чрез което се получава полиноминалния израз на обратните връзки $L(X)$.

Свиването на данните в преместващия регистър не може да мине без загуба на информация. След получаването на сигнатурата в преместващия регистър се намира само остатъкът от делението $R(X)$, докато частното $Q(X)$ се губи. Преобразуването е еднопосочно, което означава, че от определена входна последователност може да се получи точно една сигнатура, докато от една сигнатура не може да се възстанови входната последователност. Затова при сигнатурния анализ се поставя въпросът за достоверност на анализа или за вероятност за откриване на грешка. При сигнатурния анализ могат да се получат само т.нар. фалшиво-отрицателни резултати, т.е. от грешна входна поредица да се получи вярна сигнатура. Получаването на фалшиво-положителни резултати (от вярна поредица да се генерира грешна сигнатура) е знак за неправилна работа на сигнатурния анализатор.

В известните от литературата анализи се предлага следната формула за вероятността за откриване на грешка при сигнатурен анализ с 16-битов преместващ регистър:

$$P = 1 - \frac{2^{m-16} - 1}{2^m - 1} \quad , \tag{6.4}$$

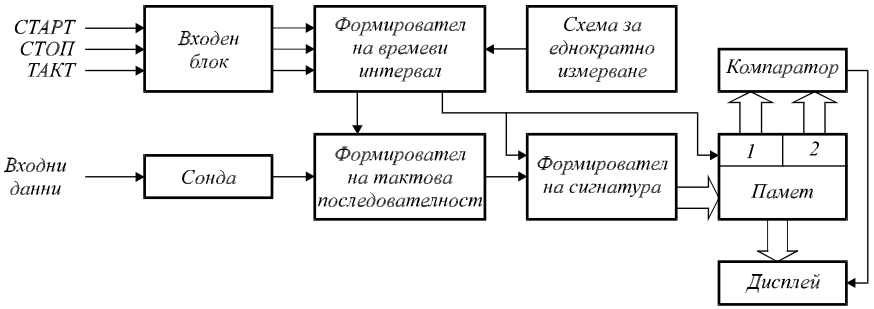
където m е дължината на потока от битове, постъпващи в анализатора.

При дължина $m < 16$ вероятността за откриване на грешка е $P = 1$. При дължина $m > 16$ и същевременно при наличие на повече от един сгрешен бит в информационния поток вероятността за откриване на грешка е не по-малка от 0,99998. Ако при последното условие сгрешеният бит е само един, вероятността за откриване на грешка е $P = 1$. Общата достоверност на сигнатурния анализ се определя на 99,998 %.

6.2. Сигнатурен анализатор

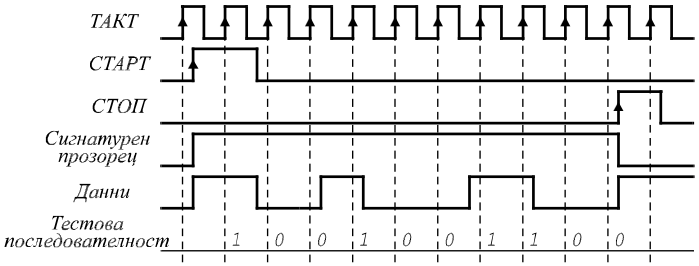
Общата блокова схема на един сигнатурен анализатор има представения на фиг. 6.4 вид. От изпитваното устройство, през входен блок в анализатора постъпват трите управляващи сигнала – СТАРТ, СТОП и синхронизиращи импулси ТАКТ. Във формирователя на времеви интервал се изработва сигнатурният прозорец за управление на преместващия регистър. Входящите данни се възприемат от сонда обикновено изнесена извън сигнатурния анализатор. Тя съдържа компаратор с управляем праг на сработване и притежава високо входно съпротивление и малък входен капацитет, за да не влияе на заснемания сигнал. Във формирователя на тестовата последователност данните се стробират от синхронизиращите импулси. Там се извършва тяхното превръщане в последователност от нули и единици. Синхронизиращите импулси тактуват работата на целия

анализатор, включително и на преместващия регистър.



Фиг. 6.4. Блокова схема на сигнатурен анализатор.

Образуваната тестова последователност от логически нули и единици постъпва в блока за формиране на сигнатурата, съдържащ преместващия регистър. Получаването на тестова последователност е показано на фиг. 6.5. Дисплеят на сигнатурния анализатор се представя от четири седемсегментни индикатора и допълнителни индикаторни светодиоди.

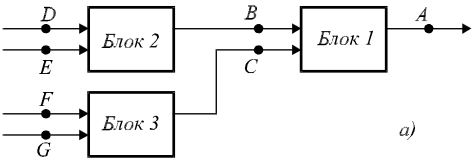


Фиг. 6.5. Формиране на тестова последователност в сигнатурен анализатор.

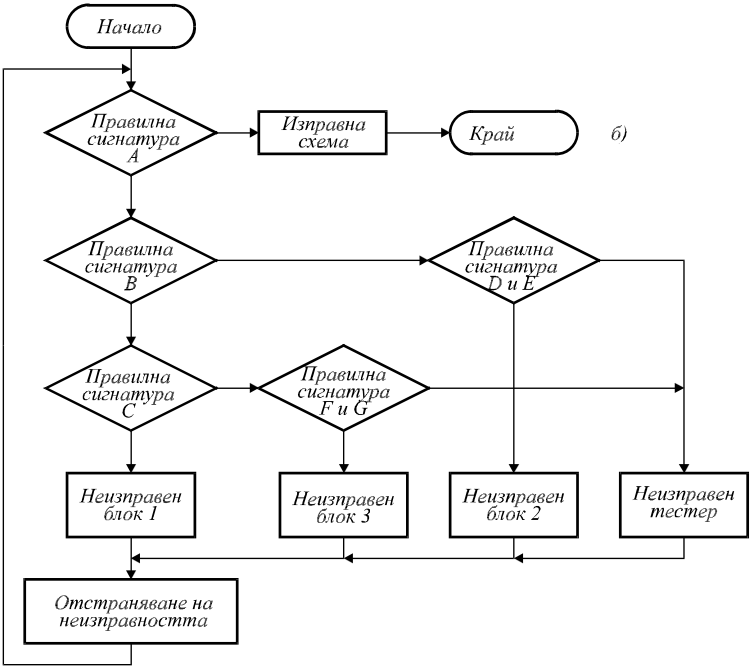
В сигнатурния анализатор са предвидени два блока за запомняне на получената сигнатура. Всяко измерване предизвиква запомняне на сигнатурата в блок 1, докато в блок 2 се прехвърля снетата от предишното измерване сигнатура. Логическият компаратор сравнява двете сигнатури от двете поредни измервания и при несъвпадението им издава върху дисплея съобщение за нестабилна сигнатура. То дава информация за случайни нарушения в работата на изпитваното устройство. Обикновено в сигнатурните анализатори се предвижда и режим на еднократни измервания.

Проверката на електронно-цифрово устройство със сигнатурен анализатор се подчинява на строга последователност на снемане на сигнатурите. Това е

илюстрирано с примерното устройство на фиг. 6.6 а, за което се прилага алгоритъм на проверка, даден на фиг. 6.6 б.



а)

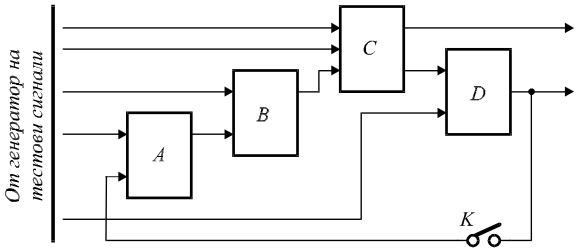


б)

Фиг. 6.6. Примерно устройство (а), за което е приложен алгоритъм на проверка със сигнатурен анализатор (б).

Масовото приложение на сигнатурния анализ е в областта на сервизната диагностика. Изделията, за които ще се прилага сигнатурен анализ, още в процеса на разработка се снабдяват с вътрешносхемни средства, позволяващи ефективно и лесно провеждане на анализа. На първо място, това са възможностите за разкъсване на съществуващите обратни връзки в режим на проверка. Ако такава възможност не съществува, откриването на повреден елемент във верига с обратна връзка не може да се извърши със сигнатурен анализ.

Примерът от фиг. 6.7 потвърждава това твърдение. При неизправност на блок B неизправни сигнатури ще се появят в неговия изход, в изхода на блок C и в изхода на блок D. При затворена обратна връзка не може да се определи дали повредата е в блок B, C или D. Ако обаче в режим на проверка обратната връзка може да се разкъса, сигнатурният анализ безпрепятствено ще укаже повредения блок B. На практика не е необходимо да се разкъсват вериги за обратна връзка, обхващащи малък брой елементи. В такива случаи е достатъчно сигнатурният анализатор да локализира повредата в дадена верига, а проверката да продължи с други, по-прости методи.



Фиг. 6.7. Разкъсване на обратна връзка при сигнатурен анализ на устройство.

Второ важно условие за удобно провеждане на сигнатурния анализ е вграждането в разработваното устройство на схема за изработване на сигналите СТАРТ, СТОП и ТАКТ. По тези сигнали в сигнатурния анализатор се изработва времевият сигнатурен прозорец.

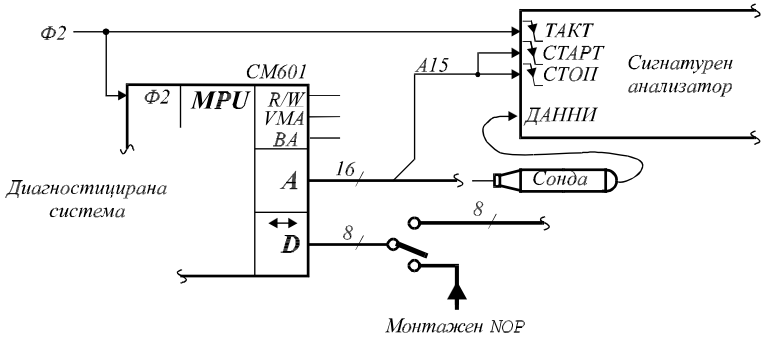
6.3. Провеждане на сигнатурен анализ

При диагностицирането на микропроцесорна система със сигнатурен анализатор с успех може да се използува режимът на свободно броене на микропроцесора. Шините за данни към системата се прекъсват и на микропроцесора се подава кодът на празната инструкция.

При този режим адресните линии се превръщат в изходи на двоичен брояч, като състоянията им циклично си повтарят. Тази цикличност идеално подхожда на сигнатурния анализ. Най-старшата адресна линия стои в логическа *0* за времето, когато микропроцесорът обхожда първата половина от адресното си поле, а в *1* – за втората половина. Така между два последователни еднакви фронта на най-старшата адресна линия се вмества едно пълно обхождане на микропроцесорното адресно поле. Сигналът от тази линия може да се използува като сигнал СТАРТ и СТОП на сигнатурния анализатор. В режим на свободно броене микропроцесорът изпълнява само операция четене.

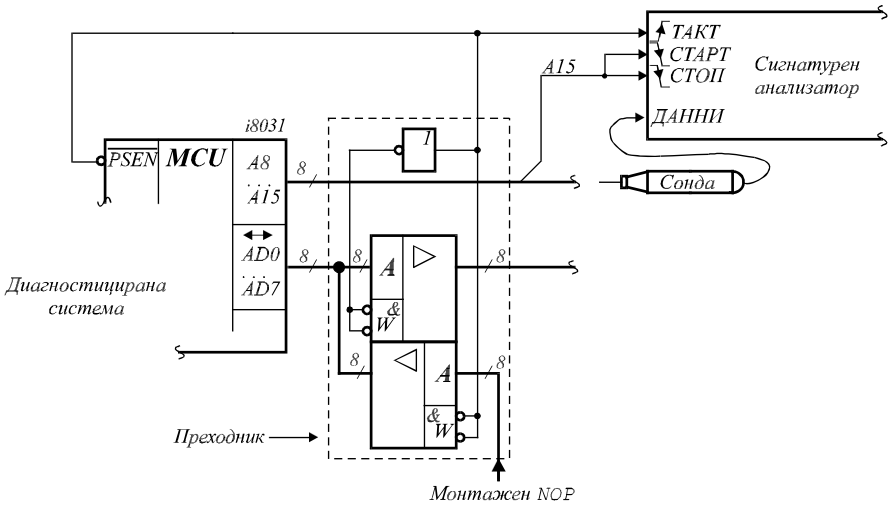
На фиг. 6.8 е показано заснемането на сигнатура от микропроцесорна система с микропроцесор СМ601. Синхронизиращ сигнал ТАКТ за логическия

анализатор се взема от системния такт на микропроцесора *Ф2* или *DBE* (*E*), като сигнатурният анализатор се синхронизира по спадащия фронт на такта.



Фиг. 6.8. Снемане на сигнатури от процесора на процесорна система със CM601.

При микропроцесорна система с 68HC11, синхронизирането се извършва по спадащия фронт на *E*, при i8086 – по нарастващия фронт на $\overline{RD}$, т.е. избира се сигнал, показващ валидно обръщение към програмното адресно пространство.



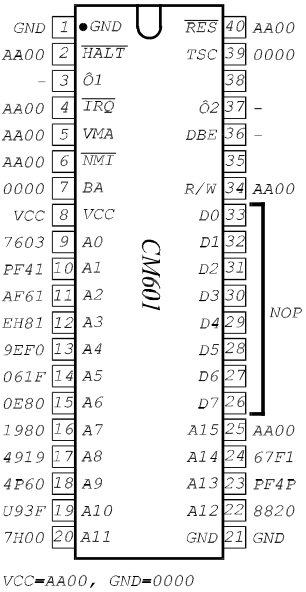
Фиг. 6.9. Снемане на сигнатури от поцесора на процесорна система с i8031.

Фиг. 6.9 показва заснемането на сигнатура от микропроцесорна система с едночиповия микрокомпютър i8031, където сигнатурният анализатор се синх-

ронизира по нарастващия фронт на сигнала за четене на програмната памет $\overline{PSEN}$.

Сигнатурите на захранващия проводник и на масата се наричат "характеристични сигнатури". По тях може да се извърши проверка на работоспособността на сигнатурния анализатор. Когато се снема сигнатурата на масата, като входни данни постъпват само логически нули. След изтичане на времеви прозорец състоянието на преместващия регистър остава нулево, тъй като само входяща единица би могла да го промени. Следователно масата дава винаги сигнатура 0000.

При снемане сигнатурата на захранването като данни ще влизат само логически единици. Получената сигнатура ще зависи от броя на отчетите в прозореца и ще бъде постоянна стойност за така подбраните СТАРТ, СТОП и ТАКТ.



Фиг. 6.10. Каталогизиране на образцови сигнатури за микропроцесор СМ601.

Когато от контролни точки в микропроцесорната система се получат сигнатури еднакви с тези на захранване или маса, може да се предположи, че е налице късо съединение със съответните проводници. С помощта на други уреди е необходимо да се установи дали действително има късо съединение, или сигналите в тези точки са статични за времето на сигнатурния прозорец. Например с логически пробник могат да се открият краткотрайни импулси в такива точки, които са извън сигнатурния прозорец и които свидетелстват, че няма късо съединение със захранващия проводник или масата.

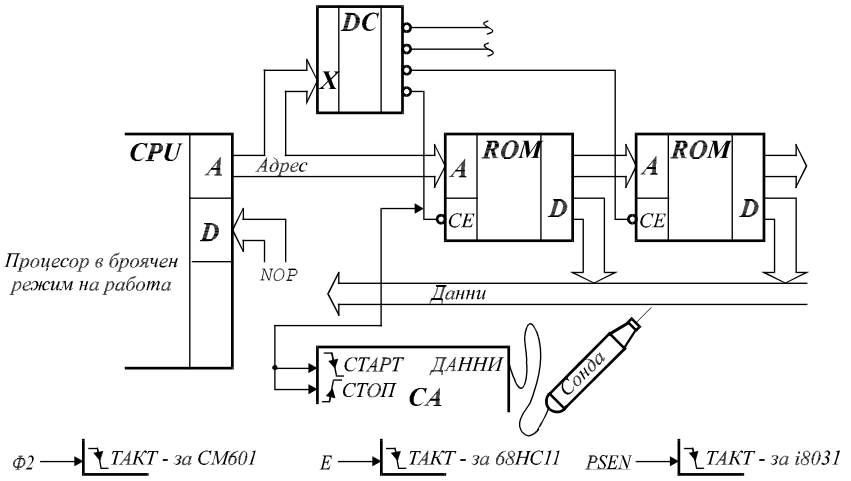
В режим на свободно броене се заснемат сигнатурите на адресните линии при всички схеми, свързани с адресната магистрала. Тези сигнатури се сравняват с нанесените в документацията сигнатури. Там където информацията е безсмислена (например на откачените линии за данни), се поставя знака “X” за безразлична сигнатура. Твърдите сигнатури на захранване и маса се означават с *VCC* и *GND*. На фиг. 6.10 е отразен начинът на задаване на сигнатури и условията за снемането им.

Режимът на свободно броене на микропроцесора може да се използува и за диагностициране на постоянните памети. Обхождайки цялото адресно поле,

микропроцесорът последователно изчита и областта, в която е разположена постоянната памет. Постановката на измерването е показана на фиг. 6.11. Сигналът за избор на постоянната памет се използува за формиране на сигнатурния прозорец. Синхронизиращият такт за сигнатурния анализатор е същият както при снемане на сигнатурите от микропроцесора. Така може да се извърши проверка на съдържащите се в постоянната памет данни.

Начинът на снемане на сигнатури чрез поставянето на микропроцесора в режим на свободно броене се нарича *автоматичен*.

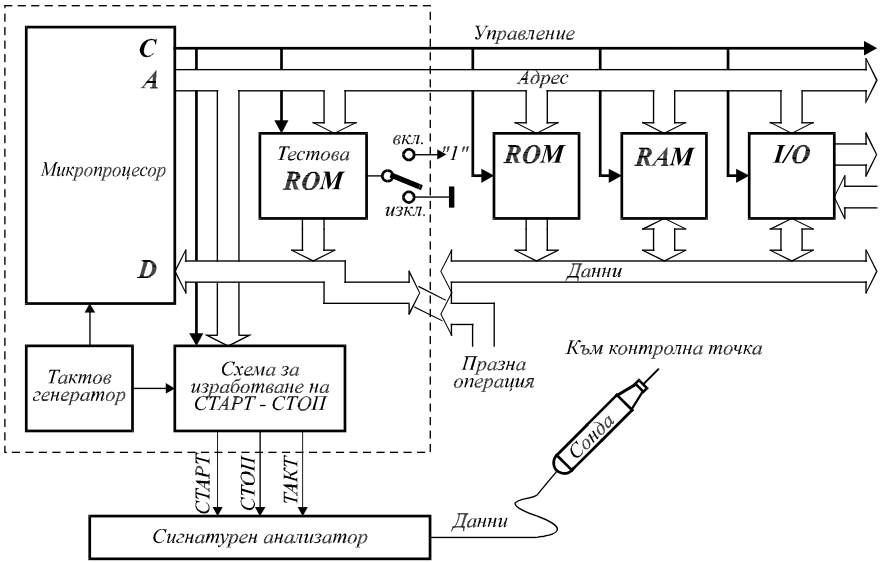
Проверката на оперативната памет, на периферните устройства и др. не може да се извърши само с използуването на режима на свободно броене на микропроцесора. За тях е необходимо да се напишат и изпълнят специални тестови програми. Всяка от тях е предназначена за проверка на една част от системата. Те се програмират в постоянна памет, която може да бъде отделна от постоянната памет, съдържаща работната програма. Пакетът тестови програми се организира като цикъл, който периодично се изпълнява при активиране на тестването. В началото на тестовия цикъл се формира сигнал СТАРТ за сигнатурния анализатор, а в края на цикъла – сигнал СТОП. Този начин на снемане на сигнатури се нарича *програмноуправляем*.



Фиг. 6.11. Снемане на образцови сигнатури от постоянната памет на микропроцесорна система.

Блоковата схема на пригодена за сигнатурен анализ микропроцесорна система е дадена на фиг. 6.12. Микропроцесорната система като цяло е обхванатa от обратна връзка по програмен път. На първо място, трябва да съществува възможност за нейното разкъсване и превключване на линиите за данни в положението тест, подаващо кода на празната инструкция към микропроцесора, за

да може той да се поставя в режим на свободно броене. На второ място, в състава на микропроцесорната система е включена постоянна памет с тестови програми за заснемане на сигнатури. На трето място е включен блок за изработване на сигналите СТАРТ, СТОП и ТАКТ. Тактовият генератор, микропроцесорът, тестовата памет и схемата за изработване на управляващите сигнали СТАРТ и СТОП образуват ядрото на микропроцесорната система. Сигнатурният анализ върху цялата микропроцесорна система се извършва под управлението на сигнали, генерирани от ядрото.



Фиг. 6.12. Блоковата схема на пригодена за сигнатурен анализ микропроцесорна система.

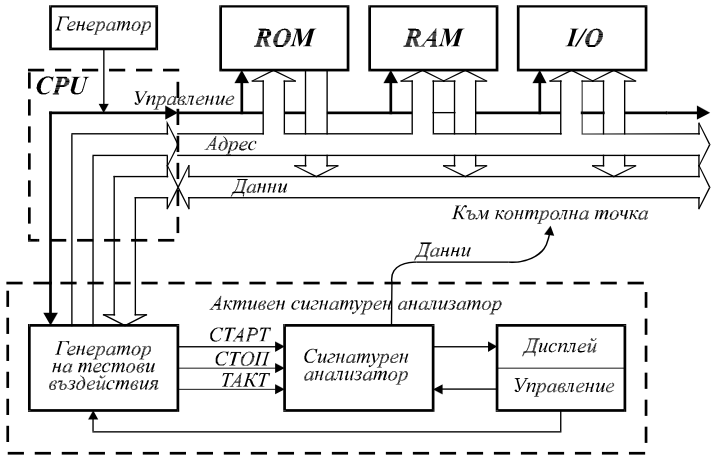
При неизправност на ядрото, за което свидетелствува неправилна сигнатура на адресните линии или на управляващ сигнал, се проверява преди всичко тактовият генератор. След него внимание се съсредоточава върху схемата за генериране на управляващите сигнали СТАРТ и СТОП. Най-лесно това се осъществява, като на сигнатурния анализатор се подаде за заснемане характеристичната сигнатура на захранването. Получаването на правилна сигнатура свидетелствува за правилността на сигналите СТАРТ и СТОП.

Локализирането на неизправността в микропроцесора или в тестовата памет се извършва чрез използуване на режима на свободно броене на микропроцесора. Неправилна сигнатура на някоя от адресните или управляващите линии ще укаже неизправност в микропроцесора, а неправилна сигнатура в информационните линии на тестовата памет – в самата нея. Ако ядрото е изправно, откри-

ването на неизправности в микропроцесорната система се извършва по описания по-горе начин.

В последно време за удобно приложение на сигнатурния анализ специално за микропроцесорни системи се появиха сигнатурни анализатори, които сами генерират тестовите въздействия върху изпитваното устройство. Те получиха наименованието "активни сигнатурни анализатори". Включването им към диагностицираната система се извършва през микропроцесорния цокъл.

Схемата на изпитване с помощта на активен сигнатурен анализатор е показана на фиг. 6.13. Отпада необходимостта от предвиждане на допълнителни схеми за сигнатурна тестопригодност. Отпада и проблемът за разкъсване на програмната обратна връзка. Този подход открива широки възможности за автоматизиране на изпитанията. В повечето случаи не се налага използването на друг кабел за връзка, освен за микропроцесорния цокъл. Активният сигнатурен анализатор съдържа в себе си гнездо, където се премества микропроцесорът на тестваното устройство. От тестваната система се получават синхронизиращите импулси, а управляващите импулси СТАРТ и СТОП се изработват в самия анализатор. Сигнатурният анализатор съдържа и тестова постоянна памет за реализиране на програмно управлявана проверка.



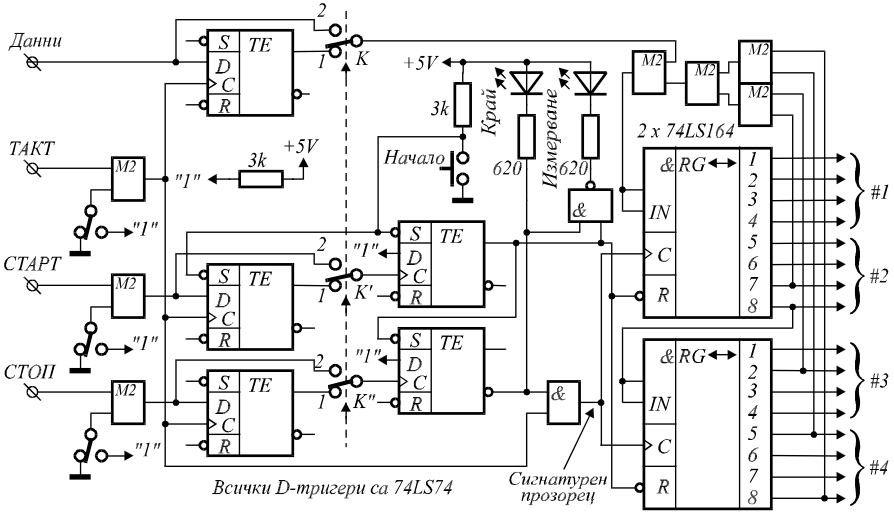
Фиг. 6.13. Свързване и тестване на микропроцесорна система с активен сигнатурен анализатор.

Електрическото и логическото свързване на активния сигнатурен анализатор към диагностицираната микропроцесорна система се извършва през адаптерен блок, различен за всеки вид микропроцесор. Съвременните сигнатурни анализатори се управляват от собствен микропроцесор и общуването с оператора се извършва чрез клавиатура и електроннолъчев дисплей. Появиха се и специални

модули, които, включени към персонален микрокомпютър с подходящо програмно осигуряване, го превръщат в сигнатурен анализатор.

6.4. Схемотехника на сигнатурен анализатор

На фиг. 6.14 е показана проста схема на сигнатурен анализатор. Тя работи с TTL нива на входящите сигнали. Шестнадесетразредният преместващ регистър е реализиран с две интегрални схеми 74LS164, а обратната връзка – с двувходови елементи XOR 74LS86. Възможността за задаване на активен фронт на сигналите СТАРТ, СТОП и ТАКТ се осигурява от елементи XOR, използувани като управляеми инвертори. Когато управлението е логическа *0*, схемата работи като повторител (активен е нарастващият фронт на преминаващия сигнал); когато управлението е *1*, схемата работи като инвертор (активен е спадащият фронт).

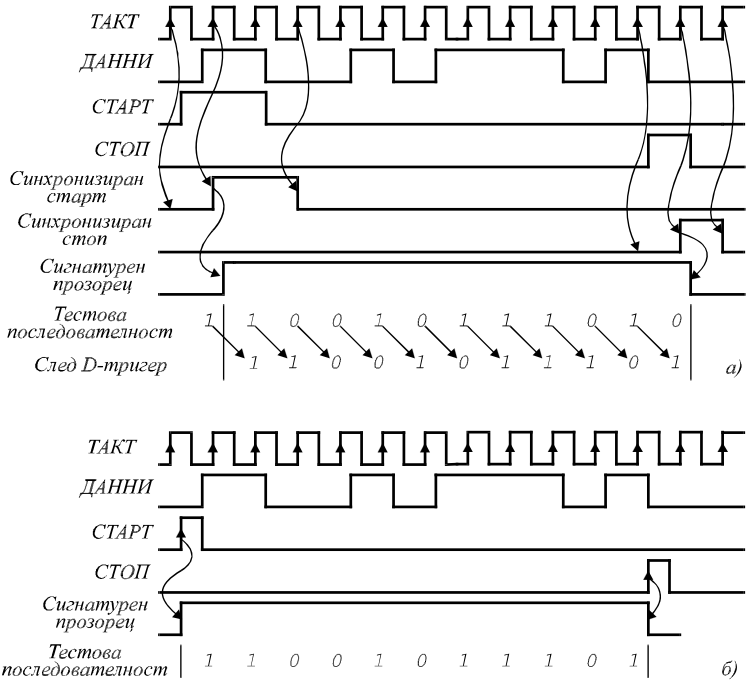


Фиг. 6.14. Принципна схема на формировател на сигнатура.

Началото на работата на анализатора се дава от оператора чрез бутон, който нулира преместващия регистър и поставя системата за управление в начално състояние, очаквайки идването на пусков импулс. След неговото идване се отваря сигнатурният прозорец и се разрешава логическата врата, пропускаща тактовите импулси към преместващия регистър. Едновременно с това се разрешава възприемането на импулс СТОП.

Показаната схема може да работи със синхронизирани или с несинхронизирани сигнали СТАРТ и СТОП. За работа със синхронизирани СТАРТ и СТОП ключът *K* трябва да се постави в положение 1. Тогава сигнатурният прозорец се

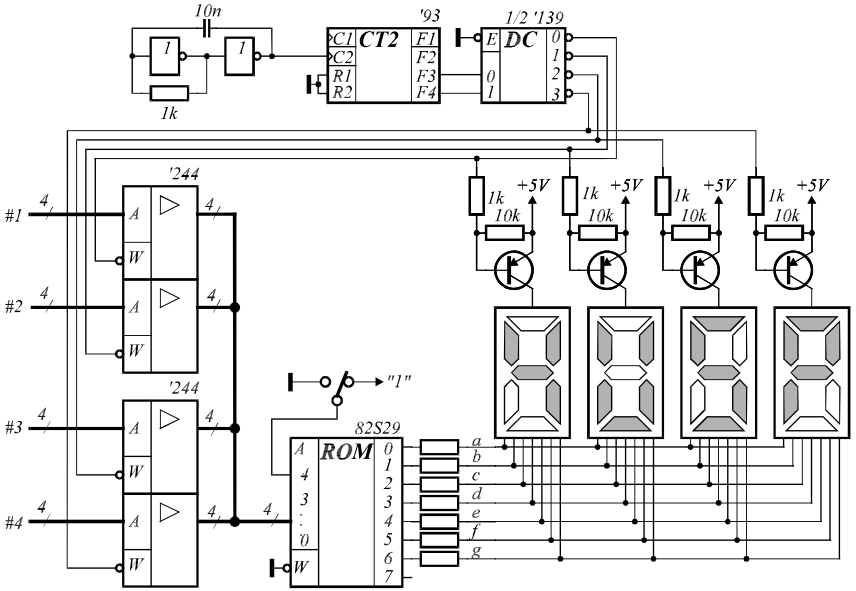
отваря по активния фронт на синхросигнала, постъпващ след активния фронт на сигнала СТАРТ. По аналогичен начин затварянето на прозореца става по активния фронт на синхросигнала, постъпващ след активния фронт на СТОП. На практика се въвежда изместване във времето на сигнатурния прозорец и за да се компенсира това изместване, се въвежда забавяне на постъпващата информация на един такт на синхронизиращите импулси чрез междинното й запомняне в *D*-тригер 74LS74. При този режим на работа се изключва възможността за неправилно формиране на сигнатурен прозорец от случайни лъжливи сигнали и образуване на нестабилна сигнатура. При него обаче е необходимо наличието на активни фронтове на сигнала ТАКТ преди и след промяна на сигналите СТАРТ и СТОП.



Фиг. 6.15. Синхронен и асинхронен режим на формиране на сигнатура.

Ако обаче не са налице сигнали СТАРТ и СТОП с достатъчна продължителност, се използува другият режим на работа, при който сигнатурният прозорец се формира непосредствено от активните фронтове на сигналите СТАРТ и СТОП, т.е. те могат да бъдат с продължителност по-малка от периода на синхросигналите ТАКТ. Този режим се осъществява при поставяне на ключа *K* в по-

ложение 2. Макар че е по-малко предпочитан, този режим може да се окаже единствен за получаването на стабилни сигнатури. Двата режима на формиране на сигнатурния прозорец са показани на фиг. 6.15 – съответно а и б.



Фиг. 6.16. Организиране на индикация за изобразяване в сигнатурната система на Hewlett-Packard.

Схемата за изобразяване на информацията от сигнатурния преместващ регистър с четири 7-сегментни индикатора е показана на фиг. 6.16. Това е класическа схема на динамична индикация. Особеното е, че кодовият преобразувател е изграден с TTL PROM 82S29 (с организация 32 x 8 и изходи с отворен колектор), за да може в него да се програмира начинът на изобразяване на числата. При поставянето на ключа *K* в положение 0 изобразяването на информацията става в шестнадесетична бройна система, а в положение 1 – в сигнатурната система на Hewlett-Packard. В табл. 6.1 е показано съдържанието на тази памет и съответстващите й изображения.

За приложението на сигнатурния анализ е необходимо да са изпълнени следните условия:

– проверяваното устройство още в етапа на разработването му да бъде пригодено към тестване чрез сигнатурен анализ;

– да бъде съставена методиката на тестване на устройството и да бъдат реализирани необходимите тестови програми;

– да бъдат на предварително снети образцови сигнатури и съставена под-

робна документация за тестване.

Табл. 6.1. Съдържание на кодовия преобразувател за изобразяване в сигнатурната система на Hewlett-Packad.

Адрес	*Данни*								*Образ*	*Адрес*	*Данни*								*Образ*
	D6	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*	*hex*			*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*	*hex*	
00	1	0	0	0	0	0	0	40	**0**	10	1	0	0	0	0	0	0	40	**0**
01	1	1	1	1	0	0	1	79	**1**	11	1	1	1	1	0	0	1	79	**1**
02	0	1	0	0	1	0	0	24	**2**	12	0	1	0	0	1	0	0	24	**2**
03	0	1	1	0	0	0	0	10	**3**	13	0	1	1	0	0	0	0	30	**3**
04	0	0	1	1	0	0	1	19	**4**	14	0	0	1	1	0	0	1	19	**4**
05	0	0	1	0	0	1	0	12	**5**	15	0	0	1	0	0	1	0	12	**5**
06	0	0	0	0	0	1	0	02	**6**	16	0	0	0	0	0	1	0	02	**6**
07	1	1	1	1	0	0	0	78	**7**	17	1	1	1	1	0	0	0	78	**7**
08	0	0	0	0	0	0	0	00	**8**	18	0	0	0	0	0	0	0	00	**8**
09	0	0	1	0	0	0	0	10	**9**	19	0	0	1	0	0	0	0	10	**9**
0A	0	0	0	1	0	0	0	08	**A**	1A	0	0	0	1	0	0	0	08	**A**
0B	0	0	0	0	0	1	1	03	**b**	1B	1	0	0	0	1	1	0	46	**C**
0C	1	0	0	0	1	1	0	46	**C**	1C	0	0	0	1	1	1	0	0E	**F**
0D	0	1	0	0	0	0	1	21	**d**	1D	0	0	0	1	0	0	1	09	**H**
0E	0	0	0	0	1	1	0	06	**E**	1E	0	0	0	1	1	0	0	0C	**P**
0F	0	0	0	1	1	1	0	0E	**F**	1F	1	0	0	0	0	0	1	41	**U**

Изпълняването на тези условия неминуемо оскъпява изделията, затова сигнатурният анализ има смисъл да се прилага при по-едросерийно производство. Силата на сигнатурния анализ е главно в сервизната диагностика, където тя може да се извършва и от невисококвалифициран персонал.

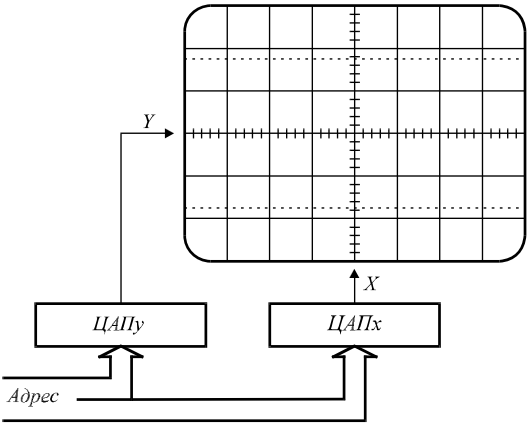
6.5. Изобразяване на динамичната карта на паметта

Динамичната карта на паметта на един микропроцесор представлява съвкупност от адресите, към които той се обръща за четене или запис при изпълнение на работната си програма. Нейното изобразяване върху екрана на осцилоскоп е по същество "фотография" на работата на адресната магистрала.

Адресните линии на микропроцесора се разделят на две части, които се подават на два цифрово-аналогови преобразувателя. Изходите на преобразувателите управляват входовете X и Y на осцилоскоп (фиг. 6.17). Така на всеки адрес от пространството на микропроцесора се съпоставя една точка от екрана на осцилоскопа.

Принципната схема на устройство за изобразяване върху екрана на осцилоскоп на динамичната карта на паметта на микропроцесор, който може да адреси-

ра пряко 64К адреса, е показана на фиг. 6.18. Шестнадесетте адресни линии от *A0* до *A15* се подават на два 8-разредни ЦАП, единият от които захранва входа Y на осцилоскоп, а другият – входа X. По този начин върху екрана се формира квадратна матрица с размери 256 на 256 точки, всяка от които съответствува на един-единствен адрес.

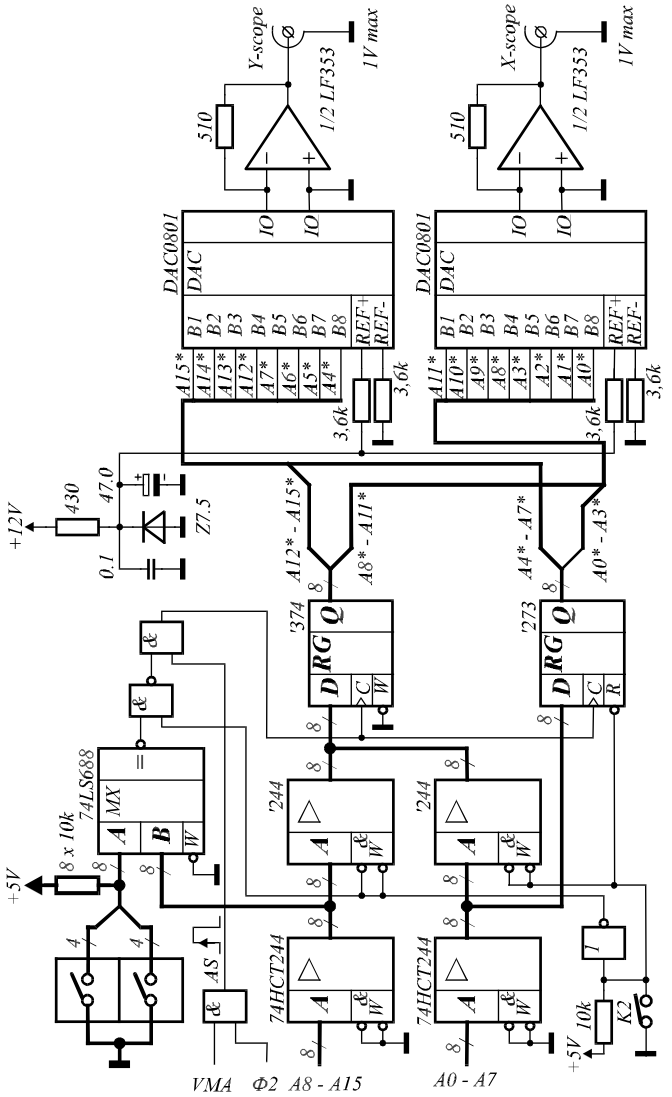


Фиг. 6.17.Управление на осцилоскоп за изобразяване на динамична карта на паметта на микропроцесорна система.

Старшите адресни линии *A12*, *A13*, *A14* и *A15* се подават съответно на *B4*, *B3*, *B2* и *B1* на $ЦАП_Y$, формиращ сигнала за Y входа, чрез което се постига разделянето на осцилоскопния екран на 16 хоризонтални ивици, всяка от които съдържа по 4К адреса. Следващите четири адресни линии *A8*, *A9*, *A10* и *A11* се подават съответно на *B4*, *B3*, *B2* и *B1* на $ЦАП_X$, в резултат на което всяка от шестнадесетте хоризонтални ивици се разделя на 16 вертикални блока, съдържащи по 256 последователни адреса от паметта. Едно такова разпределение на адресните линии дава възможност 256 последователни адреса да се изобразят в правоъгълник със страни съответно 1/16 от целия изобразяван хоризонтален размер и 1/16 от целия изобразяван вертикален размер (фиг. 6.19 а). Останалите адресни линии постъпват на входовете на двата цифрово-аналогови преобразувателя съответно от *A4* до *A7* на входовете от *B8* до *B5* на $ЦАП_Y$, а от *A0* до *A3* на входовете от *B8* до *B5* на $ЦАП_X$. Между адресната магистрала и входовете на цифрово-аналоговите преобразуватели информацията се запомня в буферна памет (*D*-тригери 74LS374 и 74LS273) по нарастващия фронт на сигнала AS. За микропроцесора СМ601 сигналът AS се формира от логическото умножение на VMA и Ф2. Адресните сигнали се поемат с буфери 74HCT244, за да не се влияе на работата на микропроцесорната система.

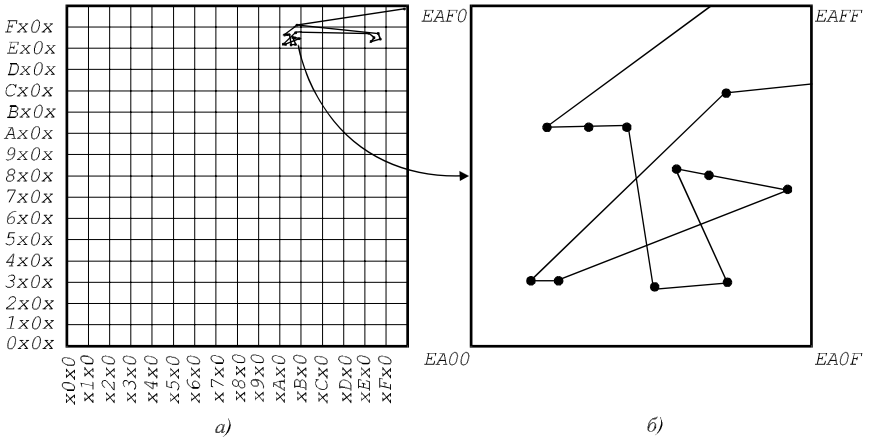
В показаната схема е предвидена възможност за разгъване на екрана и изобразяване само на една част от адресното пространство с обем 256 адреса с цел

по детайлно разглеждане на хода на микропроцесорната програма. Едно такова разгъване е показано на фиг. 6.19 б. Режимът на работа (изобразяване на цялото поле или на част от него) се определя от ключа *K1*.



Фиг. 6.18. Принципна схема на устройство за изобразяване динамичната карта на паметта на микропроцесорна система върху екрана на осцилоскоп.

При избран режим на изобразяване на 256 адреса на екрана се формира изображение само от младшите 8 адреса. Двата цифрово-аналогови преобразувателя се включват да работят като 4-разредни. За целта адресните линии *A4*, *A5*, *A6* и *A7* се подават съответно на *B4*, *B3*, *B2* и *B1* на ЦАП$_Y$, а адресните линии *A0*, *A1*, *A2* и *A3* – на *B4*, *B3*, *B2* и *B1* на ЦАП$_X$. Това превключване на адресните линии се извършва от мултиплексор, реализиран чрез буфери с три състояния 74LS244, още във входовете на междинната памет 74LS374.



Фиг. 6.19. Разпределение на адресите от динамичната карта на паметта по екрана на осцилоскопа.

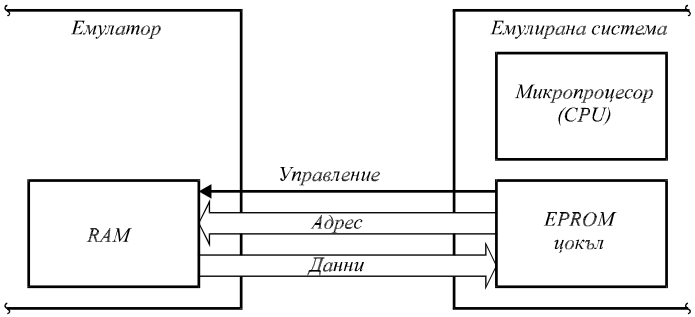
На входовете от *B8* до *B5* на двата ЦАП се подават логически *0*. Това се постига чрез нулиране на междинната памет 74LS273. Частта от общото адресно пространство, което трябва да се разгъне на екрана, се задава чрез превключвателите *K2*. Тяхното съдържание се сравнява с информацията на старшите адресни линии от *A8* до *A15* в цифров компаратор 74LS688 и при равенство се разрешава записът в междинната памет.

7. ВЪТРЕШНОСХЕМЕН ROM-ЕМУЛАТОР

Микропроцесорният вътрешносхемен емулатор е един от най-мощните уреди, използувани за настройка и диагностика на микропроцесорни системи. Голямото му неудобство е, че е процесорно зависим, т.е. един микропроцесорен емулатор може да обслужва системи с определен микропроцесор или най-много с определена група съвместими микропроцесори. В момента съществуват изключително много микропроцесори. До голяма степен изборът за работа с даден микропроцесор се определя от наличието на някакво развойно средство изобщо и на микропроцесорен емулатор за него, в частност. Конструкторът е изправен пред дилемата или да се снабди с емулатор за микропроцесора, с който той желае да работи (дори и сам да си го направи), или да извърши разработката с микропроцесор, за който вече разполага с емулатор.

7.1. Функция и структура на ROM-емулатор

Идеята за създаването на достатъчно мощно и удобно средство за настройка и диагностика на микропроцесорни системи, което да не зависи от използувания микропроцесор, е довела до появата на вътрешносхемния ROM-емулатор. Микропроцесорните системи се различават по изграждащия ги микропроцесор, по включената в тях периферия, по вида и големината на оперативната си памет. Но всички те си приличат по това, че притежават постоянна памет, в която е записана работната програма (или поне част от нея). Освен това различните фирми, произвеждащи широко използуваните постоянни памети EPROM и EEPROM, са ги унифицирали така, че те са еднакви в различните системи.

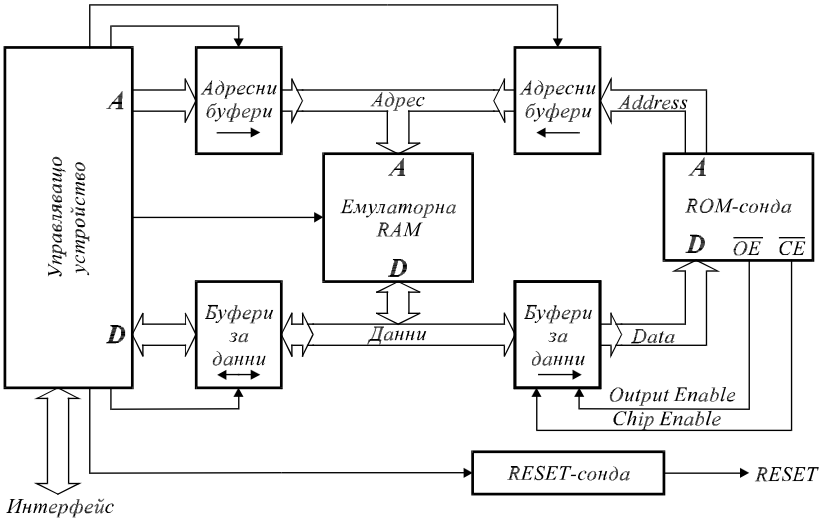


Фиг. 7.1. Включване на ROM-емулатор в EPROM цокъла на емулираната микропроцесорна система.

Масово използуваните програмируеми постоянни памети за микропроцесорни системи са 2716, 2732, 2764, 27128, 27256 и 27512, съответно с обем от

2K byte до 64K byte. Стандартизирано е и разположението на техните изводи в два типа корпуси – 24 и 28-изводен. Емулирайки тези основни видове постоянни памети, с едно устройство се създава възможността да се настройват и диагностицират почти всички микропроцесорни системи.

ROM-емулаторът се включва в една микропроцесорна система на мястото на постоянната й памет, заменяйки я с оперативна памет – фиг. 7.1. Потребителят може да чете и да пише в тази памет, т.е. за него тя е оперативна, докато тестваната система може само да чете от нея. Основното предназначение на ROM-емулатора е за настройване на програмното осигуряване на разработваната микропроцесорна система. Той с успех се използува и когато трябва да се променят вече съществуващи програми или да се подложат на тестване.



Фиг. 7.2. Блокова схема на емулаторната част ROM-емулатор.

На фиг. 7.2 е показана блокова схема на емулаторната част на един ROM-емулатор. Емулаторната памет е оперативна памет, която се включва към тестваната система вместо нейната постоянна памет. Обемът й съответствува на обема на най-големия от емулираните чипове постоянна памет. Когато се извършва емулиране на по-малки по обем постоянни памети, оперативната памет не се използува напълно. Адресирането на емулаторната памет се извършва алтернативно (взаимоизключващо се) от две страни – от страна на управляващото устройство и от страна на тестваната система. На практика достъпът до емулаторната памет е мултиплексиран, като приоритет има управляващото устройство. Мултиплексирането на адресните линии е осъществено чрез еднопосочни

буфери с високо импедансно състояние на изходите в изключено положение. Мултиплексирането на информационната магистрала също се извършва с буфери, имащи високоимпедансно състояние, като към управляващото устройство те са двупосочни, а към тестваната система – еднопосочни.

Включването на емулатора към тестваната система става чрез ROM-сонда. В нея или на входа на буферния блок се вземат мерки за защита на ROM-емулатора от некоректни сигнали, подадени от тестваната система. Тези мерки се състоят най-често в поставянето на токоограничаващи резистори и защитни диоди, както при микропроцесорния емулатор (вж. фиг. 3.17).

RESET-сондата служи за подаване на сигнал за начално установяване към тестваната система. Задължително в емулатора се предвижда възможност за управляване на полярността на сигнала *RESET*. Това е единственият сигнал, който не изхожда от гнездото на емулираната постоянна памет. С единичен проводник той се свързва с *RESET*-сигнала на тестваната система.

Управляващото устройство най-често се изгражда като микропроцесорна система. То регламентира достъпа до емулаторната памет, определя полярността на сигнала *RESET* и го управлява според изискванията на работата. Чрез периферния интерфейс управляващото устройство се свързва с оператор или със система за развитие. Някои ROM-емулатори притежават локална клавиатура и дисплей за комуникация с оператор. В управляващото устройство може да бъде включен цокъл за ROM, в който може да бъде поставена постоянна памет със записана в нея програма, която да бъде прочетена и прехвърлена в емулаторната оперативна памет. Това се използува, когато трябва да се тества или провери вече изработена и запомнена в ROM микропроцесорна програма.

Елементарните операции, които трябва да извършва управляващото устройство, са:

– прочитане на постоянна памет от ROM-цокъла и прехвърляне на съдържанието й в емулаторната памет;

– зареждане в емулаторната памет на микропроцесорна програма, получена по комуникационния интерфейс;

– осигуряване на достъп до клетка от емулираната памет и възможност за промяна на съдържанието й;

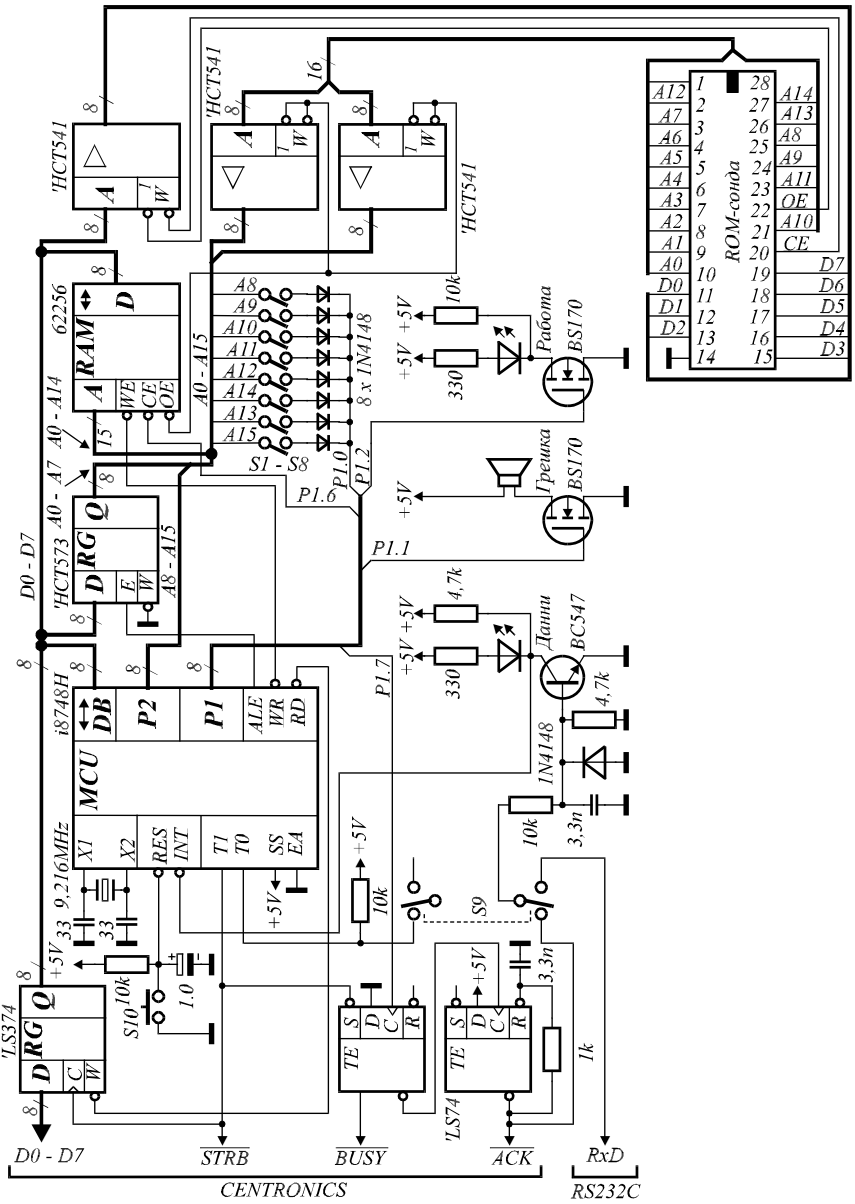
– визуализиране на масив от емулаторната памет;

– стартиране и спиране на потребителска програма.

Всички тези функции се реализират по програмен начин от т.нар. мониторна програма на ROM-емулатора, подобно на мониторната програма на микропроцесорните емулатори.

7.2. Схемотехника на ROM-емулатор

На фиг. 7.3 е показана принципна схема на ROM-емулатор. Той е изграден на базата на едночиповия микрокомпютър i8748.



Фиг. 7.3. Принципна схема на EPROM-емулатор, имащ възможност да емулира постоянна памет EPROM с обем до 32 K byte (27256).

Емулаторът може да емулира постоянна памет с обем от 1K byte до 32K byte. Предназначен е за емулиране на постоянни памети в 8-разредни микропроцесорни системи, но е възможно паралелно включване на два емулатора за да се емулира постоянна памет в 16-разредни системи. В такъв режим на работа е необходимо допълнително да се включат стробове за младшите и за старшите данни – *LDS* и *UDS*, при работата на емулаторите.

Достъпът на тестваната система до емулаторната памет се осъществява чрез 24 или 28-изводна сонда. Входящите адресни и управляващи сигнали са свързани през резистори към +5V, което гарантира, че неизползуван сигнал или сигнал във високоимпедансно състояние ще бъде възприет от емулатора като логическа единица.

Сигналите $\overline{OE}$ (Output Enable) и $\overline{CE}$ (Chip Enable), сумирани във функция ИЛИ, разрешават еднопосочния буфер 74HC541, пропускащ информационните сигнали от емулаторната памет към настройваната система.

Достъпът до емулаторната памет се управлява от извода P1.2 на едночиповия микрокомпютър. Когато той е в логическа *0*, достъп до емулаторната памет има микрокомпютърът; а когато извод P1.2 е в състояние на логическа *1*, се разрешават буферите, пропускащи адреса от тестваната система и тя има достъп до емулаторната памет. Същевременно едночиповият микрокомпютър забранява вътрешните си буфери, които издават адресните и информационните сигнали, поставя извода си $\overline{WR}$ в логическа *1* за указване на режим четене на паметта и постоянно я избира чрез извода си P1.6.

Връзката на ROM-емулатора с оператора се извършва по стандартни интерфейси RS232C или CENTRONIX. Видът на интерфейса се избира чрез механичния превключвател *S9*. Допълнителна информация за типа на емулираната ROM, за скоростта на получаване на информацията по интерфейса RS232C и за вида на получавания пакет данни, се въвежда чрез микропревключвателите *S1* ÷ *S8*, които се сканират от старшите адресни линии от *A8* до *A15*, а състоянието им се прочита от извода P1.0.

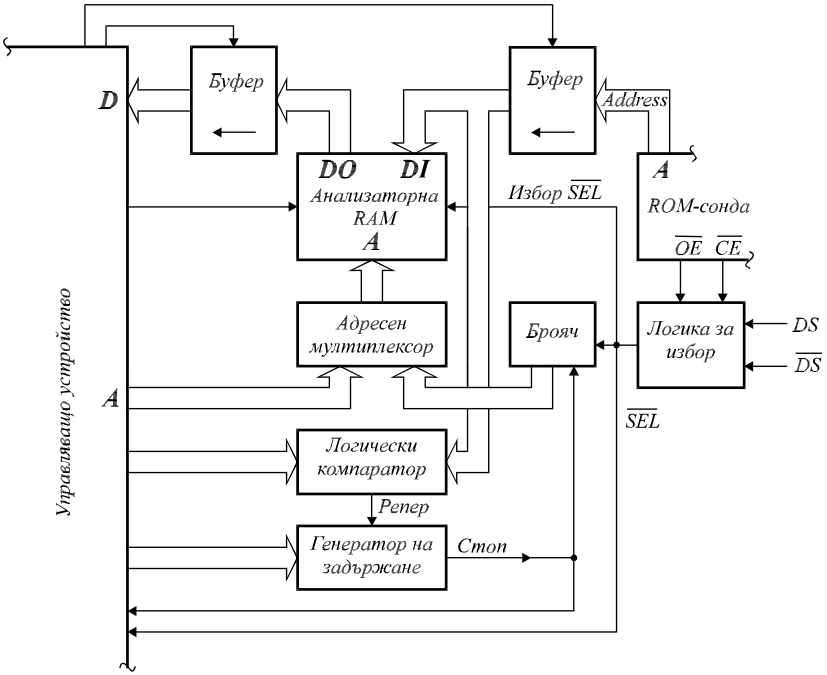
Представеният ROM-емулатор не притежава *RESET*-сонда, но има възможност за светлинна индикация на режима на работа на тестваната система, на получаваните серийни данни, както и за звукова индикация при грешка в обмена.

7.3. Анализаторни възможности на ROM-емулатор

Възможностите на ROM-емулаторите да се използуват при различни микропроцесорни системи ги направиха желан и търсен диагностичен уред. Фирмите производителки непрекъснато ги усъвършенствуват и снабдяват с нови възможности. Едно такова усъвършенствуване е снабдяването им с вграден анализатор на логически състояния, който е познат и с името *трасиращ буфер*. В този трасиращ буфер се запомнят адресите, които е обхождала настройваната система в процеса на работата си. Допълнително в трасиращия буфер могат да

се запомнят и някои други сигнали. Както при логическите анализатори, така и при ROM-емулаторите, трасиращият буфер е с кръгова организация, като потребителят може да задава точки на прекъсване (репер) и закъснение след репера.

На фиг. 7.4 е показана блокова схема на анализаторната част (трасиращ буфер) на ROM-емулатор. Тъй като в емулатора се намира настройваната програма, достатъчно е да се запомни само последователността на издаваните адреси, за да се добие цялостна представа за хода на настройваната програма. Показаната анализаторна памет е с широчина, равна на разредността на адреса.



Фиг. 7.4. Блокова схема на анализаторната част на ROM-емулатор (Trace buffer).

От сондата, през еднопосочни буфери, издаваните адреси постъпват на входовете за данни *DI* на анализаторната памет. От изходите за данни *DO* те могат да бъдат прочетени от управляващото устройство. Кръговата организация на анализаторната памет се реализира чрез адресирането и от двоичен брояч, който при препълване се нулира и започва от нула. Тактуването на записа се извършва от всяко обръщение на тестваната система към емулаторната памет. За целта се използуват сигналите $\overline{OE}$ и $\overline{CE}$, като всяко деактивиране на който и

да е от тях (или и на двата) се възприема като запис и преминаване към следваща клетка от анализаторната памет. Ако диагностицираната система е така проектирана, че сигналите $\overline{OE}$ и $\overline{CE}$ са постоянно активирани, тогава за тактуване на записа е необходимо да се вземат допълнителни сигнали от тестваната система с единични сонди. В показаната схема допълнителните сигнали са *DS*, с активно ниво логическа *1*, и $\overline{DS}$, с активно ниво логическа *0*. Двата сигнала са включени във функция ИЛИ и се използуват заедно или поотделно, в зависимост от конкретния случай. Тяхната комбинация трябва да формира активен преход на сигнала $\overline{SEL}$ за запис в анализаторната памет. От страна на управляващото устройство анализаторната памет може да е пряко адресируема. Превключването на адресирането в този случай става чрез адресен мултиплексор.

Логическият компаратор изработва репера, т.е. отделя необходимото на оператора събитие. Най-простият логически компаратор се изгражда като цифров компаратор, сравняващ текущия адрес със зададения в сравняващия регистър адрес. По-сложните логически компаратори могат да изработват репер по комбинация от постъпващи адреси.

Генераторът на задържане представлява брояч, който изброява зададено количество прочитания от настройваната система на емулаторната памет и спира процеса на запомняне на издадените адреси. Позицията, на която е спряла кръговата анализаторна памет, се установява чрез прочитане на състоянието на адресиращия я брояч, след като записът на адресите е спрял.

Нормално управляващото устройство е едно и също и за емулаторната, и за анализаторната част и представлява микропроцесорна система. То изработва всички допълнителни сигнали, необходими за правилна работа на анализаторната част.

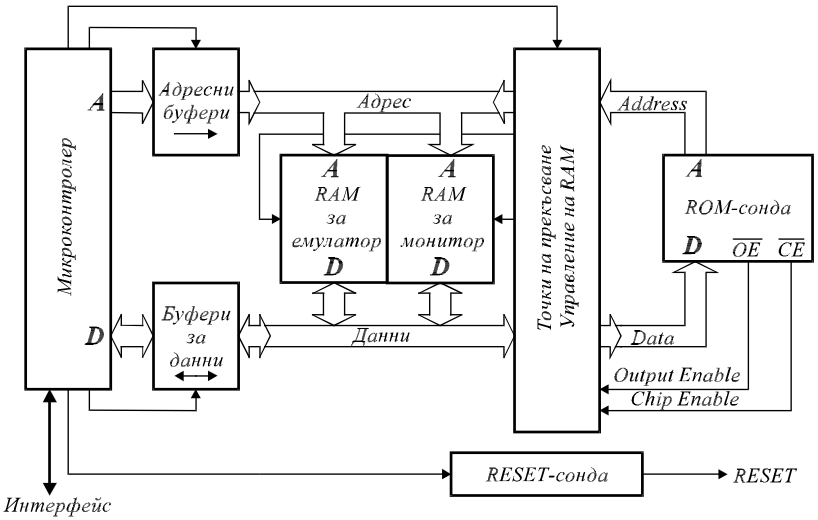
В повечето ROM-емулатори трасиращият буфер е добавка, която се поставя по желание на оператора.

7.4. Точки на прекъсване при ROM-емулатор

Както при микропроцесорния емулатор, така и при ROM-емулатора е възможно да се поставят и обслужват точки на прекъсване. Тук обаче е възможно единствено решението на поставяне на точки на прекъсване чрез адресен компаратор. Той може да бъде единичен (само за една точка на прекъсване) или организиран на базата на еднобитова памет (неограничен брой точки на прекъсване). При достигане точка на прекъсване изпълнението на потребителската програма се преустановява и управлението се предава на мониторната програма. Това означава, че успоредно с емулаторната RAM в емулатора трябва да присъства и мониторна RAM, в която да бъде заредена специфична за конкретния микропроцесор мониторна програма.

На фиг. 7.5. е показана блоковата схема на допълнителната част към емулатора за обслужване на точки на прекъсване. Адресният компаратор заедно с ло-

гиката за превключване на оперативните памети се намира в блока за точки на прекъсване и управление на RAM.



Фиг. 7.5. Организиране на точки на прекъсване при ROM-емулатор.

Възможността за обслужване на точки на прекъсване премахва процесорната независимост на ROM-емулатора. За всеки отделен процесор трябва да има подходяща мониторна програма, която да се зарежда преди работа в мониторната RAM. Освен това в блока за управление на ROM-емулатара трябва да се предвидят механизми за обслужването на точки на прекъсване за различните процесори или за налагане на определени ограничения при използуването им. Проблемът идва от факта, че някои микропроцесори извършват предварително четене на инструкции, което може да доведе до фалшиво сработване на адресния компаратор.

ROM-емулаторите, които имат възможност да общуват с оператора чрез обикновен терминал или пък имат собствени входно-изходни органи, се наричат автономни. Съществуват и ROM-емулатори, които се изграждат като периферни модули за микрокомпютърни системи и не могат да работят извън състава им. Те се наричат комплексни емулатори. ROM-емулаторите дават възможност да се извърши окончателното настройване на програмното осигуряване на микропроцесорната система, преди да бъде записано в EPROM или EEPROM. Някои производители на ROM-емулатори предлагат като добавка програматорна част за EPROM и EEPROM, позволяваща програмирането на работната програма след нейното окончателно настройване.

8. САМОДИАГНОСТИКА

Самодиагностиката има за задача да извърши проверка на работоспособността на електронната система или на отделни нейни блокове непосредствено преди работа или в процеса на самата работа.

Възможностите за самодиагностика зависят от самата електронна система. Те са особено големи и ефикасни при микропроцесорните системи. За това спомагат няколко фактора:

– програмна достъпност на микропроцесора до всички блокове на системата, което му позволява да им задава управляващи въздействия и да се информира за състоянието им;

– способност на микропроцесора да генерира по програмен път значителни тестови последователности;

– възможност на микропроцесора да извършва логическа обработка на информацията, позволяваща му локализиране на неизправности без помощта на допълнителна апаратура;

– възможност за съсредоточаване на диагностичните функции в ограничен кръг схемни елементи и следователно за разделяне на системата на проверяваща и проверявана част.

8.1. Самодиагностика в микропроцесорна система

Самодиагностиката е преди всичко програмна диагностика. Тя изисква разработването на конкретни за всяка система тестови програми. Постоянната памет, съдържаща тестовите програми, нормално винаги присъствува в състава на микропроцесорната система. В редки случаи, когато се отнася до икономия на консумирана енергия или място, тази памет може да не присъствува постоянно. В случая когато се извършва програмна диагностика, тя се поставя на мястото на паметта, съдържаща работната програма на системата. Тогава обаче се губи възможност за програмно тестване на работната програма.

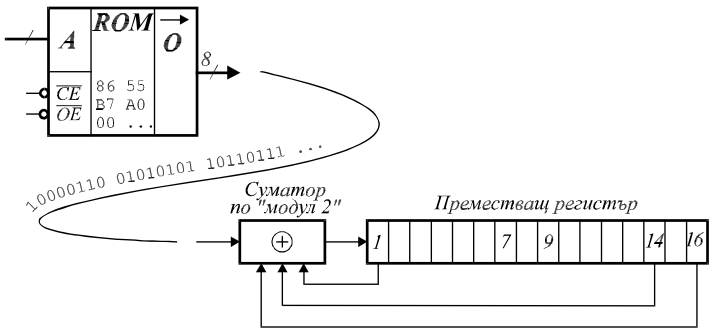
Извършването на самодиагностика в микропроцесорна система изисква нейната частична работоспособност. Необходимо е да бъдат изправни захранването на системата, тактовият генератор, линиите на микропроцесора за връзка с постоянната памет, която съдържа тестовата програма и др. Освен това е необходимо да бъдат работоспособни някои от регистрите на микропроцесора и няколко команди, включващи поне една команда за условен преход. Ако тези условия не са изпълнени, самодиагностика не може да се проведе.

Самодиагностиката може да започне с проверка на микропроцесора. Погоре беше предявено условието някои от микропроцесорните регистри да бъдат изправни и някои от командите да се изпълняват правилно. Тестовата програма за микропроцесора трябва по възможност да е изградена от ограничен набор

микропроцесорни инструкции и да ползува минимален брой негови регистри. Именно тези инструкции и регистри трябва да бъдат работоспособни съгласно условието. Проверката на микропроцесора обхваща всички останали негови регистри, команди и вътрешни магистрали. Тя се осъществява чрез запис и четене на информация във всички регистри, използуване на всички инструкции на микропроцесора с известни данни и проверка за правилността на резултата и др. Ефективността на самодиагностиката на микропроцесора зависи изключително от способностите на програмиста за съставяне на диагностични програмни тестове.

Най-често програмната проверка на постоянна памет се извършва, като се пресметне контролната сума на съдържанието й и се сравни със зададената. Затова е добре вътре в постоянната памет да се записва и контролната сума. Тя може да се съхранява в последната клетка от постоянната памет. Тогава, естествено, тази клетка не участвува в сумирането за получаването на контролната сума. В някои случаи, контролната сума може да се съхранява в други запомнящи устройства.

По-точен метод, при използуването на който съществува по-малка вероятност за взаимно маскиране на грешки, е методът с използуването на контролен цикличен код (CRC - Cyclic Redundancy Check). Той се прилага успешно за контрол на данни при запис на информация върху магнитни носители. Със същия успех методът на CRC може да се използува и за проверка на съдържанието на постоянна памет. Формирането на CRC за постоянна памет е пояснено от фиг. 8.1. Съдържанието на постоянната памет се извежда от нея дума по дума, така че се образува една голяма линейна последователност от битове.



Фиг. 8.1. Апаратно формиране на контролен цикличен код.

Тази последователност се въвежда в преместващ регистър с линейни обратни връзки, благодарение на което се получава свиване на информацията. Резултатът, останал в регистъра след въвеждането на всички битове от съдържанието

на постоянната памет, представлява именно контролния цикличен код.

Методът е същият както при получаването на сигнатурата в сигнатурния анализатор (вж. глава "Сигнатурен анализ"), с тази разлика, че при контролния цикличен код по правило се работи с друг характеристичен пораждащ полином. За формиране на контролен цикличен код с 16-разреден преместващ регистър най-широко се използува пораждащият полином

$$G(X) = X^{16} + X^{15} + X^2 + 1. \qquad (8.1)$$

Според него линейните обратни връзки се вземат от първия, четиринадесетия и шестнадесетия разред на преместващия регистър.

Методът на проверката включва изчисляване на контролния цикличен код за информацията, записана в постоянната памет, и сравняването му със зададения CRC. Ако предварително изчисленият контролен цикличен код (CRC е също остатък от делене на два полинома) е добавен към съдържанието на постоянната памет, при изчисляването на CRC ще се получи нулев остатък, тъй като $A(X) + R(X)$ точно се дели на $G(X)$. От (6.3) се вижда, че $A(X) – R(X)$ се дели точно на $G(X)$, но тъй като в аритметика по modulo2 операциите събиране и изваждане дават един и същи резултат, действието на $A(X) + R(X)$ е същото.

При самодиагностиката описаната процедура за формиране на CRC се реализира по програмен път. Следващата подпрограма илюстрира въвеждането на байт в преместващия регистър `CRC_reg` за изчисляването на CRC. Преди започване на изчисляването на контролния цикличен код `CRC_reg` трябва да се нулира, а преди всяко извикване на подпрограмата текущият байт трябва да се запише в `next_byte`. Подпрограмата е написана на C за микропроцесор MC68HC11 с ползуване на компилатора на IAR.

```
static char next_byte;
static unsigned int CRC_reg;
const unsigned int polynom = 0x8005

/*-------------------------------------------------------------
   Подпрограма за въвеждане на байт за изчисляване на CRC
   Байтовете се въвеждат със старшия бит напред
-------------------------------------------------------------*/
void CRC_calc(char next_byte)
     {
     char i, CRC_next;

     for (i = 0; i < 8; i++)
         {
         CRC_next = (0x8000 & CRC_reg) >> 8 ^ (next_byte & 0x80);
         CRC_reg << 1;
         if (CRC_next != 0) CRC_reg = CRC_reg ^ polynom;
         next_byte << 1;
         }
     }
```

За тестване на оперативна памет съществуват различни методи. Различават

се по времето на изпълняването, по точността на проверката и по пълнотата на диагностиката. Всички те се свеждат до извършване на последователен запис на определени кодови комбинации в клетките на паметта, четене и проверка на прочетеното. Адресирането на клетките и изборът на кодовите комбинации се извършват съгласно специфичен за всеки метод алгоритъм.

Алгоритъмът на "шахматния тест" изисква записване в битовете на проверяваната оперативната памет редуващи се така единици и нули, както се разполагат черните и белите полета върху шахматната дъска, и проверка на правилността на записаната информация. След това, във всеки бит се записва противоположната стойност и отново се извършва проверка. При това се изяснява може ли всеки бит от паметта да запомня 0 и 1.

Алгоритъмът на теста "запълващи нули и единици" започва с първоначално нулиране на цялата памет. В първия бит се записва 1. Проверява се записът и се изчита цялата памет, като се проверява дали някъде другаде няма промяна. След това се записва 1 във втория бит и отново се проверява цялата памет. Това продължава, докато всички битове бъдат запълнени с 1. Второ тестване се извършва по същия начин на цялата памет, като сега в обратен ред тя постепенно се запълва с нули.

При алгоритъма на "теста с допълнителни адреси", първоначално цялата памет се запълва поадресно с нули и единици, т.е. клетките с четен адрес се запълват с нули, а клетките с нечетен адрес се запълват с 1. Проверката за правилността на записа започва с първата клетка. След нея се проверява клетката, чийто адрес представлява допълнително число на адреса на вече проверената първа клетка. След това се проверява втората клетка и после клетката, чийто адрес е допълнителен на нейния и т.н., докато се провери цялата памет. После съдържанието на паметта се инвертира и процедурата се повтаря отново. Този алгоритъм на проверка на паметта представлява сериозно изпитание за вътрешните й дешифратори и позволява да се открият неизправности в системата на адресирането на оперативната памет.

Съществуват още много алгоритми за тестване на оперативна памет. Някои от тях са свързани с голяма загуба на време, като пълното им изпълнение може да отнеме няколко часа. Затова обикновено за самодиагностика се използуват прости и бързи тестове, като например "шахматния тест", а по-сложни тестове, с по-големи възможности, се използуват за откриване на вече подозирани неизправности в оперативната памет. Следващата програма реализира метода на "шахматния тест" за диагностика на оперативна памет и е предназначена за микропроцесори на Motorola MC6800/01/03/08/09/11.

```
MC6801 Cross Assembler

00001                   ****************************************
00002                   *                                      *
00003                   *             - RAMTST -               *
```

```
00004                    * ПОДПРОГРАМА ЗА ТЕСТВАНЕ НА ОПЕРАТИВ- *
00005                    * НА ПАМЕТ ПО МЕТОД НА "ШАХМАТЕН ТЕСТ" *
00006                    *                                      *
00007                    ****************************************
00008                    *   ПОДПРОГРАМАТА ЗАПОЧВА ОТ АДРЕС $F000,
00009                    * НО МОЖЕ ДА СЕ ЗАДАДЕ И ОТ ДРУГ АДРЕС.
00010                    *   ПОДПРОГРАМАТА ИЗИСКВА КАТО ВХОДНИ
00011                    * ДАННИ В КЛЕТКИ BEG И END ДА СЕ ЗАПИШАТ
00012                    * ДВУБАЙТОВИТЕ СТОЙНОСТИ НА НАЧАЛОТО И
00013                    * НА КРАЯ НА ТЕСТВАНИЯ МАСИВ ОПЕРАТИВНА
00014                    * ПАМЕТ, А В КЛЕТКА TIME ДА СЕ ЗАПИШЕ
00015                    * ВРЕМЕТО НА ЗАКЪСНЕНИЕ МЕЖДУ ЗАПИСА И
00016                    * ПРОВЕРКАТА В ЕДИНИЦИ MS.
00017                    *   УСПЕШНИЯТ ТЕСТ ЗАВЪРШВА С ВРЪЩАНЕ ОТ
00018                    * ПОДПРОГРАМАТА.
00019                    *   ПРИ ВЪЗНИКВАНЕ НА ГРЕШКА УПРАВЛЕНИЕ-
00020                    * ТО СЕ ПРЕДАВА НА ПОСЛЕДНИЯ ЕТИКЕТ В
00021                    * ПОДПРОГРАМАТА, КОЕТО ОБЕЗПЕЧАВА РАЗШИ-
00022                    * РЯВАНЕТО Й, КАТО IX СЪДЪРЖА АДРЕСА НА
00023                    * ГРЕШКАТА. СТЕКЪТ НЕ СЕ ВЪЗСТАНОВЯВА.
00024                    ****************************************
00025A F000                       ORG  $F000
00026                    ****************************************
00027                    RAMTST   EQU  *
00028                    * ТЕСТВАНЕ С 01010101
00029A F000  86 55               LDAA #$55
00030A F002  8D 0D               BSR  INT
00031A F004  8D 18               BSR  DELAY
00032A F006  8D 22               BSR  TEST
00033                    * ТЕСТВАНЕ С 10101010
00034A F008  86 AA               LDAA #$AA
00035A F00A  8D 05               BSR  INT
00036A F00C  8D 10               BSR  DELAY
00037A F00E  8D 1A               BSR  TEST
00038A F010  39                  RTS
00039                    *
00040                    * ПОДПРОГРАМА ЗА ЗАПИС В ПАМЕТТА НА АССА
00041                    *
00042A F011  FE 7010     INT     LDX  BEG
00043A F014  09                  DEX
00044A F015  08          LOOP1   INX
00045A F016  A7 00               STAA 0,X
00046A F018  BC 7012             CPX  END
00047A F01B  26 F8               BNE  LOOP1
00048A F01D  39                  RTS
00049                    *
00050                    * ПОДПРОГРАМА ЗА ФОРМИРАНЕ НА ЗАКЪСНЕНИЕ
00051                    *
00052A F01E  FE 7014     DELAY   LDX  TIME
00053A F021  C6 A5       LOOP3   LDAB #165
00054A F023  5A          LOOP2   DECB
00055A F024  26 FD               BNE  LOOP2
00056A F026  09                  DEX
00057A F027  26 F8               BNE  LOOP3
00058A F029  39                  RTS
00059                    *
00060                    * ПОДПРОГРАМА ЗА ПРОВЕРКА НА ПАМЕТТА
00061                    *
00062A F02A  FE 7010     TEST    LDX  BEG
```

```
00063A F02D  09                 DEX
00064A F02E  08 LOOP4           INX
00065A F02F  A1 00              CMPA 0,X
00066A F031  26 02              BNE  ERROR
00067A F033  20 F9              BRA  LOOP4
00068                ERROR      EQU  *
00069                ************************************
00070                * РЕЗЕРВИРАНЕ НА РАБОТНИ КЛЕТКИ
00071A 7010                     ORG  $F7010
00072A 7010     0002 BEG        RMB  2
00073A 7012     0002 END        RMB  2
00074A 7014     0002 TIME       RMB  2
00075                ************************************
00076                           END
```

Показаната програма извършва последователно записване на шестнадесетичните числа 55 и AA в паметта и проверка за правилността на записа. Когато се тества динамична оперативна памет, между записа и проверката трябва да се въведе закъснение (около 1 минута), за да се разбере дали работи схемата за опресняване.

Многообразието на входно-изходните устройства в микропроцесорните системи и спецификата на тяхното взаимодействие с околната среда определя сложността на проблема по самодиагностиката им. За микропроцесора входно-изходните устройства представляват набор от програмно достъпни регистри. Онези вътрешни регистри, които могат да се записват и четат, се проверяват с методите за диагностика на оперативна памет. Другите регистри, за които това е невъзможно, се проверяват за работоспособност по специфични за конкретния случай косвени признаци.

Програмната диагностика на периферията на входно-изходните устройства зависи силно от индивидуалната специфика на конкретната микропроцесорна система. При възможност в нея се вграждат превключватели, позволяващи свързването на изходящите към входящите канали на микропроцесорната система. Излизащата от системата информация може да бъде отново прочетена. Проверката се осъществява чрез циклично извеждане чрез изходните канали на определени кодови комбинации и проверка за получаването им по входните канали. За случай когато това е невъзможно, проверката на входно-изходните устройства се извършва с активната намеса на оператора. Например, когато в устройството има клавиатура и индикация, диагностичната програма може да прочита натиснат клавиш и да го изобрази на дисплея или чрез дисплея да укаже на оператора кой клавиш да натисне.

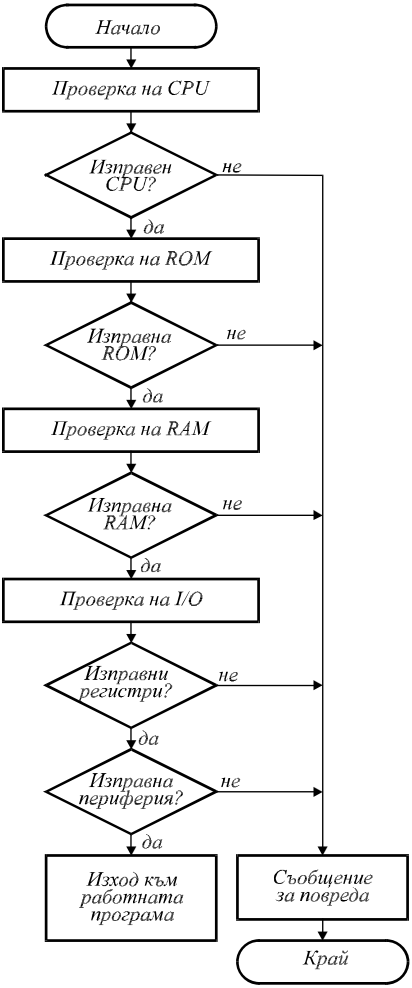
8.2. Провеждане на самодиагностика

Обобщена блокова схема на алгоритъм за извършване на самодиагностика е показана на фиг. 8.2. Съществуват три основни начина за активиране на самодиагностика:

– при всяко привеждане на устройството в начално състояние (винаги при включване на захранването) преди започване на работа;

– по време на работа, когато микропроцесорът разполага с "излишно" време;

– по заявка на оператора.



Фиг. 8.2. Алгоритъм за извършване на самодиагностика.

Разновидност на третия начин за активиране на самодиагностиката представляват вградени диагностични точки, които се активират от оператора. На възлови места в работната програма се поставят ключови разклонения, към които програмата се отклонява само когато устройството е поставено в диагностичен режим на работа. Навлизането в диагностичната програма е съпроводено с информиране на оператора за мястото на отклонение от работната програма. За целта се използува съществуващият дисплей на устройството или специален такъв. Допълнително диагностичната програма индицира състоянието (статуса) на системата, т.е. съдържанието на избрани клетки и регистри, по които може да се съди за правилната работа на устройството.

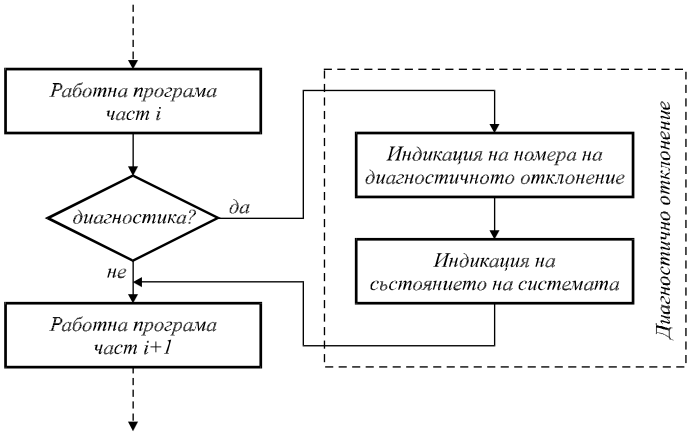
Връщането от диагностичната програма в работната се извършва през същото разклонение. Блоковата схема на примерно диагностично разклонение е показано на фиг. 8.3. Такава диагностична програма позволява проследяване на алгоритъма на работа на проверяваното устройство.

От изложеното става ясно, че осигуряването на възможност за самодиагностика при електронните устройства е свързано с изпълнението на две взаимосвързани условия:

– разработването на специализирани за всяко устройство тестови програми, генериращи управляващи

тестови сигнали и извършващи анализ на получената информация;

– вграждането на допълнителни апаратни средства, осигуряващи на устройството възможност за тествуване чрез самодиагностика.



Фиг. 8.3. Организация на вградени диагностични отклонения.

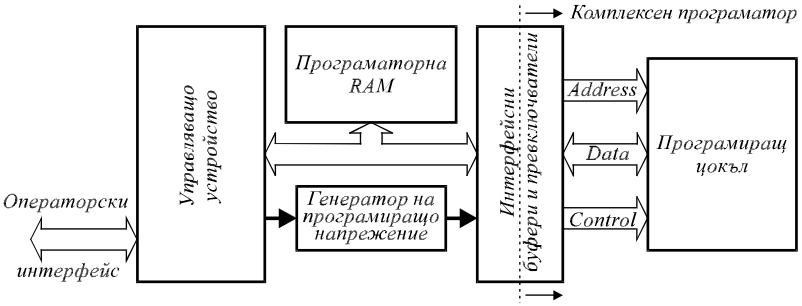
Имайки предвид, че всичко това допълнително увеличава стойността на изделието, конструкторът внимателно трябва да прецени кога и доколко е необходимо да се вгражда самодиагностика. В повечето случаи е достатъчно самодиагностиката грубо да локализира неизправността, а нейното точно откриване да бъде поверено на други методи.

9. ПРОГРАМАТОРИ

След като работната програма на една микропроцесорна система бъде написана и транслирана, тя трябва да се зареди в постоянната й памет за изпълнение. Това е завършващата фаза на настройката на микропроцесорната система. Програмирането се извършва чрез специализирани устройства, наречени програматори.

9.1. Програматори за EPROM

Едни от най-често използуваните постоянни памети за микропроцесорни системи и микроконтролери са изтриваемите програмируеми постоянни памети EPROM. Блоковата схема на програматор за тях е представена на фигура 9.1. Той се състои от управляващо устройство (най-често микропроцесорна система), което по паралелен или сериен канал се свързва с оператора (развойна система, работна станция или терминал). В програматора по правило присъствува оперативна памет с обем достатъчен, за да поеме обектовия код на програмата, която трябва да се запише в постоянната памет.

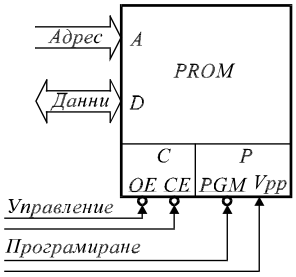


Фиг. 9.1. Блокова схема на програматор за EPROM.

В програматора се изработват програмиращите напрежения за различните видове EPROM. Програмираният EPROM се поставя в програмиращия цокъл на програматора. Ако EPROM е изтриваем с ултравиолетова светлина, той трябва предварително да бъде изтрит, при което всичките му битове се установяват в логическа *1*. Изтриването се извършва в отделно устройство, съдържащо източник на ултравиолетова светлина с дължина на вълната по-малка от 4.10^{-7} μm.

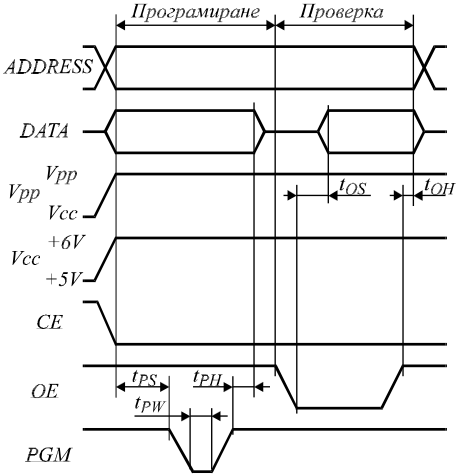
През интерфейсни буфери и превключватели програматорът задава към програмиращия цокъл необходимите адресни, даннови и управляващи сигнала. Превключвателите са необходими, тъй като съществуват определени разлики

във функциите и разположението на някои изводи при различните типове постоянна памет. В състава на управляващите сигнали влизат и допълнителните сигнали за програмиране на постоянната памет.



Фиг. 9.2. Типични управляващи сигнали за програмиране на EPROM.

На фиг. 9.2 са показани типичните управляващи сигнали за ултравиолетовоизтриваем EPROM (примерът е за ‘2764). Допълнителните сигнали за програмирането са *Vpp*, с който се задава програмиращо напрежение, и $\overline{PGM}$, с който се определя продължителността на програмиращия импулс. Програмирането се извършва в определена последователност на активиране на сигналите, която може да бъде различна за някои типове EPROM. Типичната последователност на сигналите при програмиране и проверка на байт в EPROM е представена с времедиаграмите им на фиг. 9.3.



Фиг. 9.3. Типични времедиаграми на сигналите при програмиране на EPROM.

Програмирането започва с установяването на данните за запис и адреса, на който те ще се записват. От управляващите сигнали в този момент трябва да се зададе необходимата стойност на програмиращото *Vpp* и захранващото *Vcc* напрежение, както и да се активира изборът на чипа $\overline{CE}$. След изчакване на времето на установяване t_{PS} се подава програмиращ импулс $\overline{PGM}$ с определена

от производителя продължителност t_{PW}. Сигналите към EPROM трябва да останат валидни след дезактивирането на програмиращия импулс поне за времето на изчакване t_{PH}.

След програмирането на байта може да се премине към неговата проверка. Режим "проверка" е същият както режим "четене", с тази разлика, че четенето на байта за проверка се извършва при наличие на програмиращо напрежение и на повишено захранващо напрежение. Данните се появяват време t_{OS} след активирането на $\overline{OE}$ и остават валидни още време t_{OH} след неговото дезактивиране. Някои типове EPROM нямат режим "проверка" и при тях записаният байт може да се провери в нормален режим "четене".

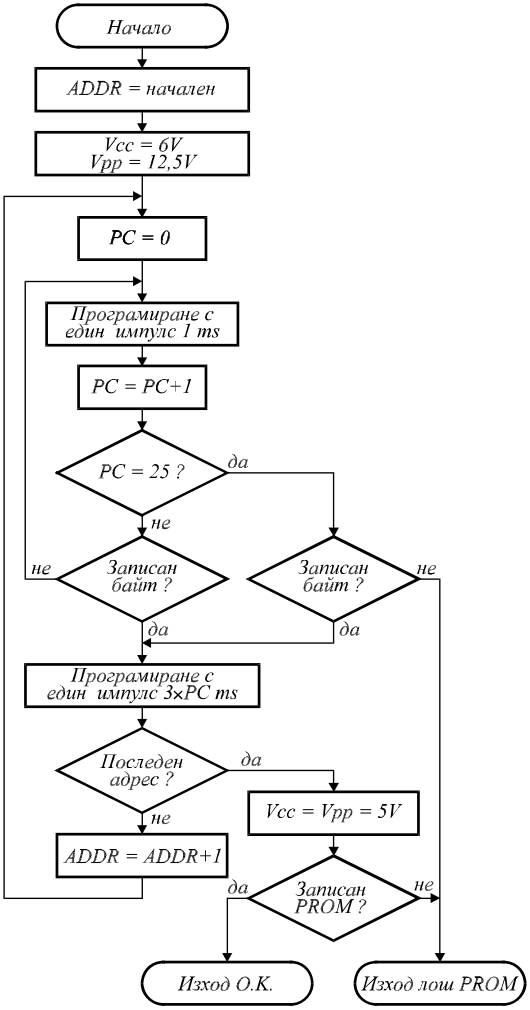
При различните типове EPROM съществуват някои разлики във функциите и местоположението на управляващите сигнали. За най-известните ултравиолетово изтриваеми EPROM от серията 25/27xxx тези разлики са дадени в табл. 9.1 и те трябва да се имат предвид при изграждането на програматор.

Табл. 9.1. Управляващи сигнали за програмиране и проверка на EPROM.

EPROM	Функция	Извод	Четене	Програмиране	Проверка
2758	*AR*	*19*	*L*	*L*	*L*
25/2716	$\overline{CE}$ / *PGM*	*18*	*L*	*Pulse* $L \to H \to L$	*L*
	$\overline{OE}$	*20*	*L*	*L*	*L*
	Vpp	*21*	*+5V*	*+25V*	*+25V*
2732/32A	$\overline{CE}$	*18*	*L*	*L*	–
	$\overline{OE}$ / *Vpp*	*20*	*L*	*Pulse* **Vpp*	–
2532	*PD* / $\overline{PGM}$	*20*	*L*	*Pulse* $H \to L \to H$	–
	Vpp	*21*	*+5V*	*+25V*	–
2764, 27128	$\overline{CE}$	*20*	*L*	*L*	*L*
	$\overline{OE}$	*22*	*L*	*X*	*L*
	$\overline{PGM}$	*27*	*H*	*Pulse* $H \to L \to H$	*H*
	Vpp	*1*	*+5V*	*+Vpp*	*+Vpp*
2564	*PD* / $\overline{PGM}$	*22*	*L*	*Pulse* $H \to L \to H$	–
	$\overline{CE1}$	*21*	*L*	*L*	–
	$\overline{CE2}$	*27*	*L*	*L*	–
	Vpp	*1*	*+5V*	*+25V*	–
27256	$\overline{CE}$	*20*	*L*	*Pulse* $H \to L \to H$	*H*
	$\overline{OE}$	*22*	*L*	*H*	*L*
	Vpp	*1*	*+5V*	*+Vpp*	*+Vpp*
27512	$\overline{CE}$ / *PGM*	*18*	*L*	*Pulse* $L \to H \to L$	–
	$\overline{OE}$ / *Vpp*	*20*	*L*	*+Vpp*	–

За програмирането на EPROM се предлага бърз (интелигентен) алгоритъм

на програмиране, който позволява максимално да се съкрати времето на програмиране при запазване на надеждността на записа. Той е показан на фиг. 9.4.



Фиг. 9.4. Алгоритъм за бързо (интелигентно) програмиране на EPROM.

Алгоритъмът предвижда програмиране на байт с импулси с продължителност по 1 *ms* и последваща проверка за верността на записа. Броят на импулсите се съхранява в програмно организиран брояч – *PC*. При установяване на правилен запис се извършва допълнително програмиране на байта с импулс, имащ

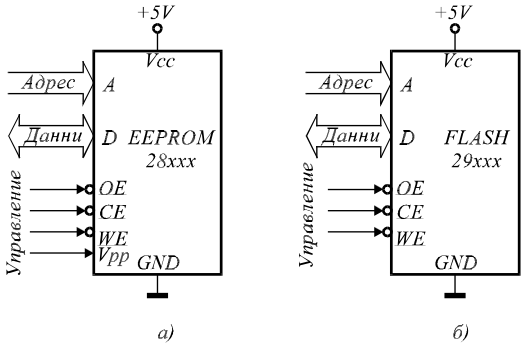
продължителност 3×*PC ms*. В повечето съвременни EPROM за ускоряване на програмирането е предвидена възможност то да се извършва при повишено захранващо напрежение *Vcc = +6V*.

Тъй като в околната светлина (слънчева, от луминесцентни лампи и др.) се съдържат ултравиолетови лъчи, е възможно при продължително изложение на EPROM на такава светлина да настъпи промяна на записаната в тях информация. Затова след запис прозорчето на чипа задължително се покрива с непрозрачна лепенка.

В микропроцесорните системи масово се използуват и електрически изтриваеми постоянни памети EEPROM. При тях изтриването на паметта се извършва с електрически импулс. По-старите от тях (серия 28xxx) имат същите функционални управляващи сигнали, както ултравиолетово изтриваемите PROM, и се програмират по същите алгоритми на същите програматори. Допълнително в програматорите, които поддържат и EEPROM, е предвидена възможност за електрическото им изтриване.

9.2. Програмиране на FLASH памети

Огромен скок в развитието на PROM е появата на FLASH технологията за EEPROM. FLASH паметите не се нуждаят от отделен програматор. Техният запис (програмиране) се извършва в самата микропроцесорна система, където са вградени. Функционалните им управляващи сигнали съответствуват на тези на статична оперативна памет (фиг. 9.5) – притежават сигнал $\overline{WE}$ за разрешение на записа.



Фиг. 9.5. Типични управляващи сигнали за запис на FLASH EEPROM: а) с външно програмиращо напрежение; б) с вграден генератор на програмиращо напрежение.

Първите FLASH памети (от серията 28Fxxx) приличат на старите EEPROM по това, че се нуждаят от външно програмиращо напрежение *Vpp* и допълнително трябва да се предвиди управлението му в режим на запис и изтриване. Съвременните FLASH памети (от серията 29Fxxx) вътрешно си изработват

програмиращото напрежение от захранващото напрежение *Vcc* .

Записът и изтриването на FLASH паметите е организирано от вграден в схемите автомат (машина на състоянията), която се управлява от специален команден регистър. Командният регистър сам по себе си не заема никакво адресируемо пространство. Той се зарежда с използването на специални поредици от команди, съдържащи адресна и даннова информация. Поредиците са уникални и е практически невъзможно те да се формират случайно при нормалната работа на микропроцесорната система. Като пример в табл. 9.2. са показани поважните команди и пораждащите ги поредици за FLASH паметта Am29F010.

Табл. 9.2. Основни команди на FLASH памет Am29F010.

Цикъл	I		II		III		IV		V		VI	
Команда	A	D	A	D	A	D	A	D	A	D	A	D
Четене/ Начално установяване	5555	AA	2AAA	55	5555	F0	RA	RD				
Запис на байт	5555	AA	2AAA	55	5555	A0	PA	PD				
Изтриване на чипа	5555	AA	2AAA	55	5555	80	5555	AA	2AAA	55	5555	10
Изтриване на сектор	5555	AA	2AAA	55	5555	80	5555	AA	2AAA	55	SA	30

Командният регистър се записва при $\overline{WE} = 0$, $\overline{CE} = 0$ и $\overline{OE} = 1$. Адресите се стробират по спадащия фронт на $\overline{WE}$, а данните – по нарастващия фронт на $\overline{WE}$. Операцията "запис" се задава еднократно, а вътрешната логика провежда програмирането, като краят на програмирането се определя от състоянието на бит D7, когато той е равен на записваната информация в този бит. Така схемата се връща в режим на четене.

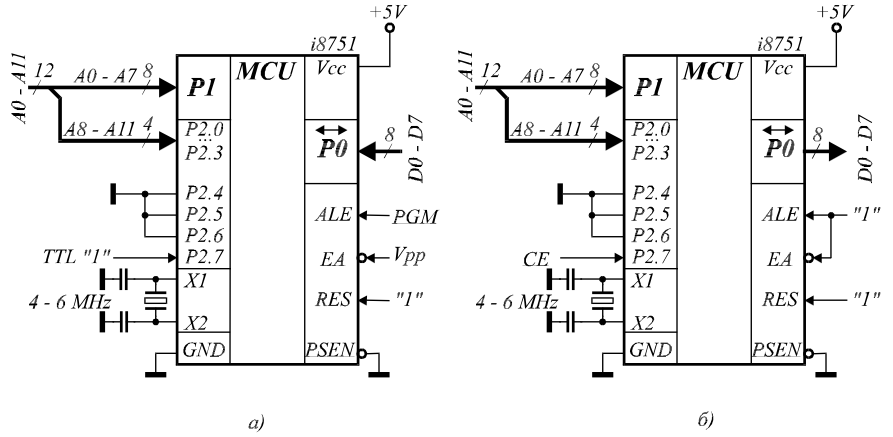
9.3. Програмиране на едночипови микрокомпютри

При едночиповите микрокомпютри постоянната памет е вградена вътре в самия чип. Той може да бъде ултравиолетов изтриваем PROM, EEPROM или FLASH. Съществуват различни технологии за програмирането на едночипови микрокомпютри, които могат да бъдат обобщени най-общо в три технологии: програмиране на програматор, самопрограмиране на стенд и софтуерно програмиране по сериен канал.

Първата технология се отнася най-вече за едночиповите микрокомпютри, притежаващи вградена ултравиолетово изтриваема постоянна памет. Те се изтриват предварително, след което се поставят в програматор за EPROM, поддържащ програмирането на този тип микроконтролери. Типичен пример за този тип микроконтролери е i8751, чието програмиране е показано на фиг. 9.6.

За да се програмира, i8551 трябва да получи тактов сигнал от 4 до 6 MHz, за да може вътрешната системна магистрала да поеме подаваните адреси и данни за програмиране. Адресът, на който се програмира даден байт, се задава на Port

1 и на изводите от *P2.0* до *P2.3* на Port 2. В същото време данните се подават на Port 0. Изводите от *P2.4* до *P2.6* и $\overline{PSEN}$ трябва да бъдат в логическа *0*, докато на входа *RES* трябва да има ниво по-високо от +2,5 *V*. По време на програмирането на байт на извода $\overline{EA}/Vpp$ трябва да има приложено напрежение +21*V*, а на извода $ALE/\overline{PGM}$ се задава програмиращ импулс с ниско ниво и продължителност 50 *ms*.



Фиг. 9.6. Програмиране на микроконтролера i8751: а) режим на запис; б) режим на проверка.

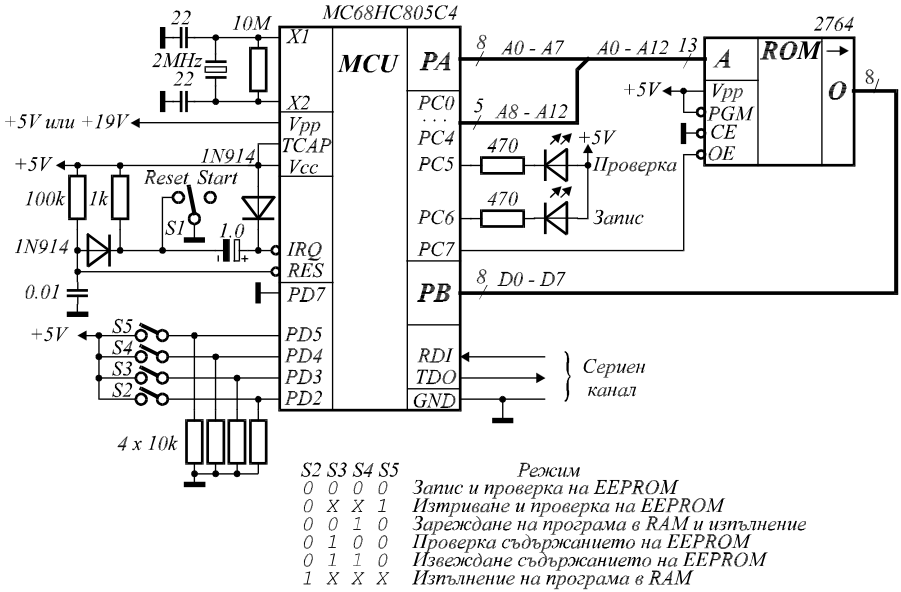
Ако битът за сигурност не е програмиран, съдържанието на постоянната памет на i8751 може да бъде прочетено за проверка на правилния запис. Състоянието на управляващите сигнали при четене е същото както и при запис, с тази разлика, че е свалено програмиращото напрежение, като изводът $\overline{EA}/Vpp$ се държи в логическа *1* и същевременно на *P2.7* се подава сигнал $\overline{CE}$ за разрешение на чипа с ниско ниво. В това състояние данните могат да бъдат прочетени от Port 0.

Самопрограмирането е характерно за по-модерните едночипови микрокомпютри. Те се програмират на специален стенд, в който е пренесен чрез друга постоянна памет (най-често EPROM) обектовият код на работната програма. Ултравиолетово изтриваемите едночипови компютри трябва предварително да бъдат изтрити, докато тези с електрически изтриваеми постоянни памети могат да бъдат изтрити на самия стенд. Този тип микрокомпютри притежават вграден фирмен софтуер, който им дава възможност да изчитат паметта, пренасяща обектовия код на програмата и да го запишат във вътрешната си програмируема постоянна памет. Такъв начин на програмиране е използуван при едночиповите микрокомпютри на Motorola от серията MC68HC705 и MC68HC805. Първите са

с ултравиолетово изтриваема постоянна памет, а вторите – с електрически изтриваема. На фиг. 9.7 е представен пример с програмиращия стенд за микроконтролера MC68HC805C4.

Режимът на работа на стенда се определя от самия микроконтролер според подадената му двоична комбинация на изводите му от *PD2* до *PD5*. На специален извод *Vpp* се подава програмиращото напрежение, което при по-старите версии е +19 *V*, докато при новите е +5 *V* (вътрешно се изработва високото програмиращо напрежение).

Операцията започва като, при изключени захранващо и програмиращо напрежение, в стенда се поставят едночиповият микрокомпютър и пренасящата обектовия код постоянна памет. На входовете се задава операцията, която ще се провежда и след това се включват захранващото и програмиращите напрежения. Старт на операцията се задава чрез подаване на напрежителен импулс на входа *IRQ* по-висок от +6 *V* (около +8 *V*). В показания стенд това става чрез превключване на ключа *S1* в дясна позиция.

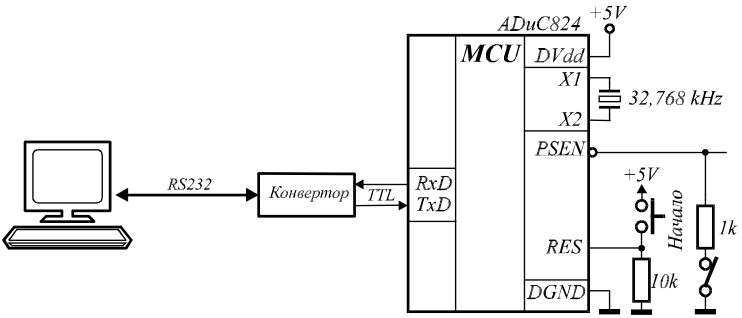


Фиг. 9.7. Стенд за паралелно самопрограмиране на едночипов микроконтролер MC68HC805C4 от междинен EPROM.

Вграденият фирмен софтуер в MC68HC805 позволява автоматично извършване на зададените от превключвателите операции, посочени на фиг. 9.7. При операцията "Запис и проверка на EEPROM" съдържанието на външния 8K

EPROM се копира във вътрешния EEPROM на микроконтролера. Адресите, които не се отнасят до EEPROM масива, остават незасегнати. Непрограмираните байтове във външния EPROM трябва да съдържат данни `FF`. Процесите на запис и успешната проверка се индицират чрез светодиоди. MC68HC805 може да изведе цялото съдържание на вътрешния EEPROM през серийния канал. Данните се извеждат в NRZ формат (стартов бит, осем бита данни и един стопов бит). Първият извеждан адрес е `0020`, а последният – `1FFF`, като неизползваните области се прескачат. При 2 *MHz* кристал за осцилатора скоростта на предаване е 4800 *bit/s*.

Софтуерното програмиране по сериен канал е заложено в съвременните едночипови микрокомпютри. То позволява програмирането да се извърши на място, в самото устройство, при вграждане на минимални допълнителни приспособления. Такива едночипови микрокомпютри притежават специален вграден фирмен софтуер, чрез който програмирането им се извършва по сериен канал от PC или работна станция. Като пример на фиг. 9.8 е показано серийното програмиране на микроконтролера на Analog Device ADuC824 (с процесорно ядро на 8031). Режимът на серийно програмиране се активира, като изводът $\overline{PSEN}$ се свърже към маса през съпротивление 1 $k\Omega$ по време на сигнала за начално установяване.



Start ID	№ Data Bytes	Data1	Data	Checksum

б)

Фиг. 9.8. Програмиране на микроконтролера AduC824 по сериен канал: а) схема на свързване; б) формат на програмиращите пакети информация.

Веднага след влизането в режим на програмиране вграденият сериен адаптер автоматично се конфигурира за приемо-предаване със скорост 9600 *bit/s* и от микроконтролера се изпращат 25 байта със служебна информация. След като ги получи, програмиращата станция може да започне изпращането на информа-

цията към микроконтролера. Информацията се предава в ASCII код, в пакети със следната структура:

Start ID – начало на пакета, състоящ се от байтовете 07hex и 0Ehex;

№ Data Bytes – общ брой на данновите байтове, който може да бъде между 1 и 25;

Data1 – команда към микроконтролера;

Data – байтове с данни;

Checksum – контролна сума на пакета.

Командите към микроконтролера са няколко на брой, но за целите на програмирането се използуват основно две – 'C' (43hex) за изтриване на програмната памет и 'W' (57hex) за програмиране на блок от програмната памет.

Командата 'C' не съдържа поле Data. При командата 'W' първите три байта от полето Data съдържат началния адрес, от който започва програмирането на следващите в полето Data байтове. След запис на първия байт адресът за запис автоматично се увеличава с единица и следващият байт се записва на следващия адрес. Това продължава до края на пакета.

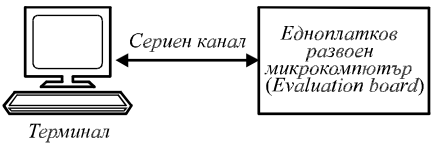
След всеки пакет от страна на микроконтролера могат да бъдат върнати ACK (положителен отговор) или NAK (отрицателен отговор). Отрицателен отговор се връща, когато пакетът не е приет правилно или когато информацията не е записана.

10. СИСТЕМИ ЗА РАЗВИТИЕ

Системите за развитие (Development system) представляват микропроцесорни и микрокомпютърни конфигурации, снабдени със средства за разработка и настройка на програмната и апаратната част на микропроцесорни системи. По-настоящем могат да се определят два типа системи за развитие – малки развойни системи и мощни развойни станции.

10.1. Малки развойни системи

В малките развойни системи съществува голямо разнообразие. Широко разпространено устройство за проектиране и настройка е едноплатковият развоен микрокомпютър (Evaluation board). Той представлява основна типична конфигурация на определена микропроцесорна система, която може да се използува като прототип на устройство, за което да се разработи и настрои работна програма. В типичния случай в едноплатковия развоен микрокомпютър се оставя свободно пространство за допълнителни интегрални схеми. Поради огромното разнообразие на задачите, които се решават с определена микропроцесорна система, едноплатковият развоен микрокомпютър обикновено е специализиран за определен тип задачи. Той съдържа основното микропроцесорно ядро и допълнителни схеми за решаването на този кръг задачи (например за обработка на аналогова информация). Най-често той комуникира с оператора по сериен канал чрез терминал или работна станция – фиг. 10.1.



Фиг. 10.1. Типична връзка с едноплатков развоен микрокомпютър.

Въпреки че е трудно да се постави граница между големия брой малки системи за развитие, следващото по-съвършено средство се нарича комплект за настройка (Evaluation kit). Тези системи могат да решават по-широк кръг задачи. Те могат да работят самостоятелно, като за целта те притежават собствена клавиатура и индикация, но могат да работят и свързани по сериен канал към отделен терминал или PC. Тези системи са полезни за бързо експериментиране и за целите на обучението.

Малките развойни системи работят под управление на програма, наречена Монитор (Monitor). Тази програма осъществява комуникацията с оператора и реализира функциите за настройка и диагностика. Независимо от това дали комплектите за настройка работят самостоятелно или комплексно с PC, мони-

торната програма притежава две основни части – комуникатор и команден интерпретатор.

Комуникаторът осъществява връзката с оператора и е апаратно зависим от използувания тип на връзката (клавиатура, индикация, отдалечен канал и др.).

Командният интерпретатор дешифрира и изпълнява командите на оператора. Той също може да бъде апаратно зависим (например от начина, по който са реализирани функциите за поставяне точки на прекъсване и трасиране на програми).

Макар че всички команди на мониторната програма обслужват цялостната настройка и диагностика на микропроцесорното устройство, те могат да бъдат условно разделени на две групи – команди, които обслужват основно диагностиката на апаратната част, и такива, които обслужват предимно диагностиката на програмната част на микропроцесорната система.

Основните команди, които обслужват диагностиката на апаратната част, могат да бъдат специфицирани, както следва:

– визуализация на клетка с възможност за промяната й;
– визуализация на масив от адресното пространство;
– тест на оперативна памет;
– преместване и сравняване на масиви.

Като команди, обслужващи предимно диагностиката на програмната част, могат да бъдат посочени:

– зареждане и извеждане на обектов код във и от паметта;
– поставяне и изтриване на точки на прекъсване;
– стартиране на програма;
– трасиране на инструкции;
– асемблиране и реасемблиране.

Мониторните програми могат да притежават и допълнителни служебни команди, като модифициране на адресното поле, преконфигуриране на серийния интерфейс и др. Практикува се и изграждането на мониторни програми, които не изискват допълнителен хардуер за реализиране на функциите си и като такива могат да бъдат вграждани в готовата система, превръщайки я в прототип. Тъй като всяка мониторна програма изиска определено количество памет ROM и RAM от микропроцесорната система, където е вградена, целесъобразно е колкото се може повече команди и задачи да се преместят на PC, където ресурсите са практически неограничени. В този случай програмата на PC се нарича отдалечена програма за настройка (remote debugger program).

10.2. Развойни станции

Съвременните развойни станции представляват микрокомпютърни конфигурации, снабдени със специализирани програмни и апаратни средства за разработка и настройка (Debugging tools) на микропроцесорни системи. Апаратните

и програмните ресурси на развойната станция се управляват от операционна система. Отделните специализирани програмни средства се зареждат за изпълнение при използването им.

С появата на първите микропроцесори всяка от фирмите, произвеждащи микропроцесорни фамилии, се стараеше да ги обезпечи с развойни станции. Те се изграждаха със същите микропроцесори, за чиято развойна дейност бяха предназначени. Такива бяха системите INTELLEC на Intel, EXORCISER и EXORMAX на Motorola и др. Голямото разнообразие на съществуващите микропроцесори доведе до появата на универсални развойни станции, предназначени за разработка и емулиране на голям брой микропроцесори и с възможности за едновременно обслужване на няколко операторски пулта, като например системата HP64000 на Hewlett-Packard.

Развитието на компютърната техника и появата на мощни и евтини персонални компютри, промени стратегията на производителите на развойни станции. Сега, те се стараят преди всичко да произвеждат програмни и апаратни атрибути за широко разпространените персонални компютри, които да ги превръщат в мощни развойни станции.

10.3. Програмни атрибути на система за развитие

Освен стандартните програми, които организират и обслужват операционната система, системата за развитие притежава специфични за нея програмни продукти, които определят предназначението ѝ като развойно средство за разработка на микропроцесорни системи. Тези специфични програми могат да бъдат разделени условно на две групи: програмни продукти за разработка на програмно осигуряване за микропроцесорни системи и програмни продукти за настройка. Програмите за разработка могат да бъдат обобщени в следния порядък.

Текстови редактор – това е първата програма, до която прибягва съставителят на микропроцесорни програми. С текстовия редактор се набира текстът на микропроцесорната програма при използването на конкретен език. Освен набирането на нов текст с него могат да се правят и корекции на стари текстове.

Исторически първи са се появили редовите редактори. При тях измененията в текстовия файл се нанасят след като операторът въведе целия ред. Някои от редовите редактори работят само с номерирани редове. Редовете се съхраняват в паметта по нарастващ ред на номерата им. Ако трябва да се нанесат корекции в някой ред, операторът написва новото съдържание на реда, поставя му същия номер и стария ред се заменя с новия. За да може между старите редове в програмата да се вмъкват нови редове, е необходимо първоначалната номерация на редовете да се нанася с някаква стъпка – например 10. Така, ако първоначалната номерация на редовете е 10, 20, 30 и т.н., нов ред с номер 26 ще бъде поставен между редове 20 и 30. Съществуват и редови редактори, които могат да работят

и с неномерирани редове. При тях с помощта на курсора операторът задава къде се намира новият ред и той заема съответното място в паметта. Понастоящем редовите редактори се използуват само в мониторните програми на малките системи за развитие

В съвременните системи за развитие се използуват екранни редактори, наричани още символни, тъй като редактираният текст динамично се изобразява на екрана, а корекция във файла се прави при всеки набран символ. С екранните редактори се работи значително по-удобно, отколкото с редовите; техните възможности са по-големи, затова и почти изцяло са изместили редовите. В онези системи за развитие, при които конзолата е реализирана като отделен видеотерминал, се изисква високоскоростен и високоинтелигентен обмен на информацията между компютъра и видеотерминала.

Транслатори – след като микропроцесорните програми са набрани и записани в текстови файл, е необходимо те да бъдат преведени на езика на микропроцесора – на машинен език. За преобразуването на програми, написани на асемблерски език в програми на машинен език, се използува програма, наречена Асемблер. Обработваните от асемблера програми се наричат изходни програми, а получаващата се на машинен език програма се нарича обектна програма. Последната автоматично се запомня като обектен файл. Асемблерът включва в себе си изпълнителни команди, които директно се превеждат на машинен език и псевдокоманди, които не се превеждат, а се възприемат като директиви за изпълняване на определени функции.

По начина на работа асемблерите се делят на еднопасови, двупасови и многопасови. Най-често се използуват двупасови асемблери. Те извършват двукратен преглед на текстовия файл. Първият преглед на изходния текст се използува за присвояване на адреси на всички етикети в програмата. Етикетите заедно с присвоените адреси се записват в таблица. При втория преглед на програмата се извършва транслирането и в обектна програма. Еднопасовите асемблери се използуват при мониторните програми на малките системи за развитие.

Съществуват няколко вида асемблери, сред които най-разпространени са макроасемблерът, кросасемблерът, резидентният асемблер и метаасемблерът.

Макроасемблерът позволява използуване в изходния текст на макрокоманди и макроопределения. Макроопределението представлява именувана последователност от изпълнителни команди и директиви със символични параметри. Операторът поставя в текста на програмата макроопределението, за да замести поредицата от предписаните му последователности, но винаги с указване на конкретните параметри. Макрокомандата представлява макроопределение с указани конкретни параметри. Замяната на макрокомандата с предписаната й последователност от команди се извършва от асемблера в процеса на транслиране на програмата. Полученият обектов файл по нищо не се отличава от файл, получен без използуването на макрокоманди. Макрокомандите и макроопреде-

ленията са въведени в макроасемблера единствено за удобство на програмиста.

Кросасемблерът е вид асемблер, който изготвя машинен код за друг микропроцесор, различен от този, с който е изградена развойната система. Кросасемблерите често се пишат на езици от високо ниво, като по този начин стават машинно независими.

Резидентният асемблер е пълна противоположност на кросасемблера, тъй като той създава машинен код за процесора, с който е изградена развойната система. Така транслираната програма може да се зареди и изпълни в средата на системата за развитие.

Метаасемблерът е най-универсален асемблер, тъй като може да се използува за всеки микропроцесор. За да се работи с него обаче е необходимо предварително да му бъдат указани правилата на асемблирането за асемблерския език на конкретния микропроцесор.

Освен на асемблерски език програмите за микропроцесорни системи могат да бъдат написани и на езици от високо ниво като Fortran, Pascal, C и др. Превеждането на изходна програма, написана на език от високо ниво, се извършва от програма, наречена Компилатор. Превеждането се извършва на асемблерски език или директно на машинен език. Основният недостатък на използуването на езици от високо ниво е, че компилаторите по правило генерират машинна програма, която отстъпва по ефективност на програма, написана от опитен програмист на асемблерски език.

Макар и все по-рядко, при превеждането на програми, написани на език от високо ниво, се използуват интерпретатори, които не транслират изходната програма в асемблерски или машинен език. Интерпретаторът само обезпечава пооператорно, непосредствено изпълняване на изходната програма. Недостатък на интерпретаторите е малката скорост на процеса на интерпретиране на програмата. При това тя се интерпретира всеки път, когато се стартира за изпълнение. Интерпретатори продължават да се използуват при микропроцесорните системи на промишлените контролери.

Свързващ редактор – повечето от транслаторите в системите за развитие могат да изготвят премествaем обектов код, т.нар. обектов файл. Той представлява премествaема програма, която може да бъде заредена за изпълняване в указано място от паметта на микропроцесорната система. Когато приложната програма за дадена микропроцесорна система е със значителна дължина, изготвя се на различни езици или от няколко програмисти, тя се разработва на части. Всяка част представлява отделен обектов файл. Свързващият редактор е предназначен за обединяване в зададена последователност на тези обектови файлове и преобразуването им в абсолютен файл, съдържащ цялата програма в абсолютни адреси, която е готова за изпълнение. Образуваните абсолютни файлове служат за настройване на програмата, а след настройката й – за програмирането им в постоянна памет.

Обикновено всяка фирма, произвеждаща развойни програмни продукти, осигурява в комплект текстови редактор, транслатори и свързващи редактори. Съвременните продукти са насочени към графичните интерфейси на персоналните компютри и са обединени в единна среда (Shell), което ги прави удобни и лесни за работа.

По-маловажни програмни продукти, обслужващи настройването на програмното осигуряване за микропроцесорна система, са следните:

Зареждаща програма – тя е предназначена да поеме готовия абсолютен файл с програмата за микропроцесорната система и да го зареди за изпълнение в паметта на системата за развитие. Зареждащата програма се използува само ако централният микропроцесор на системата за развитие е съвместим с микропроцесора, за който е разработвана приложната програма. В повечето развойни системи зареждащата програма е команда на операционната система.

Настройваща програма (Debugger program) – тя се използува само тогава, когато разработената програма е заредена за настройване в системата за развитие. Тя служи за локализиране и поправяне на грешки в заредената потребителска програма. Подобно на мониторната програма на емулатор настройващата програма има възможност да осигурява достъп на потребителя до клетка или масив от паметта, да поставя точки на прекъсване в потребителската програма и да я стартира, да трасира участъци от нея и др. В повечето системи за развитие настройващата програма е оформена като самостоятелен системен файл в операционната система.

Симулатор (Simulator) – това е програмен продукт, който като правило работи на развойна станция. Компютър на развойната станция използува своя процесор, за да симулира инструкциите на прототипния микропроцесор или микроконтролер. Симулаторът притежава два режима: команден режим и режим на изпълнение на програма. В командния режим се зарежда кодът на програмата. Възможно е изпълнение на единична инструкция. При всички случаи симулаторът се връща в командния режим при спиране на изпълнението. Могат да бъдат проверени и модифицирани регистри и клетки от паметта и операторът може да прецени дали се удовлетворяват времевите характеристики.

Комуникационни програми – предназначени са за осъществяване на връзка между компютъра на развойната система и включените в състава й автономни уреди, като: емулатори, логически и сигнатурни анализатори, програматори и др., включително и малки развойни системи и прототипи. Физически комуникацията се осъществява по стандартни интерфейси RS232, IEEE488, USB и др.

Системата за развитие може да включва още редица програмни продукти, участвуващи в процеса на разработка и настройка на микропроцесорни системи, като: програми за управление на логически и сигнатурен анализатор, програма за програмиране на постоянни памети и др. Някои програмни продукти са алтернативни на други. Например зареждаща и настройваща програма не се из-

ползуват при работа с автономен емулатор, който от своя страна се нуждае от комуникационна програма и т.н. Някои от изброените програмни продукти може да не съществуват в явен вид и да не са обособени като самостоятелни, а да влизат в състава на други. Въобще разнообразието на програмни продукти за системите за развитие е много голямо и единствени критерии за тяхната употреба могат да бъдат удобството при работа и възможностите, които предлагат на потребителя.

За проектиране на микропроцесорни системи се използуват и много други програмни продукти, като: системи за проектиране и анализ на схеми и печатни платки, системи за работа с програмируеми логически матрици и др.. Но тяхната употреба е за електрониката като цяло, поради което те не се причисляват към спецификата на системите за развитие.

10.4. Апаратни атрибути на система за развитие

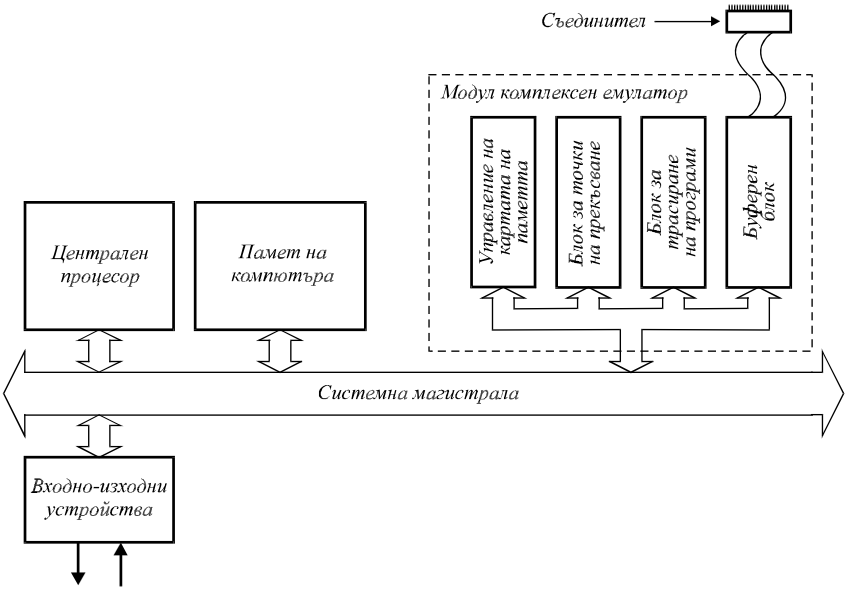
Освен стандартната микрокомпютърна конфигурация системата за развитие притежава специфични за нея апаратни атрибути, които определят предназначението и като развойно средство за настройка на микропроцесорни системи. По правило тези атрибути се изискват и използуват при наличието на опитен образец (апаратен прототип). Те могат да бъдат комплексни и автономни. Автономните уреди могат да работят и самостоятелно, а свързването им в системите за развитие се извършва по стандартен интерфейс. Комплексните уреди се изграждат в системата за развитие и могат да работят само в нейния състав. Поважните апаратни атрибути на системите за развитие са посочени по-долу.

Вътрешносхемен микропроцесорен емулатор – Комплексният емулатор се изгражда в системата за развитие с използуване на нейните апаратни ресурси. Когато процесорът на развойната система е съвместим с този на прототипа, системата като цяло започва да играе ролята на емулатор, а връзката й с прототипа се осъществява през буферен модул, свързващ системата за развитие с микропроцесорния цокъл на настройваната система. Този буферен модул се нарича още емулаторен преходник.

Така изграденият вътрешносхемен микропроцесорен емулатор максимално използува ресурсите на развойната система (процесор, памет и др.), а притежава само онези специфични блокове, които са присъщи само за микропроцесорен емулатор – управление на картата на паметта, поставяне на точки на прекъсване, трасиране на програми и т.н.

С такава структура се дава възможност да започне съвместно настройване на апаратната и програмната част дълго преди окончателното завършване на апаратната част на разработваното устройство. Последната се включва в системата постепенно, при което се изключват съответните ресурси на системата за развитие и техните функции започват да се изпълняват от реалната настройвана система. На практика в такава система за развитие един и същи процесор из-

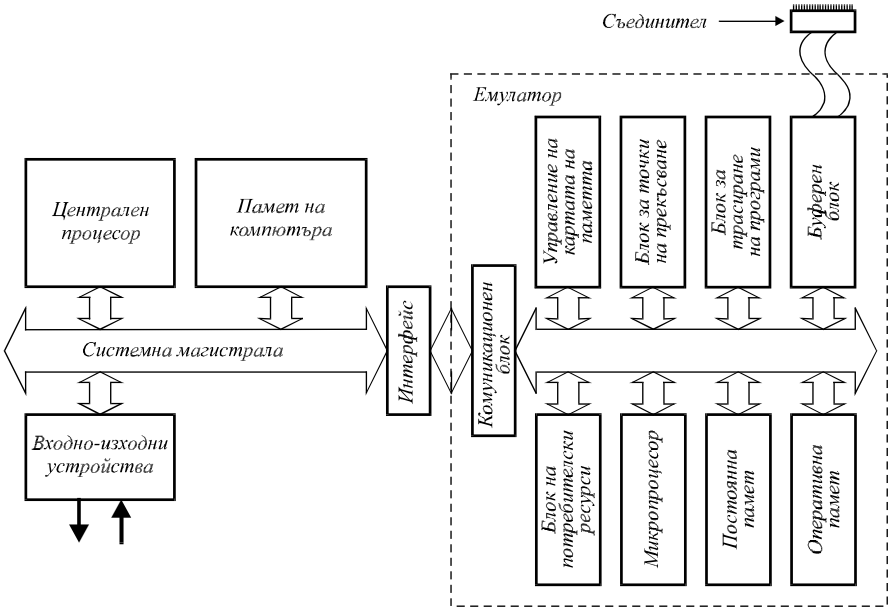
пълнява и системните, и емулационните функции. Емулационните дейности са оформени като системна команда. Структурната схема на реализиран по този начин вътрешносхемен микропроцесорен емулатор е показана на фиг. 10.2.



Фиг. 10.2. Структурна схема на комплексен емулатор, използуващ процесора и апаратните ресурси на системата за развитие.

В много случаи комплексният емулатор се изгражда със собствен микропроцесор, който е поставен в подчинено положение от компютърния процесор. Връзката помежду им се осъществява по системната магистрала. Налице е силно свързана двукомпютърна система, при която системният микропроцесор организира процеса на разработката, като изпълнява редактирането, асемблирането и др., а емулаторният е зает само с настройката и диагностиката на разработваното устройство. В този случай емулаторът разполага с всички необходими блокове за провеждането на емулирането, както и със собствена мониторна програма – емулаторът е полуавтономен.

Структурната схема на такава организация на емулатора е показана на фиг. 10.3. Тази структура е по-универсална, тъй като могат да се емулират съвсем различни от компютърния процесор микропроцесори. Създава се възможност в една система за развитие да се включват различни емулатори и да се провежда настройка и диагностика на микропроцесорни системи с различни микропроцесори.



Фиг. 10.3. Свързване на полуавтономен емулатор в система за развитие.

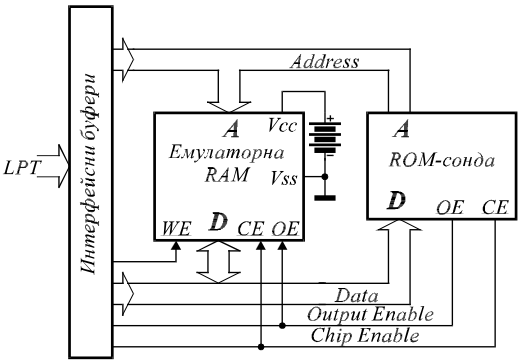
Логически анализатор – той по принцип е сложно и скъпо устройство, затова и неговото присъствие в състава на система за развитие е само пожелателно. Най-широко разпространение е добила настройката на микропроцесорно изделие с трасиране на работната програма. С поставянето на точки на прекъсване и с трасирането на програми обаче работата на настройваната микропроцесорна система излиза от рамките на реалното време. Затова и когато една микропроцесорна система се проектира да работи в реално време, нейното настройване много трудно може да се проведе чрез трасиране на работната ѝ програма. Логическият анализатор е този, който може да извършва диагностика на микропроцесорна система, работеща в реално време, т.е. да извършва трасиране на нейната програма в реално време.

Най-често съвместно със системите за развитие се използуват автономни, универсални логически анализатори, представляващи напълно самостоятелни прибори. Получиха популярност и комплексни логически анализатори, които, вградени в системата за развитие, я превръщат в мощен логически анализатор.

В някои от по-старите системи за развитие са вградени комплексни магистрални логически анализатори, предназначени да анализират работата на системата, когато тя се използува като прототип на микропроцесорно изделие. Те вземат сигнали за анализ от системната магистрала на системата за развитие.

Тези сигнали също както и микропроцесорните сигнали съдържат в концентриран вид работата на системата (а оттам и на настройваната система). Магистралните анализатори са предназначени основно за анализ на работата на многопроцесорни системи.

ROM-емулатор – този тип емулатори също могат да се изграждат като комплексни и автономни. Обикновено комплексните ROM-емулатори са присъщи на малките развойни системи, докато при развойните станции се използуват предимно автономни емулататори. Като евтини и удобни за работа в комплекта на развойна станция се предлагат полуавтономни ROM-емулатори – фиг. 10.4. Те съдържат оперативна памет (съответствуваща по обем на емулираната постоянна памет) с батерийно захранване, която се зарежда с обектовия код на емулираната програма от развойната станция (най-често през принтерния порт). След зареждането тя се пренася и се включва в тестваната система на мястото на емулираната постоянна памет.



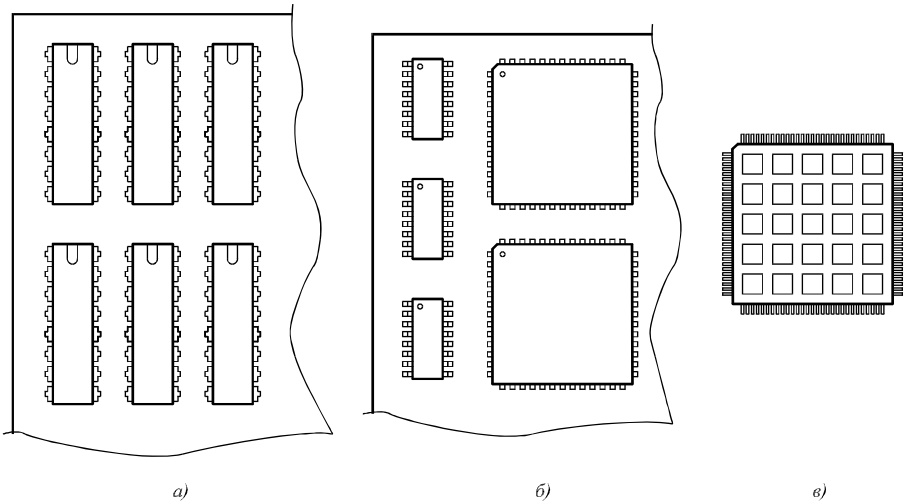
Фиг. 10.4. Полуавтономен вътрешносхемен ROM-емулатор.

Програматор. След настройването на работната програма на една микропроцесорна система в повечето случаи тя се записва в постоянна енергонезависима памет, която се поставя в разработваната система. Само след това приложната програма се изпълнява в нейната окончателна работна среда. За нейното програмиране в състава на системата за развитие е необходимо да бъде включен автономен или комплексен програматор. Обикновено неговото програмно осигуряване е включено като файл в състава на операционната система.

В състава на системата за развитие могат да бъдат включени и други устройства, спомагащи за извършването на настройка и диагностика на микропроцесорни системи – тестер за въздействия, сигнатурен анализатор и др.

11. ГРАНИЧНА СКАНИРАЩА ЛОГИКА

През 1985 г. няколко европейски и североамерикански фирми се организират в т. нар. Обединение за тестови изследвания – Joint Test Action Group (JTAG). Тяхната основна задача е да се реши проблемът с производственото тестване на асемблирани печатни платки (Printed Circuit Board – PCB), които стават все по-сложни с използването на все по-малки и комплексни интегрални схеми. Техните разработки довеждат до създаването на интерфейса за тестови достъп (Test Access Port – TAP) и архитектурата на граничната сканираща логика. Решението е стандартизирано като IEEE Std. 1149.1-1990. Това е стандарт за включване на тестови точки вътре в самите интегрални схеми и протокол за обмен на информация с тях.

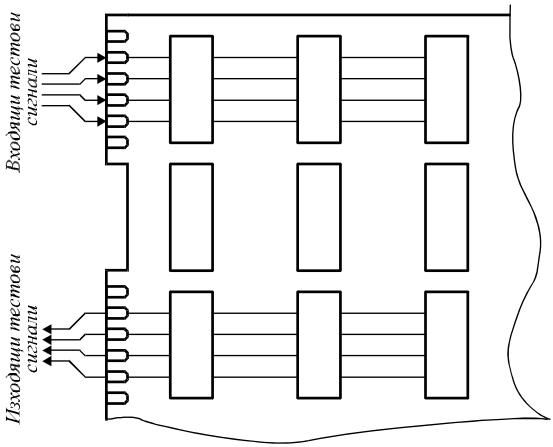


Фиг. 11.1. Сложност на печатните платки и физически достъп за тестване: а) минало; б) настояще; в) бъдеще.

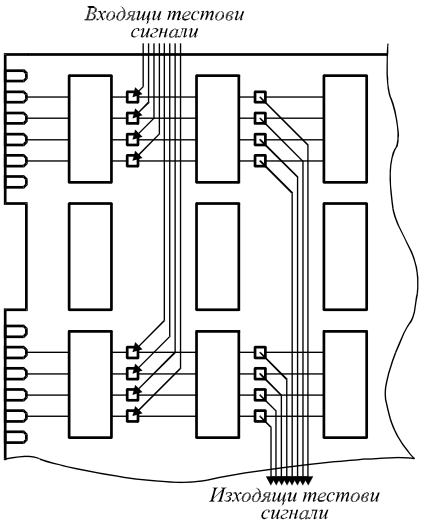
Провеждането на тест на асемблирани печатни платки цели откриването на дефекти (например къси съединения или прекъснати вериги) между интегралните схеми и другите компоненти на платките. Достигането на тази цел става все по-трудно с увеличаване на сложността и намаляване габаритите на интегралните схеми – вж. фиг. 11.1.

Един от основните методи за тестване на печатни платки използва специално проектиран на печатната платка тестови конектор или съществуващ функционален конектор, през който се въвеждат тестови сигнали и се отнемат реак-

циите на определени сигнали (фиг. 11.2). Способността на този метод за функционалното тестване обаче силно намалява с увеличаване на функционалната плътност на компонентите върху печатната платка.



Фиг. 11.2. Функционално тестване, използуващо изводите на конектор.



Фиг. 11.3 Функционалното тестване, използуващо контролни точки с контактни площадки върху платката.

Дори само първичните входове и изходи да се използуват като тестови точки, сложността на пораждащата тестова последователност и на предизвиканата тестова последователност драматично нарастват с увеличаване на сложността на платката.

Тестването на печатни платки с използуването на контролни точки с контактни площадки върху платките (фиг. 11.3) е следващият опит да се подобри откриването на грешки в комплексните платки. Това тестване също силно намалява своите възможности с увеличаване на сложността и плътността на елементите върху печатната платка. Използуването на интегрални схеми с малка стъпка между изводите довежда до усложняване и оскъпяване на контактуващите сонди с контролните точки. В някои случаи

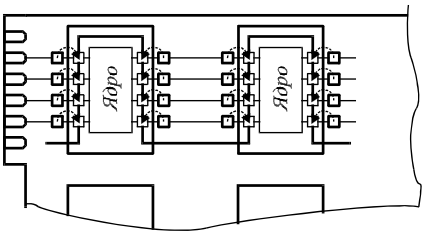
този физически достъп изобщо е невъзможен – например при мултичипните модули (MCM).

Увеличената функционална сложност на интегралните схеми довежда и до други проблеми с тестването. За да се тестват изходите на една интегрална схема, т.е да се накарат те да приемат определени състояния, функционирането на устройството често трябва да бъде манипулирано с дълга и сложна задаваща тестова поредица от сигнали на входовете на интегралните схеми. Същите проблеми съществуват и при провеждането на продължителен тест на входовете на интегралните схеми.

Идеята за гранично тестване е изградена на базата на тестването с контролни точки в платките. При граничното тестване обаче контролните точки са заместени с гранично сканирани клетки BCS. Тези "виртуални" точки са поставени вътре в самите интегрални схеми при техните входове и изходи, т.е. те се намират в краищата на връзките (фиг. 11.4). В резултат се реализират две основни предимства:

– към "вградените контролни точки" повече не е необходим физически достъп;

– продължителността на теста не зависи от сложността на интегралната схема.



Фиг. 11.4. Вграждане на диагностични контролни точки в изводите на интегралните схеми.

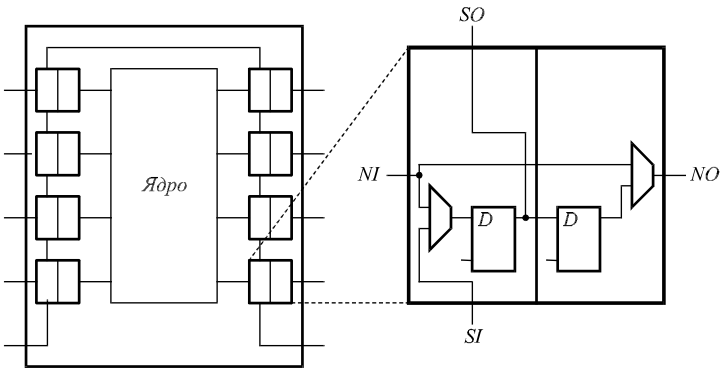
Накрая, за да се подобри и минимизира начинът на управление и наблюдаване на тези гранични сканиращи клетки, те са проектирани така, че да могат да бъдат свързвани последователно и да формират преместващ регистър с достъп от два извода – тестови информационен вход и тестови информационен изход. Управлението на граничната сканираща логика е изградено като машина на състоянията, която работи синхронно със сигнал за тестови такт, по който се синхронизират всички операции в логиката и под управлението на единствен сигнал – избор на тестови режим. Проектирани са допълнителни управляващи структури, регламентиращи действието на логиката. Схемата може да работи в нормален режим, при който тя изпълнява функционалното си предназначение, може да мине в режим на тест или диагностика, а също така са предвидени механизми за комбиниране на двата режима – нормална работа с едновременна диагностика.

11.1. Основа на граничното сканиране

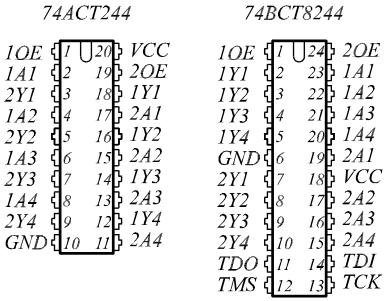
Основните клетки на гранично сканиращата логика са т.нар. гранични ска-

ниращи клетки (BSC – Boundary Scan Cell). Именно това са гъвкавите тестови точки, вградени във всеки цифров вход и изход на схемата. Граничната сканираща клетка разрешава на входовете и изходите да спазват предназначението си и да пропускат информацията през себе си. Граничната сканираща клетка може да бъде използувана за управление и контролиране състоянието на съответния извод, осигурявайки информация за извода чрез серийния вход (SI) или приемайки информация през серийния си изход (SO). Фиг. 11.5 показва типичната структура на гранична сканираща клетка. Всяка гранично сканираща схема съдържа минимум четири допълнителни изводи:

– тестови информационен вход (TDI – Test Data Input);
– тестови информационен изход (TDO – Test Data Output);
– тестови избор на режим (TMS – Test Mode Select);
– тестови такт (TCK – Test Clock).



Фиг. 11.5. Типична структура на гранична сканираща клетка.

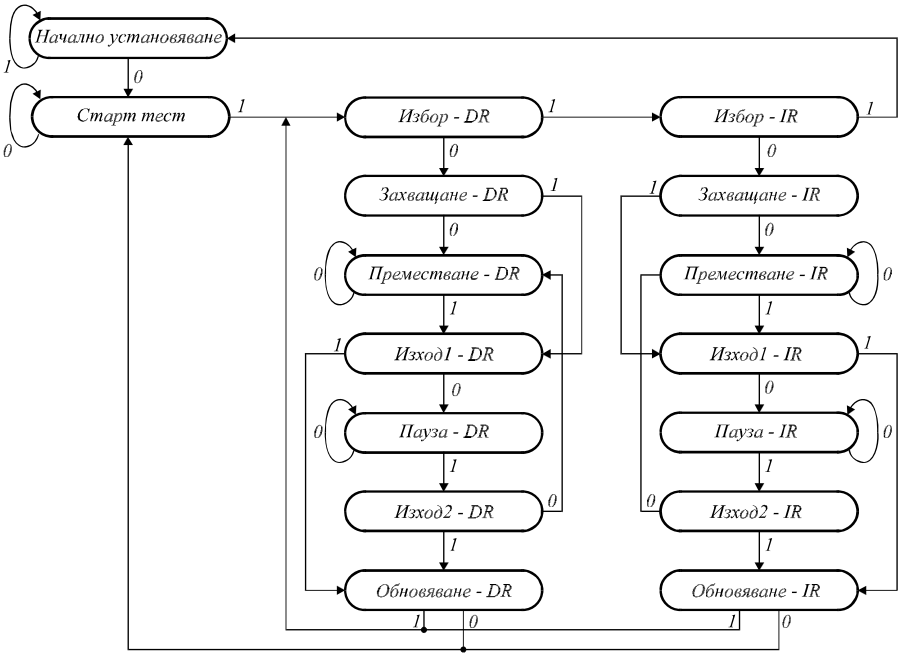


Фиг. 11.6. Разположение на изводите на нормален буфер ‘244 със съответствуващия му ‘8244, с вградена гранична сканираща логика.

TDI и TDO изводите обслужват пътя, по който последователната информация влиза и излиза във и от схемата. TMS и TCK изводите управляват състоянието на схемата, поставяйки я или в тестови, или в нормален режим на работа. В някои случаи е налице пети извод TRST, който може да поставя в начално състояние тестовата логика и да връща схемата в нормален режим на работа. Фиг. 11.6 показва четирите основни допълнителни извода, въведени в октална (състояща

се от осем еднакви елемента с общо управление) схема на един гранично сканиращ буфер.

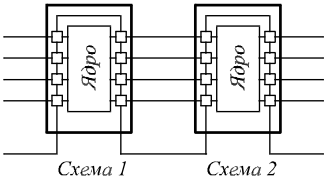
Сигналите TMS и TCK управляват порта (интерфейса) за тестови достъп (TAP), който е организиран като машина на състоянията. TAP контролира серийното сканиране на информацията за инструкцията или данните през схемата и тяхното съвместяване с JTAG стандарта. Състоянията на TAP контролера са последователни, а преходът между тях се осъществява според състоянието на TMS по време на нарастващия фронт на TCK, както е показано на фиг. 11.7.



Фиг. 11.7. Диаграма на състоянията на машината на TAP контролера (условието за прехода е състоянието на TMS по време на нарастващия фронт на TCK).

Комуникацията с гранично сканираните схеми се извършва по сериен път чрез последователното свързване на серийните изходи TDO със серийните входове TDI. Този тип връзка е показана на фиг. 11.8. Тя позволява да се намалят до минимум физическите диагностични връзки със схемите. Тестването на връзка между две схеми, притежаващи вградена гранична сканираща логика, се извършва в сложна последователност. Първо, потребителят трябва последователно да помести данни (например само единици) в схема 1 през TDI и да изпълни инструкция, която позволява информацията да се изпрати към паралел-

ните изходи на схемата.



Фиг. 11.8. Последователна информационна връзка между две схеми с гранично сканиране.

Схема 2 трябва да бъде командвана да захване идващата на нейните паралелни входове информация, да прехвърли тази информация в съответстващата на всеки вход гранична сканираща клетка и след това последователно да изведе данните през JTAG конектора към външния тестер за инспектиране на резултата. Всяко несъответствие в информацията може да бъде обратно проследено и да бъдат локализирани изводите, където се е случила грешката.

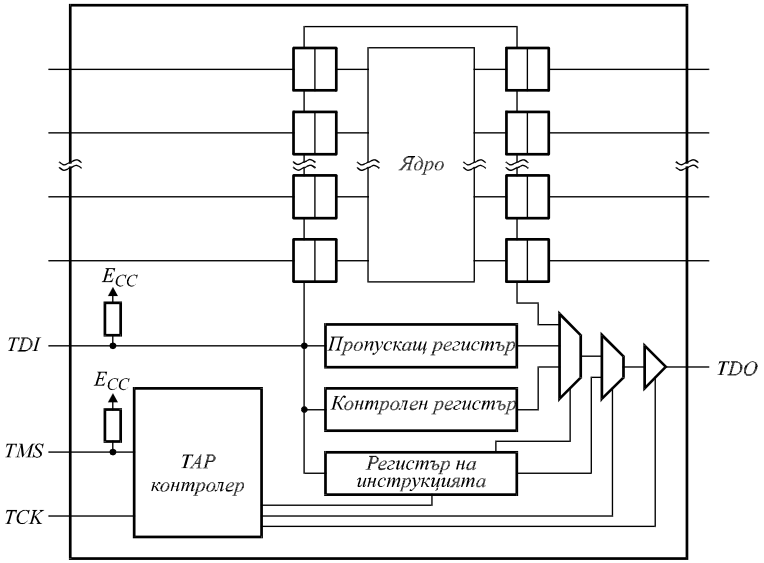
11.2. Организация на граничната сканираща логика

Както беше казано по-горе, граничната сканираща логика се основава на принципите на 4-битовия тестващ интерфейс, изграден според изискванията на стандарт IEEE 1149.1-1990 (JTAG). Този стандарт предопределя състава и организацията на граничната сканираща логика. С малки изключения, свързани с конкретните особености на отделните логически схеми, вградената гранична сканираща логика е идентична в различните схеми. Все пак в следващото изложение конкретните примери и разглеждания ще бъдат направени на базата на граничната сканираща логика в октална интегрална схема 74BCT8244 (осем неинвертиращи буфера).

Функционалната блокова схема на гранична сканираща логика е представена на фиг. 11.9. Взаимодействието на логиката с 4-битовия тестови интерфейс се осигурява от TAP контролера. Тестовите инструкции, тестовите данни и тестовите управляващи сигнали се изпращат по серийния канал през входа TDI и могат да бъдат изведени през изхода TDO. За TAP контролера са важни само два сигнала от тестовата шина – TMS и TCK. Той поема синхронизиращия сигнал TCK и сигнала за контрол на състоянието TMS и генерира подходящи вътрешносхемни управляващи сигнали за граничните сканиращи структури вътре в схемата. Входящите данни се стробират по нарастващ фронт на TCK, а изходящата информация се формира по падащия фронт на TCK. Тази алтернативност във входните и изходните операции осигурява възможност за възприемането на данните при тяхната валидност за половината на TCK цикъла.

Машината на състоянията на TAP контролера се състои от 16 състояния. Налице са 6 стабилни състояния (индицирани с кръгови връзки в диаграмата на състоянията от фиг. 11.7) и 10 нестабилни състояния. Стабилни състояния са онези, в които TAP контролерът може да стои неограничен брой цикли при изпълняване на дадено условие.

В диаграмата на състоянията са налице два основни пътя – единият за достъп и управление на избрания даннов регистър, а другият – за достъп и управление на регистъра на инструкциите. Алгоритмично двата пътя са еднакви и могат да се преминат само алтернативно.



Фиг. 11.9. Архитектура на граничната сканираща логика.

11.3. Основни регистри на граничната сканираща логика

Граничната сканираща логика съдържа два типа регистри. Първият тип се състои от един-единствен регистър – регистъра на инструкцията (IR – Instruction Register). Втория тип се състои от няколко регистъра, наречени с общото име даннови регистри (DR – Data Register). В него влизат граничният сканиращ регистър (BSR – Boundary Scan Register), закъсяващият (байпас) регистър (BR – Bypass Register) и граничният контролен регистър (BCR – Boundary Control Register). Съставът на данновите регистри може да бъде различен при различните схеми.

С изключение на байпас регистъра всеки тестови регистър може да бъде разглеждан като двоен регистър, състоящ се от преместващ регистър и следящ паралелен регистър (регистър в сянка). Байпас регистърът е различен в това, че той се състои само от преместващ регистър. По време на съответното състояние на захващане (Захващане-IR за регистъра за инструкцията, Захващане-DR за даннов регистър) съответният преместващ регистър може да бъде паралелно за-

реден от източник, специфициран от текущата инструкция. По време на съответното състояние на преместване (Преместване-IR или Преместване-DR) съдържанието на преместващия регистър се извежда през TDO, докато ново състояние се въвежда от TDI. По време на съответното състояние на обновяване (Обновяване-IR или Обновяване-DR) следящият паралелен регистър се обновява от преместващия регистър.

Регистърът на инструкцията (IR) е дълъг осем бита и съобщава на схемата каква инструкция да бъде изпълнена. Информацията, съдържаща се в инструкцията, включва режим на работа (нормален режим на работа, при който схемата изпълнява нормалните си логически функции, или тестови режим, при който нормалните логически функции са забранени или променени), тестовата операция, която трябва да се изпълни, кой от данновите регистри да бъде избран за включване в сканиращия път по време на сканиране на даннов регистър и източника на данните за захващане в избрания даннов регистър по време на състоянието Захващане-DR.

По време на състоянието Захващане-IR IR се зарежда с определена двоична стойност. Докато се извършва преместване на инструкцията вътре в регистъра, тази стойност се извежда през TDO и може да бъде възприета като проверка, че IR се намира на сканиращия път. По време на състоянието Обновяване-IR стойността, която е била въведена в IR, се зарежда в неговия следящ паралелен регистър. В този момент става обновяване на текущата инструкция и едва тогава се възприемат и въздействуват евентуални промени в режима. При включване на захранването или при състоянието на начално установяване IR се установява в стойност, която избира байпас регистъра.

Граничният сканиращ регистър (BSR) е с различна дължина в зависимост от изводите на схемата (за ‘8244 той е 18 бита дълъг). Той се състои от по една гранична сканираща клетка (BSC) за всеки нормално функциониращ входен извод и по една BSC за всеки нормално функциониращ изходен извод. BSC се използува за следните функции, които могат да се прилагат и едновременно: 1) Да съхранява тестовите данни, които трябва да бъдат приложени вътрешно към входовете на нормалната логика в ядрото и/или външно към изходните изводи на схемата; 2) Да захваща данните, които се появяват вътрешно в изходите на нормалната логика на ядрото и/или външно на входните изводи на схемата.

Източникът на данните, откъдето се зарежда BSC по време на състоянието Захващане-DR, се определя от текущата инструкция. Съдържанието на BSC може да бъде сменено по време на състоянието Старт тест, както е определено от текущата инструкция. Съдържанието на BSC не се променя по време на състоянието Начално установяване.

Граничният контролен регистър (BCR) е дълъг два бита. BCR се използува като допълнителен указател за някои инструкции, за да се изпълнят допълнителни тестови операции, които не са включени в основния стандарт. По време

на състоянието Захващане-DR съдържанието на BCR не се променя.

Закъсяващият (байпас) регистър (BR) е еднобитов и с неговото избиране се съкращава дължината на системния сканиращ път, като по този начин се редуцира броят на битовете от тестовата поредица, която трябва да се приложи за да се изпълни тестовата операция. По време на Захващане-DR байпас регистърът се зарежда с 0.

11.4. Основни функции на граничната сканираща логика

Функциите на граничната сканираща логика се задават от кода на операцията, зареден в регистъра на инструкцията. За окталния буфер ‘8244 те са 16 на брой, 8 от които се изпълняват в нормален режим на работа, а останалите 8 – в тестови режим. По-важните функции могат да бъдат накратко представени в следващото изложение:

Гранично сканиране – граничният сканиращ регистър е избран на сканиращия път. Данните, появяващи се на входните изводи на схемата, се захващат във входните гранични сканиращи клетки и в същото време данните, които се появяват на изходите на нормалната логика, се захващат в изходните гранични сканиращи клетки. Данните, които са били сканирани във входните гранични сканиращи клетки, се появяват на входовете на нормалната логика, докато данните сканирани, в изходните гранични сканиращи клетки, се появяват на изходните изводи на схемата. Схемата работи в тестови режим.

Байпас сканиране – байпас регистърът е избран на сканиращия път. По време на състоянието Захващане-DR в байпас регистъра се зарежда 0. Схемата работи в нормален режим.

Обикновено гранично сканиране – граничният сканиращ регистър е избран на сканиращия път. Данните, появяващи се на входните изводи на схемата, се захващат във входните гранични сканиращи клетки, докато данните, появяващи се на изходите на нормалната логика, се захващат в изходните гранични сканиращи клетки. Схемата работи в нормален режим.

Контрол на високоимпедансно състояние – байпас регистърът е избран на сканиращия път. По време на състоянието Захващане-DR в байпас регистъра се зарежда 0. Схемата работи в модифициран тестови режим, в който всички изходни изводи на схемата са поставени във високоимпедансно състояние. Входните изводи на схемата остават опериращи и схемата изпълнява нормалните си логически функции.

Контрол на граничната сканираща логика за логически нива – байпас регистърът е избран на сканиращия път. По време на състоянието Захващане-DR в байпас регистъра се зарежда 0. Данните, намиращи се във входните гранични клетки, се появяват на входовете на нормалната логика, докато данните в изходните гранични сканиращи клетки се появяват на изходните изводи на схемата. Схемата работи в тестови режим.

Четене на граничната логика – граничният сканиращ регистър е избран на сканиращия път. Стойността в BSC остава непроменена по време на състоянието Захващане-DR. Тази инструкция е полезна за проверка на данните след изпълнение на тестова операция за паралелен сигнатурен анализ.

Самотестване на граничната логика – граничният сканиращ регистър е избран на сканиращия път. Всички гранични сканиращи клетки инвертират своето текущо съдържание по време на състоянието Захващане-DR. По този начин съдържанието на следящия граничен регистър може да бъде прочетено навън, за да се провери целостта на преместващия регистър и следящия паралелен регистър на BSC. Схемата работи в нормален режим.

Превключване на граничните изходи – байпас регистърът е избран на сканиращия път. По време на състоянието Захващане-DR в байпас регистъра се зарежда 0. Клетките на преместващия регистър на изходните гранични сканиращи клетки инвертират своето съдържание при всеки нарастващ фронт на TCK по време на състоянието Старт-тест. В същото състояние те се обновяват в следящия паралелен регистър и се извеждат на съответния изходен извод при всеки спадащ фронт на TCK. Данните във входните гранични сканиращи клетки остават непроменени и се прилагат към входовете на нормалната логика. Данните, появяващи се на входните изводи на схемата, не се захващат във входните гранични сканиращи клетки. Схемата работи в тестови режим.

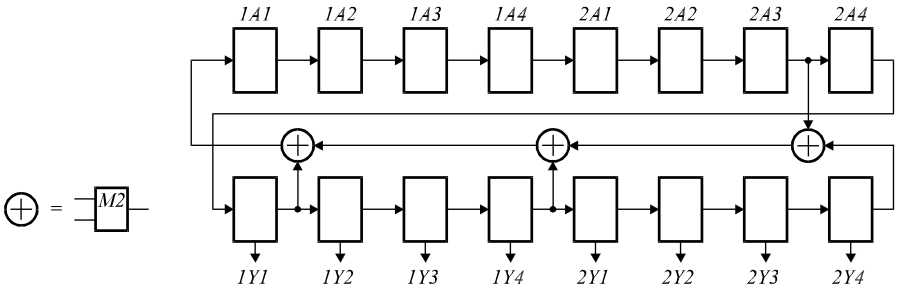
Сканиране на граничния контролен регистър – граничният контролен регистър е избран на сканиращия път. Стойността на BCR остава непроменена по време на състоянието Захващане-DR. Тази операция трябва да се изпълни преди операцията Старт на тест, за да специфицира следващите тестови операции.

Старт на тест – байпас регистърът е избран на сканиращия път. По време на състоянието Захващане-DR в байпас регистъра се зарежда 0. Схемата работи в тестови режим. По време на състоянието Старт-Тест се изпълнява една от четирите тестови операции за ‘8244, специфицирани от съдържанието на граничния контролен регистър:

– ***стробиране на входовете и превключване на изходите (TOPSIP)***, при което постъпващите данни на входните изводи на схемата се захващат в преместващия регистър на входните гранични сканиращи клетки при всеки нарастващ фронт на TCK. Тези данни се обновяват в следящия паралелен регистър на входните гранични сканиращи клетки и се прилагат към входовете на нормалната логика. Данните в преместващия регистър от елементите на изходните гранични сканиращи клетки се инвертират при всеки нарастващ фронт на TCK, обновяват се в следящия граничен регистър и се прилагат към изходните изводи на схемата при всеки падащ фронт на TCK;

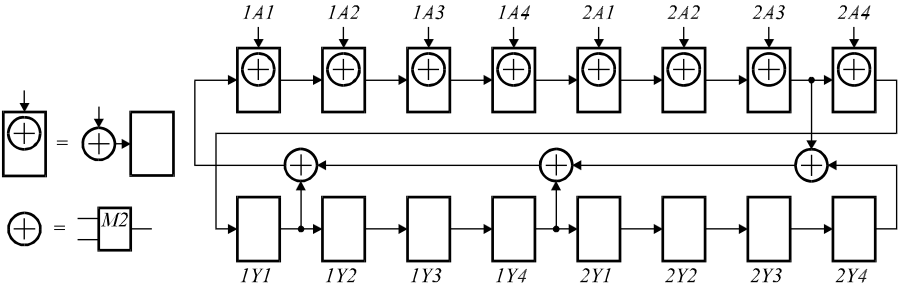
– ***псевдослучаен генератор на тестови поредици (PRPG)***, който се формира от осемте входни и осемте изходни гранични сканиращи клетки според фиг. 11.10. Псевдослучайната последователност се генерира при всеки нараст-

ващ фронт на TCK, обновява се в следящия паралелен регистър и се прилага към изходните изводи на схемата при всеки спадащ фронт на TCK. Данните от псевдослучайната последователност също се обновяват в следящия паралелен регистър на входните гранични сканиращи клетки и се прилагат към входовете на нормалната логика в качеството на стумулиращи въздействия. Преди да се изпълни тази операция, е необходимо BSR да се зареди със стойност различна от нула;



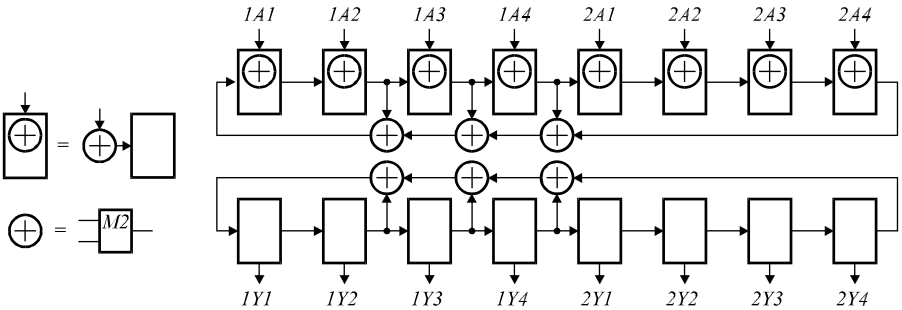
Фиг. 11.10. Конфигурация на генератор на псевдослучайни последователности.

– ***паралелен сигнатурен анализ (PSA)***, при който постъпилите данни на входните изводи на схемата се компресират в 16-битова паралелна сигнатура в преместващия регистър на елементите от преместващия регистър на граничните сканиращи клетки при всеки нарастващ фронт на TCK. Тези данни се обновяват в следящия регистър на входните периферни сканиращи клетки и се прилагат към входовете на нормалната логика. Данните в следящите регистри на изходните гранични сканиращи клетки остават непроменени и се прилагат към изходите на схемата. На фиг. 11.11 е показано формирането на сигнатурен регистър от граничните сканиращи клетки. Преди да се изпълни тази операция, е необходимо BSR да бъде зареден с инициализираща стойност;



Фиг. 11.11. Конфигурация на формировател на сигнатура.

– ***едновременен сигнатурен анализ и генератор на псевдослучайни поредици (PSA/PRava*** и opGP), приостъпващите данни на входните изводи на схемата се компресират в 8-битова паралелна сигнатура в преместващия регистър от входните гранични сканиращи клетки при всеки нарастващ фронт на TCK. Тези данни се обновяват в следящия паралелен регистър на входните гранични сканиращи клетки и се прилагат към входовете на нормалната логика. В същото време се формира 8-битов псевдослучаен генератор от преместващ регистър, образуван от изходните гранични сканиращи клетки при всеки нарастващ фронт на TCK, обновява се в следящия регистър на изходните гранични клетки и се прилага към изходните изводи на схемата. На фиг. 11.12 е показано формирането на 8-битов генератор на псевдослучайни последователности и сигнатурен формирователь. Преди стартирането на операцията е необходимо да се зареди в сигнатурния формирователь подходяща начална стойност, различна от нула.



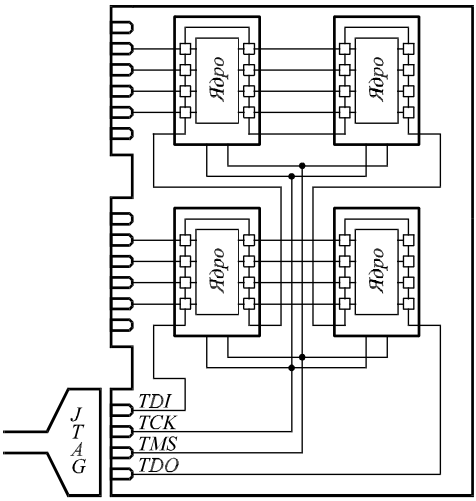
Фиг. 11.12. Конфигурация на едновременен генератор на псевдослучайни последователности и сигнатурен формирователь.

11.5. Тестване на цифрови устройства с гранична сканираща логика

JTAG стандартът е проектиран основно за тестване на ***ниво печатна платка***, т.е. елементите на електронното устройство са съсредоточени в рамките на една платка, затова и най-големи предимства се реализират на това ниво. На всички етапи на провеждането на диагностиката – проектанска, производствена и сервизна, се увеличава "наблюдаемостта" и "контролируемостта" върху работата на устройството, без да се засягат функциите на предавателните и приемните схеми. Реализира се основното предимство – не е необходим физически достъп до елементите в платката при провеждането на диагностиката. Всички тестове се провеждат чрез стандартизирания JTAG куплунг с неговите основни четири сигнала (фиг. 11.13). Поставянето на допълнителен куплунг на печатната платка съвсем незначително я оскъпява, но предимствата, които се достигат, напълно оправдават това оскъпяване.

При проектирането на гранична сканираща верига на ниво печатна платка се

препоръчва да се използува проста, последователна верига (регламентираната от стандарта организация тип "кръг"). Препоръчва се да се буферират сигналите TCK и TMS и да се извърши проверка на опроводяването, терминирането и времевите характеристики на тези сигнали. В прототипа трябва да бъдат проверени всички TAP сигнали за връзка и посока. Неизправност в някой от тези сигнали може да доведе до неправилна работа на TAP контролера. Най-опасно е неправомерното извеждането на схемите с гранична логика от нормален режим на работа и въвеждането им в някой от тестовите, където изходите на схемата са управлявани от граничната сканираща логика и могат да изпаднат в конфликт с тези или други схеми от устройството.

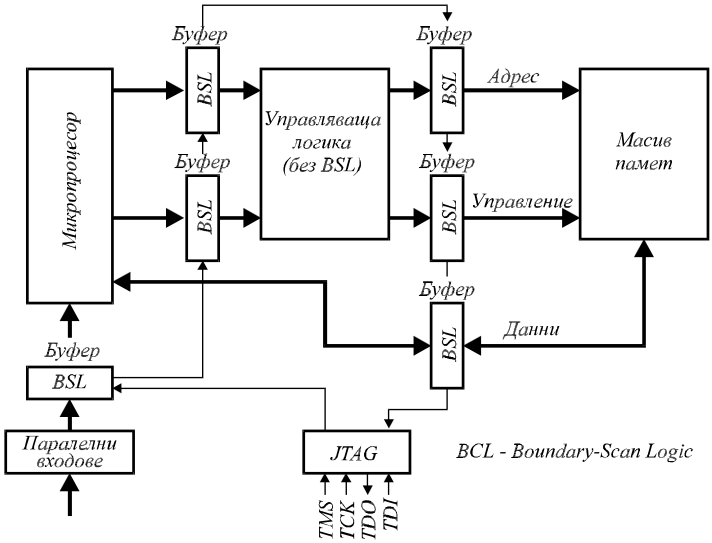


Фиг. 11.13. Диагностика с гранично сканиране на ниво печатна платка.

Предимствата на периферната сканираща логика могат да се оползотворят най-пълно, ако още в процеса на проектирането се поставят схеми с гранично сканиране навсякъде, където е възможно. Това се улеснява от факта, че много обикновени схеми притежават произвеждани функционални аналози с вградено гранично сканиране.

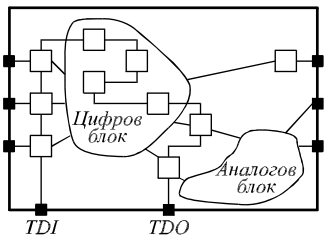
В много случаи е подходящо да се поставят схеми с гранично сканиране дори на места, където не е необходимо тяхното наличие за нормално функциониране на устройството. Повишаването вследствие на това на цената на устройството се компенсира от повишената достоверност и тестопригодност на изделието. Например при въвеждането на паралелна информация в микропроцесорна система е удобно да се въведе допълнително буфериране чрез схема с гранично сканиране. Диагностичните възможности на устройството се увеличават, ако съществуващ блок логика без гранично сканиране се "огради" от схеми с гра-

нично сканиране (фиг. 11.14).



Фиг. 11.14. Примерно вграждане на гранична сканираща логика при микропроцесорна система.

Особено силно проличават предимствата на граничната сканираща логика при прилагането ѝ на ***ниво интегрална схема*** (ниво електронен чип за големи и свръхголеми интегрални схеми), където практически липсва възможност за вътрешен физически достъп.

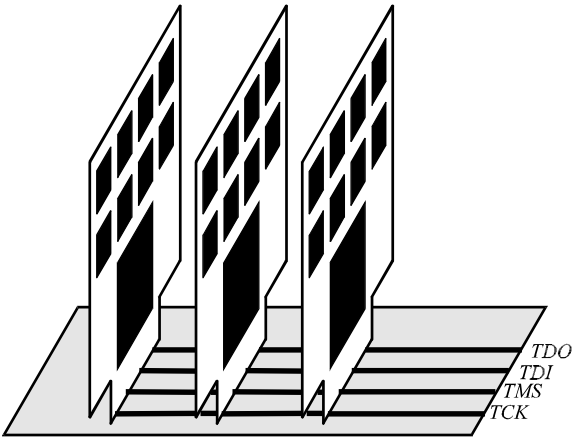


Фиг. 11.15. Диагностика с гранично сканиране на неговия ниво ин схема.

Организирането на гранично тестване на ниво интегрална схема изисква замяна на нормалните запомнящи елементи (тригери, регистри и др.) с техните гранично сканируеми двойници, които да тогава могат дят сканираща верига (фиг. 11.15). При пълното гранично сканиране всички запомнящи елементи са заменени със сканиращи. Така схемата всъщност се явява разделена на блокове от комбинационна логика, свързани помежду си с паралелните входове и изходи на сканиращи преместващи регистри. При частичното гранично сканиране на замяна се подлагат само избрани запомнящи

елементи. То се прилага, когато поради функционални особености на схемата диагностиката не може да се извърши поотделно, т.е при провеждането й не могат да се изключат някои блокове.

Гранично сканиращата диагностика с успех се прилага на ниво система, като тук обаче съществуват някои особености, които трябва да се имат предвид при организирането и използването. Това се отнася за случаите, когато системата е разделена на отделни модули (платки), свързани помежду си посредством основен модул или основна платка – вж. фиг. 2.16.



Фиг. 11.16. Гранично сканиране на системно ниво.

Сигналите TDI и TDO на отделните модули са последователни изводи. Те могат да бъдат свързани по организацията тип "кръг" в проста последователна верига. Това на пръв поглед просто решение има някои недостатъци. Освен че сканиращият път може да се окаже твърде дълъг, то отстраняването на някой от модулите ще го прекъсне и ще доведе до невъзможност за прилагането на диагностика. С други думи, диагностиката трябва да се провежда при присъствието на всички модули едновременно.

Алтеративно решение, предлагано от стандарта IEEE 1149.1, е използването на организация тип "звезда". При нея всички модули непосредствено получават сигнали TCK и TDI и изпращат сигнал TDO, но всеки модул получава отделен TMS сигнал. В тази конфигурация само един от модулите е разрешен сканиращ достъп в даден момент. Когато модулът е разрешен, асоцииранят към него TMS сигнал е активен, докато всички други TMS сигнали са неактивни. Проблемът при тази организация е, че с увеличаването на модулите в устройството расте и броят на поддържаните TMS сигнали.

Тук е необходимо да се каже, че съществуват и се предлагат други начини

за организация на сканиращата верига на системно ниво, като например използването на специален адресируем следящ порт (ASP), но те вървят с допълнение към стандартния протокол на стандарта IEEE 1149.1.

Както стана ясно, процесът на диагностика с гранично сканиране се извършва чрез JTAG порта от тестер. В някои случаи това може да бъде специализиран тестер за определено изделие, но най-често е скъп адаптер за компютър с подходящо програмно осигуряване или работна станция. Тези тестери като минимум изпитват TAP на ниво платка под контрол на векторен файл.

Ключ към успеха при граничното сканиране е използването на компютърно автоматизирана техника (computer-automated-engineering – CAE) и специализираните развойни средства за разработка и провеждане на диагностика с гранична сканираща логика, където процесът на тестване се извършва от високо автоматизиран генератор за тестови последователности (ATPG). Това са скъпо струващи програмни продукти, които автоматично генерират тестови последователности за прототипа или за изделието, които се прилагат към порта за тестови достъп. Най-добрите от тях генерират тестове за TAP и описание на езика за гранично сканиране BSDL и за целостта на сканиращия път. Те генерират и следващите типове от тестове за структурен тест на устройство: граничен вътрешносхемен тест, виртуални връзки и взаимодействие и виртуален тест на схеми и блокове.

При проектирането в среда на компютърно автоматизирана техника проектантът разполага виртуално с всички желани нива на наблюдаемост и управляемост върху вътрешните връзки в платката или чиповете.

Най-добрите диагностични продукти, основаващи се на граничното сканиране, използват управление на база данни за сканиращия път, което позволява интерактивен поглед и управление само на тези елементи и блокове от печатната платка, които представляват интерес. Такива продукти напълно скриват сложността на TAP протокола и гранично-сканиращата верига от потребителя и осигуряват ефективна проектантска диагностика, като: паралелни стимулгенератори, анализатор на логически състояния, анализатор на времеви съотношения и др.

Такива продукти поддържат различни методи за генериране на тестови вектори: интерактивен, CAE паралелен с автоматизирано преобразуване в сериен и гранично-сканиращия ATPG (базиран на стандартен формат за обмен на информация като Serial Vector Format – SVF). Те също описват гранично-сканиращата йерархия, използвайки индустриални стандартни формати като BSDL, Hierarchical Scan Description Language HSDL и езика за описание на файловете на връзките EDIF (Electronic Design Interchange Format). Те са чувствителни към особеностите на изделието, така че неправилна работа с граничната логика да не доведе до физическата му повреда.

12. ЛОГИЧЕСКИ ПРОБНИЦИ

Логическите пробници са най-простите контролно-диагностични уреди. Те имат незаменимо приложение при оперативна диагностика на електронни цифрови устройства, за търсене на елементарни повреди. Дори при използуването на мощна диагностична апаратура логическите пробници са удобно допълнение за по-точна локализация на неизправността. Основно място в широката гама логически пробници заемат логическата сонда, генераторната сонда и токовата сонда.

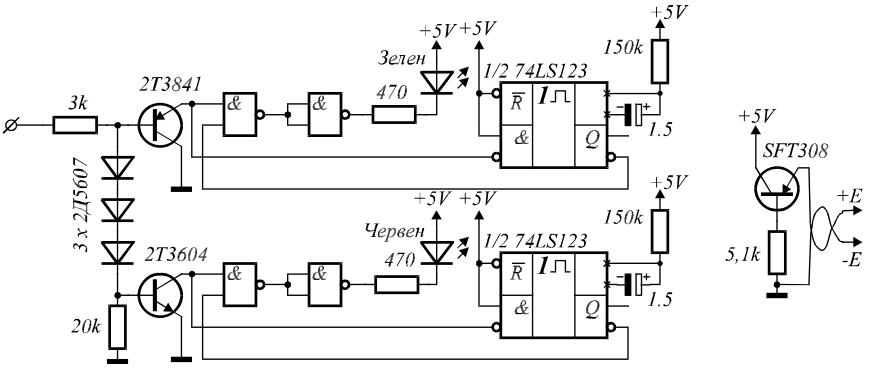
12.1. Логическа сонда

Логическата сонда, наричана още тестер за логическо състояние, представлява малогабаритен и евтин уред, позволяващ да се определи логическото състояние на някоя точка в проверяваното устройство. Логическите сонди могат да определят принадлежността на постояннотоковото ниво на сигнала към зоната на нулевото, единичното или неопределеното състояние. Допълнителни възможности на логическите сонди са индициране на единични импулси, определяне на доминиращото състояние на непрекъснато променящ се сигнал и др.

Примерна схема на логическа сонда е показана на фиг. 12.1. Тя е предназначена да възприема TTL нива. Индицирането на логическа *0* се извършва чрез зелен светодиод, а на логическа *1* – чрез червен. Входният сигнал постъпва едновременно в две вериги – едната за възприемане на логическата *0*, а другата – за възприемането на логическата *1*. Последователно във веригата за възприемане на логическата *0* е включен силициев транзистор от PNP тип по схема на емитерен повторител. Напрежението му база-емитер (0,6 *V*), извадено от прага на превключване на TTL елемента (1,2 *V*), формира допустимо ниво на входния сигнал не повече от 0,6 *V*, за да се индицира той като логическа *0*. Входящият сигнал ще се индицира като логическа *1*, ако нивото му е по-високо от 2,4 *V*. Това е постигнато, като е формиран праг на сработване по веригата на логическата *1*, съставен от 4 силициеви PN прехода (три диода и прехода база-емитер на NPN транзистора).

Постъпващите входни импулси допълнително се разширяват от два чакащи мултивибратора 74LS123 до около 100 *ms* – стойност достатъчна, за да бъде забелязано от човешко око "премигването" на светодиодите. Единият от чакащите мултивибратори разширява импулсите с логическо ниво *1*, а другият – с логическо ниво *0*. Това дава възможност да бъдат наблюдавани тесни импулси (по-широки от 40 *ns*). Логическата сонда се захранва с напрежение +5 *V*. Защита от обратно включване се осъществява с германиев PNP транзистор, който при правилно включване се насища и пропуска захранването към сондата. При неправилно включване германиевият транзистор се запушва и предпазва сондата от

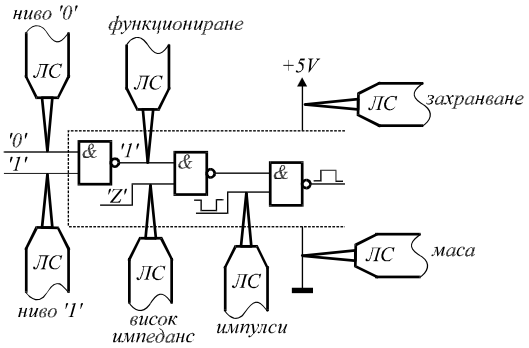
обратно захранващо напрежение. Използуван е германиев транзистор поради по-високото му обратно пробивно напрежение в сравнение със силициев.



Фиг. 12.1. Принципна схема на логическа сонда.

Логическите сонди се изработват в малогабаритен корпус, позволяващ да бъде държан в ръка. Индикаторите и командните органи са разположени непосредствено на корпуса. Той е снабден с контактен накрайник за отнемане на сигнала и гъвкав шнур – за захранване. Нормално се използува стабилизираното захранване на диагностицирания уред.

Основното предназначение на логическата сонда е проверката на подаването на захранване на цифровите схеми, контролиране на статичните нива в логическите елементи, функционирането им, наличието на импулси и др. – фиг. 12.2.



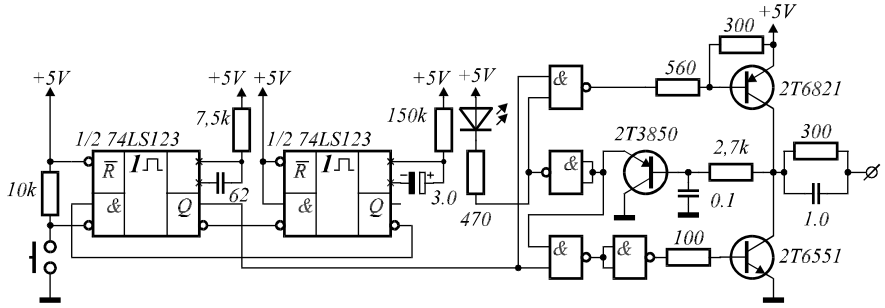
Фиг. 12.2. Основно предназначение на логическата сонда.

Разновидност на логическата сонда представлява логическият клипс. Той се изработва във вид на контактна щипка, която се закрепя на корпуса на интег-

ралната схема с двуредни щифтови изводи. В горния край на логическия клипс са разположени светодиоди, отразяващи логическото състояние на съответните изводи на интегрални схеми. Логическият клипс получава захранване от самата интегрална схема, върху която е захванат. Той може да обслужва определена фамилия логически схеми с някои ограничения, тъй като съществува голямо разнообразие в корпусите и разположението на захранването по тях. Обикновено логическите клипсове не притежават разширители за визуализиране на кратки импулси.

12.2. Генераторна сонда

Генераторната сонда е стимулиращ уред, предназначен за формиране и изпращане на импулси с определена амплитуда и продължителност във веригите на диагностицираното електронно-цифрово устройство. Тя може да изработва единични импулси, импулсни поредици и групи импулси. Обикновено генераторната сонда формира импулси с продължителност около 250 *ns* и изходен ток до 1 *A*. Освен че такива импулси могат да захранват едновременно голямо количество входове, те могат да атакуват и вериги, към които има свързани и изходи на логически елементи. Благодарение на малката продължителност на импулсите логическите елементи не се повреждат.

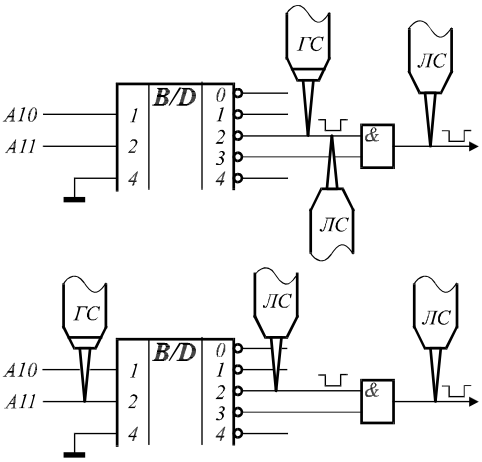


Фиг. 12.3. Принципна схема на генераторна сонда.

На фиг. 12.3 е показана примерна принципна схема на генераторна сонда, имаща възможност за изработване на единични импулси и импулсни поредици. При кратко натискане на бутона се изработва единичен импулс с продължителност около 200 *ns* от чакащ мултивибратор 1/2 74LS123. При задържането на бутона в натиснато положение се дава разрешение на двата чакащи мултивибратора в 74LS123 да работят като генератор на импулсна поредица с период около 200 *ms*.

Изходното стъпало на генераторната сонда притежава високоимпедансно състояние, затова при нормални условия включването на сондата в точка от тес-

тваната система не влияе на нейната работа. Тя анализира нивото на сигнала в допирната точка и формира автоматично импулси с противоположна полярност. Генераторната сонда притежава същата защита от неправилно подаване на захранващото напрежение, както и логическата сонда (на фиг. 12.3 тя не е показана).



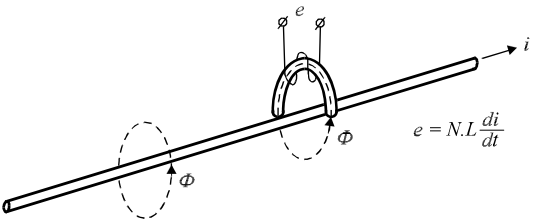
Фиг. 12.4. Съвместно използуване на генераторна и на логическа сонда

Конструктивно генераторната сонда се изработва подобно на логическата сонда. Двете сонди се използуват в повечето случаи заедно. Методът, при който за контролиране работата на цифрова схема се използуват съвместно генераторна сонда и логическа сонда, се нарича тестване "стимул-реакция". Чрез генераторната сонда в схемата се подават стимулиращи импулси (без да се изключват логическите елементи), а чрез логическата сонда се следи за тяхното разпространение и получената реакция. Познавайки проверяваната схема, операторът може да проследи веригите на разпространяване на сигналите и да определи изправността или неизправността на схемата. Съвместното използуване на двете сонди е пояснено от фиг. 12.4.

12.3. Токова сонда

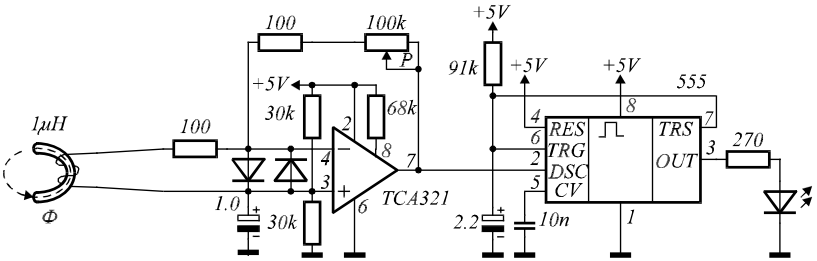
Токовата сонда е сравнително по-малко използуван логически пробник. Тя е предназначена за контрол на тока в единични проводници (в това число и проводници на печатни платки), като реагира само на импулсни токове. Протичащият в проводника ток образува около него магнитно поле, като измененията на тока довеждат до промяна в магнитния поток Φ. Тази промяна на потока се използува в токовата сонда за индуциране на електродвижещо напрежение в бобина, монтирана на върха на сондата.

Бобината се състои от подковообразен феритен магнитопровод с намотки около него. Тя се ориентира така, че създаденият около проводника магнитен поток да преминава през магнитната й верига – фиг. 12.5. При промяна на тока, потокът също се променя и в намотката на бобината се индуцира електродвижещо напрежение *e*. То е правопропорционално на броя на намотките *N* в бобината, на нейната индуктивност *L* и на скоростта на промяна на тока в изследвания проводник *di/dt*. Сондата не реагира на протичането на постоянен ток по изследвания проводник. Обикновено чувствителността на една токова сонда може да се регулира в границите от 1 *mA*/μ*s* до 1 *A*/μ*s*. Оформя се по подобен на другите пробници начин.



Фиг. 12.5. Принцип на действие на токовата сонда.

Примерна схема на токова сонда е показана на фиг. 12.6. Тя съдържа в себе си усилвател на електродвижещото напрежение, получено от токочувствителната бобина. За усилвател е използуван висококоростният (SR = 50 *V*/μ*s*) операционен усилвател TCA321. Той се характеризира още и с малкото си захранващо напрежение (до ±2 *V*), което дава възможност токовата сонда да се захранва от напрежение +5 *V*. Усилването на усилвателя се регулира от 1 до 1000 с потенциометъра P, което дава възможност за промяна на чувствителността на сондата от 1 *A*/μ*s* до 1 *mA*/μ*s*. Токочувствителната бобина е изработена така, че произведението й *N.L* да има стойност 1 μ*H*.

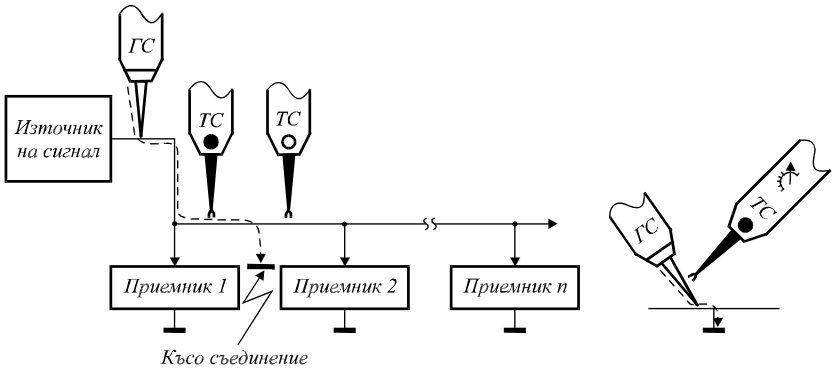


Фиг. 12.6. Примерна принципна схема на токова сонда.

След усилвателя е включен интегрален таймер 555, работещ в моновибрато-

рен режим, чиято задача е да разшири получените напрежителни импулси до стойност около 250 *ms*, за да могат те да бъдат регистрирани от човешко око. При захранване от +5 *V* неговият праг на задействуване е вътрешно формиран на 1,67 *V*. При същите условия нулевата линия в изхода на TCA321 е на 2,5 *V*, така че задействуване на моновибратора ще се получи, когато напрежителният импулс в изхода на усилвателя превиши 0,83 *V*. Токовата сонда трябва да има защита от неправилно включване на захранващото й напрежение, както при другите сонди.

За успешното използуване на токовата сонда е необходим значително поголям опит, тъй като операторът трябва да може да регулира чувствителността й и правилно да я ориентира върху изследвания проводник. Токовата сонда обикновено се използува съвместно с генераторната сонда и е много удобна при търсене на физическото място на повреда в магистрално организирани линии. Откриването на такава неизправност е илюстрирано от фиг. 12.7.



Фиг. 12.7. Приложение и използуване на токова сонда.

При магистралната организация на дадена микропроцесорна система към един изход (например адресна линия на микропроцесора) са свързани паралелно входове на много елементи. При нормална работа сигналът, предаван по нея, сменя състоянието си. Повреда в един от елементите се разпространява по цялата линия. Например при късо съединение към маса във входа на някой елемент цялата магистрална линия ще стои в логическа *0*. Тогава може да се предполага, че късо съединение има в която и да е от паралелните вериги. За откриването му в началото на неизправната линия се подават поредица импулси от генераторната сонда. Поради късото съединение в линията напрежението се задържа на ниво логическа *0*. От генераторната сонда към мястото на късото съединение тече ток. Проверката се състои в придвижване на токовата сонда по линията до изчезване на показанието за електрически ток. Така се определя фи-

зическото място на късото съединение. Преди започването на работа чувствителността на токовата сонда трябва да се настрои според токовите импулси на генераторната сонда.

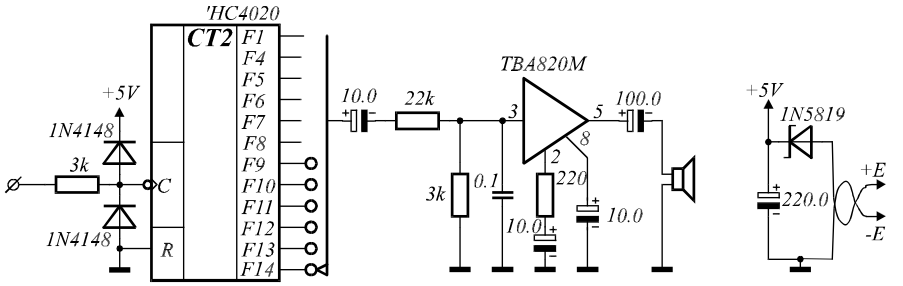
Логическите пробници са удобни малогабаритни уреди за проверка както при прости, така и при сложни електронни цифрови устройства и системи и затова трябва да бъдат неотменима принадлежност за всеки инженер по електроника.

12.4. Звукова сонда

Звуковата сонда позволява на оператора да прослушва сигналите, действащи в микропроцесорната система. С други думи, тя представлява алтернатива на обикновената логическа сонда, която осигурява само визуална индикация на логическите състояния и следователно не позволява да се правят обосновани предположения за поведението на импулсния сигнал в проверяваната линия.

Прослушвайки сигналите в микропроцесорната система, може да се разбере какво става. С помощта на звуковата сонда се удава не само да се фиксира активността в конкретна линия, но и да се оцени честотата на импулсите и наличието на периодичност в сигналите. По звука могат да се различат сигналите на отделните линии, синхронизацията и разрешаването на схемите.

Принципът на действие на звуковата сонда е достатъчно прост. Високочестотните сигнали, действащи в микропроцесорната система, се преобразуват в сигнали с по-ниска звукова честота с помощта на двоичен честотен делител. Изходните сигнали на делителя се формират и подават на обикновен усилвател на звукова честота.



Фиг. 12.8. Примерна принципна схема на звукова сонда.

Принципната схема на звуковата сонда е показана на фиг. 12.8. Броячът HC4020 осигурява делене на честотата на входните сигнали. Коефициентът на делене се задава с превключвател и може да варира от 2^9 до 2^{14}. Входната верига, състояща се от резистор и диоди, защитава сондата от пренапрежения. Ин-

тегралната схема TBA820M е усилвател на звукова честота с фиксиран коефициент на усилване.

Настройването на звуковата сонда се състои в подбиране на подходящ коефициент на делене на брояча. Това се извършва, като сондата се допре до линия със системен такт от микропроцесорната система. При подходящо избиране на коефициент на делене трябва да се чуе "чист" тон с честота около 1 *kHz*.

При така настроената звукова сонда е целесъобразно да се прослушват сигналите от адресните линии, от линиите за данни и от управляващите линии. Тези сигнали са богати на хармоници и сигналите от тях са с характерен "дрезгав" звук. Проверката на микропроцесорна система със звукова сонда е най-добре да се извършва на принципа на сравняването на сигнали от идентични точки на две микропроцесорни системи, едната от които е изправна.

13. ЗАКЛЮЧЕНИЕ

Различните методи за диагностика на електронните устройства с микропроцесорно управление имат различна достоверност. Най-добри резултати се постигат, когато те се прилагат според случая и в подходяща последователност и комбинация едни с други. Кой от методите ще се използува, зависи от самото диагностицирано устройство и от възможностите на оператора. Конкретни препоръки не могат да се дадат, но трябва да се спазва определена последователност при провеждането на диагностика на електронни устройства с микропроцесорно управление.

Цел на диагностиката е да се открият и отстранят неизправности както в апаратната, така и в програмната част на микропроцесорното устройство. Преди всичко операторът трябва да се опита да получи предварителна информация за повредата чрез самодиагностиката на устройството. Ако самодиагностика не съществува или повредата е такава, че самодиагностиката не работи, е необходимо да се пристъпи към последователна проверка, както примерно е дадено по-долу. При положение че самодиагностиката даде някакъв резултат, някои от проверките могат да бъдат прескочени.

Преди всичко трябва да се проверят спомагателните блокове около микропроцесорната система. Такива са захранващият блок, блокът на датчиците, на изпълнителните механизми, на сигналните преобразуватели и др. Операторът трябва да знае, че вероятността да се повреди управляващата микропроцесорна система в едно комплексно съоръжение е най-малка. Затова е напълно погрешна практиката проверката да започва от микропроцесорната система. На първо място е необходимо да се провери захранването на устройството – започвайки от предпазителите и стигайки до конкретните изработени захранващи напрежения. Не е нужно операторът да се нахвърли направо върху микропроцесорната система, с логически анализатор например, и да установи, че повредата е в изгорял предпазител на захранващо напрежение. Едва след като се установи, че повредата не е в блоковете извън микропроцесорната система, се пристъпва към нейната проверка.

Проверката започва с потенциалите и сигналите на асинхронните входове на микропроцесора. Това се отнася за входовете на начално установяване, прекъсване, готовност и др. Ако сигналите и напреженията, подавани на тези входове, не са в норма, микропроцесорът няма да работи. Следващата проверка, която трябва да се извърши, е на входовете за тактовите импулси на микропроцесора. Както нивата, така и времевите съотношения на импулсите трябва да бъдат правилни. Някои микропроцесори, в това число и MC6800, имат строги изисквания към тези сигнали.

Изброените дотук проверки предхождат същинската част на диагностиката.

Макар че извършването им е елементарно, те позволяват да бъдат открити голяма част от появяващите се повреди. Обикновено те се провеждат с традиционните контролно-измервателни уреди – волтметри, осцилоскопи и др.

Следващите две проверки, които трябва да се направят, са проверките на микропроцесора и на микропроцесорните цикли за четене и запис. Тук е мястото на методите на статичните въздействия, на режима на свободно броене, на вътрешносхемния микропроцесорен емулатор и по-рядко на логическия и сигнатурния анализ. Тези проверки стесняват областта, в която се намира повредата. Записът и четенето са основните операции в една система с микропроцесор. Когато се работи с микропроцесорен емулатор, е удобно да се зациклят кратки програми за запис и последващо четене от определени адреси и синхронизирайки осцилоскоп по избора на тези адреси, да се наблюдават сигналите в системата.

За микропроцесорите, които имат отделно адресно поле за входно-изходните устройства, трябва да бъдат проверени входно-изходните операции за запис и четене.

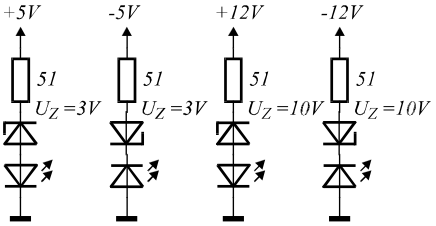
Ако системата изпълнява правилно операциите за запис и четене, може да се заключи, че магистралните линии, буферите и другите схеми между микропроцесора, паметта и периферните устройства са работоспособни. Следва проверка на специализираните периферни блокове на електронното устройство, чието диагностициране по правило се извършва най-трудно. Тук успешно могат да се използуват методите на вътрешносхемната емулация, на сигнатурния и логическия анализ.

За осигуряване на ефективна диагностика на електронните устройства е полезно да се вземат конструктивни мерки още в процеса на разработката. Те не само улесняват, но и облекчават диагностичните процедури. На първо място това са мерки, които улесняват включването на контролно-диагностичната апаратура. Някои от тях бяха изложени при разглеждането на сигнатурния анализ. Целесъобразно е най-важните сигнали и вериги на устройството да бъдат изведени на куплунги или в контролни точки. Желателно е да се предвиди допълнителен магистрален съединител за включване на външна апаратура.

Много полезен подход е поставянето на по-важните електронни елементи в цокли. Това важи най-вече за микропроцесора. Освен че при евентуални съмнения той може да бъде заменен, през неговия цокъл може да се осъществи достъп до жизнено важните магистрали на системата. Имайки предвид, че повечето диагностични устройства използуват тази възможност, препоръчително е винаги когато не съществуват ограничения, микропроцесорът да се поставя в цокъл. Желателно е също използуването на цокли за постоянната памет поради възможността за поставянето на диагностични програми и евентуалното използуване на ROM-емулатори. Поставянето на други елементи в цокъл зависи от виждането на конструктора. Цоклите трябва да бъдат с високо качество, за да се

осигури надеждна работа на системата! В противен случай монтирането им може да бъде причина за поява на трудно определими неизправности.

Оперативна информация за работата на едно електронно устройство може да се получи от вградени специализирани индикатори. Например поставянето на индикатори за захранващите напрежения веднага би указало тяхното излизане от норма (приблизително) или отпадането им. На фиг. 13.1 са показани четири индикаторни вериги за най-често използуваните захранващи напрежения при електронните устройства. При падането на някое от напреженията повече от 1 *V* съответствуващият светодиод угасва.



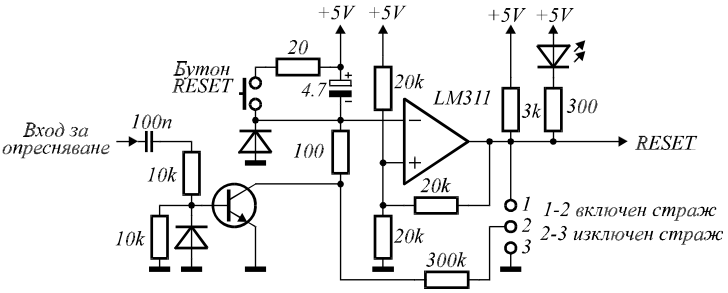
Фиг. 13.1. Индикаторни вериги за захранващи напрежения.

Практикува се поставянето на светодиодни индикатори и за основни и значими за работата на системата сигнали. Това са някои от асинхронните входове на микропроцесора, като: входове за прекъсване, готовност, сигналите по серийните канали за вход и изход и др. За индицирането на сигнали, където се очаква да има краткотрайни импулси, е необходимо да се поставят разширители на импулсите, подобни на тези в логическата сонда. Конструктивно индикаторите се разполагат върху печатните платки, където са достъпни само при сервизно диагностициране, или върху операторския пулт на устройствата.

Една полезна диагностична схема представлява т.нар. "часовников страж" (watchdog). Тя следи работата на микропроцесорната система и когато микропроцесорът излезе от управление, му подава сигнал за начално установяване и индицира това действие. Схемата съдържа таймер, който се запуска по програмен път. Във всички разклонения на работната програма се поставят команди, запускащи този таймер. Най-дългият интервал от време, през което се повтаря запускането, трябва да е по-къс от продължителността на генерирания времеви интервал и при правилна работа на системата таймерът непрекъснато ще се презапуска, преди да е изтекъл времевият интервал. Ако по някаква причина микропроцесорът излезе от управление и пропусне запускаща команда, таймерът завършва времевия интервал и часовниковият страж подава сигнал за начално установяване на системата.

Пример за такава следяща система е даден на фиг. 13.2. Тя съдържа генератор на нискочестотни правоъгълни сигнали (релаксатор) с период около 1 *s*, реализиран на базата на интегралния аналогов компаратор LM311. При включване

на захранващото напрежение той започва да работи в режим на свободни колебания, като първият импулс на изхода му е с ниско ниво (сигнал за начално установяване). След края на импулса микропроцесорът навлиза в работната си програма и трябва да започне да опреснява схемата на часовниковия страж. Опресняването се състои в периодично подаване на импулси през разделителен кондензатор (за да се гарантира, че постоянно ниво няма да блокира схемата).



Фиг. 13.2. Принципна схема на "часовников страж".

В конкретния пример опреснителните импулси напомпват през транзистор заряд в хрониращия кондензатор (хрониращата верига на релаксатора е формирана от капацитет 4,7 μF и резистор 300 $k\Omega$) и го поддържат постоянно зареден под единия праг на релаксатора. Генерациите са прекратени и изходът остава във високо ниво. Ако микропроцесорът не успее да навлезе в работната си програма, или вследствие на повреда или зашумяване излезе от нея, опресняването спира и свободните генерации на релаксатора се възобновяват. Започва периодично да се изпраща сигнал за начално установяване, до появата на ново опресняване. Към този сигнал е свързана и индикаторна светодиодна верига. Мигането на светодиода е указание за оператора, че микропроцесорът е излязъл от режим на правилна работа.

Някои микропроцесори имат вградена в себе си схема на часовников страж. Такъв е случаят с едночиповия микрокомпютър MC68HC11. Неговата вградена система на часовников страж действува по описания по-горе начин, но е показателна със сигурните си защити от погрешно сработване. Това дава повод за малко по-сериозното й описание.

В основата на часовников страж на MC68HC11 стои вътрешен таймер, който трябва да се презапуска (опреснява) от потребителската програма по-често, отколкото е времевият му интервал. Ако това не се извърши, т.е. не бъде опреснен таймерът в определеното време, часовников страж подава сигнал за начално установяване за самия едночипов микрокомпютър и за цялата система (изводът $\overline{RESET}$ на MC68HC11 може да действува като изход). Три са основните

защити, осигуряващи сигурна работа на часовниковия страж.

Първо, това е начинът на презапускане на таймера. То се извършва на две стъпки. Първата стъпка е запис на шестнадесетичното число `55` в запускащия таймера регистър COPRST. Втората стъпка е запис на шестнадесетичното число `AA` в същия регистър. Едва след втората стъпка се извършва фактическото запускане на таймера. Между двете стъпки могат да се изпълняват различни инструкции, но времето между стъпките трябва да е в рамките на интервала на таймера. Използването на двустъпково запускане на таймера и специфичните комбинации `55` и `AA` (и двете представляват редуващи се нули и единици) правят практически невъзможно случайното му презапускане.

Второ, това е начинът на определяне на времевия интервал на таймера за часовниковия страж. Могат да бъдат задавани четири фиксирани времеви интервала. Тези четири комбинации се определят от състоянието на двата бита *CR0* и *CR1* във вътрешния регистър OPTION на едночиповия микрокомпютър. Четирите състояния на тези два бита определят коефициенти 1, 4, 16 и 64 на броене в таймера на часовниковия страж за формиране на времевия му интервал. Таймерът се тактува от честота, представляваща системния такт *E*, разделен на 2^{15}. След начално установяване състоянието на битовете *CR0* и *CR1* е нула, което задава най-малкия интервал на таймера. Състоянието на тези битове може да бъде променено чрез запис само по време на първите 64 системни цикъла след начално установяване.

Трето, това е начинът на разрешаване и забраняване на работата на часовниковия страж. Това става чрез бита NOCOP в регистъра CONFIG. Ако битът е 1, часовниковият страж не е разрешен, а ако е 0 – е разрешен. Характерно за целия регистър CONFIG е, че той представлява памет от типа EEPROM, при което неговото изтриване и записване се извършва със специална програмна процедура. Това прави практически невъзможна случайната забрана на часовниковия страж по време на неговата работа.

Бидейки тактувана от микрокомпютърния такт *E*, системата на часовниковия страж ще престане да работи, ако тактът спре. В MC68HC11 обаче е предвидена друга система, която следи такта и може да подаде сигнал за начално установяване, ако честотата му падне под определена стойност. Препоръчва се двете системи да се ползуват заедно.

При програмното обслужване на часовниковия страж (презапускането на таймера) трябва да се спазва принципът запускащите инструкции да не се поставят в програми за обслужване на прекъсвания. Съществуват ситуации, при които в една микропроцесорна система върви обслужване на прекъсвания, а главната програма не функционира.

Допълнителна сигурност за правилното изпълнение на програмите представлява логика, която следи за валидни кодове на инструкции. По време на работа тази логика следи постъпващата информация по магистралата за данни.

Появата на невалиден код на инструкция предизвиква сработване на логиката и тя подава сигнал за прекъсване. Такава логика е вградена в някои микропроцесори като MC6803, MC68HC11 и др.

Ефективността на микропроцесорната диагностика силно зависи от контролно-диагностичната апаратура, с която тя се провежда. Но най-големият фактор за извършването й е операторът. Разнообразната и с големи възможности контролно-диагностична апаратура се обезсмисля без високо квалифициран персонал, който да я приведе в действие. И обратно, богатите познания на оператора, личният му опит и способности до голяма степен могат да заменят липсата на скъпоструваща апаратура и успешно да проведат диагностиката на микропроцесорната система.

ЛИТЕРАТУРА

1. Воробьев, Н., В. Горбунов и др. Микропроцессоры. В 3 кн. Кн. 3. Средства отладки. Лабораторный практикум и задачник. М., Высшая школа, 1986.

2. Домнин, С., Е. Иванов, Л. Муренко. Средства комплексной отладки микропроцессорных устройств. М., Энергоатомиздат, 1988.

3. Златаров, В., Р. Иванов, Г. Михов. Приложение на микропроцесорни системи в електронни устройства. С., Техника, 1984.

4. Иванов, Р., Г. Михов. Електронни цифрови устройства и системи. С., Техника, 1990.

5. Иванов, Р., Микропроцесорна схемотехника – учебник за дистанционно обучение. Технически университет – София, 1997.

6. Каракехайов, З., К. С. Кристенсен, О. Винтер. Проектиране на вградени микрокомпютърни системи с микроконтролери. София – Москва, PENSOFT, 2000.

7. Керезов, А. Микропроцесорна схемотехника – ръководство за лабораторни упражнения. Технически университет – София, 2000.

8. Кофрон, Д. Практически методи за проверка на микропроцесорни системи. С., Техника, 1984.

9. Михов Г., Контрол и диагностика на микропроцесорни системи. София, Технически университет – София, 1994.

10. Мясников, В., М. Игнатьев и др. Микропроцессоры. Системы программирования и отладки. М., Энергоатомиздат, 1985.

11. Рафикузаман, М. Микропроцессоры и машинное проектирование микропроцессорных систем. В 2 кн. Кн. 2. М., Мир, 1988.

12. Смикаров, А., Ц. Василев, И. Цанков, С. Смикарова. Специализирани микрокомпютърни системи. Русе, Авангард принт ООД, 1999.

13. Уильямс, Г. Отладка микропроцессорных систем. М., Энергоатомиздат, 1988.

14. Фергусон, Д., Л. Макари, П. Уилльямз. Обслуживание микропроцессорных систем. М., Мир, 1989.

15. Янев, К., А. Егоров и др. Наръчник. Проверка и настройка на цифрови устройства. С., Техника, 1990.

16. Tooley, M. Practical Digital Electronics Handbook. PC Publishing, London, 1998.

17. Сп. ELEKTOR, декември 1987 г. - FRG.

18. Boundary-Scan Logic, Data Book. Texas Instruments Inc. 1997.

19. MC68HC805C4 8-bit EEPROM Microcomputer Programming Module. AN966, Motorola Inc, 1985.

СЪДЪРЖАНИЕ

Графичен дизайн: доц. д-р инж. **Георги Славчев Михов**
Рецензент:
проф. д-р инж. **Рачо Маринов Иванов**

Стилов редактор и коректор:
Стояна Иванова Саева

НАСТРОЙКА И ДИАГНОСТИКА НА МИКРОПРОЦЕСОРНИ СИСТЕМИ
Българска
Първо издание

Подписана за печат м. април 2003 г.
Излязла от печат м. април 2003 г.
Печатни коли 20.00 Формат 60×84/16
Тираж 100 бр. Поръчка 27/30.04.2003 г.
Цена 5.70 лв. **ISBN 954-438-340-9**

МП Издателство на Технически университет – София

В учебника са изложени въпроси, свързани с теоретичните и практическите аспекти на настройката и диагностиката на микропроцесорни системи, а също и на използуваните методи и контролно-диагностична апаратура. Разгледани са тестери за статични въздействия, вътрешносхемни микропроцесорни и ROM емулатори, логически и сигнатурни анализатори, програматори, прости контролно-измервателни уреди и др. Внимание е обърнато на развойните средствата за диагностика на микропроцесорните системи в реално време и на съвместната настройка на апаратната и програмната им част. Представени са новите методи за контрол и диагностика на микропроцесорни системи с използуването на гранично сканираща логика. Дадени са конструктивни сведения за онези блокове от контролно-диагностичната апаратура, които определят нейните функции.

При представянето на материала са преследвани две основни цели: получаване на познания за провеждането на настройка и диагностика на микропроцесорни системи; получаване на познания за устройството на електронните диагностични уреди, с цел тяхното ползуване и разработване.

Учебникът е предназначен за студенти от висшите технически учебни заведения. Той може да бъде използуван и от специалисти, работещи в тази област.

5.70 лв.