Настройка и диагностика на микропроцесорни системи

Автор: гл. ас. д-р инж. Любомир Богданов

Европейски съюз

ПРОЕКТ BG051PO001--4.3.04-0042
„Организационна и технологична инфраструктура за учене през целия живот и развитие на компетенции”
Проектът се осъществява с финансовата подкрепа на
Оперативна програма „Развитие на човешките ресурси”,
съфинансирана от Европейския социален фонд на Европейския съюз
Инвестира във вашето бъдеще!

ЕСФ България
Европейски социален фонд

Съдържание

Видове постоянна памет

В зависимост от предназначението, паметите в една микропроцесорна система може да се разделят на два вида:
*памет за данни
*памет за инструкции

Видове постоянна памет

Памет за данни – енергозависима памет /съдържанието ѝ се губи при премахване на захранването/, в която се съхраняват временни резултати от изчисления в хода на изпълнението на програмата. Технологията, която се използва най-често за този вид памет е RAM.

Памет за инструкции (програмна памет) – енергонезависима памет /съдържанието ѝ се запазва дори при премахнато захранване/, в която се съхранява програмата на системата. Част от тази памет може да се зареди в RAM, за да се подобри производителността на системата, но името на този сегмент си остава “програмна памет”. Използваната технология е ROM, PROM, EPROM, EEPROM, Flash и Ferroelectric.

Видове постоянна памет

ROM (**R**ead **O**nly **M**emory) – енергонезависима памет, която се “програмира” по време на производството на чипа.
*Тя не може да бъде изтривана или записвана наново.
*Среща се и понятието Mask ROM.

Стойностите на битовете се реализират по различни начини:
*чрез метални контакти (връзка към Vdd означава 1, връзка към Vss означава 0);
*различни видове MOS транзистори (с индуциран или вграден канал)
*MOS транзистори с тънък или дебел оксиден слой на гейта (тънък – ниско прагово напрежение V_{GS}, дебел – високо прагово напрежение V_{GS}).

Видове постоянна памет

PROM (**P**rogrammable **R**ead **O**nly **M**emory) – енергонезависима памет, която се програмира от потребителя чрез електричество.

*Програмирането се състои в прегаряне на “бушони”, играещи ролята на битове. Алтернативното логическо състояние се представя със здрав, непрегорен бушон.

*Бушоните се реализират с полупроводникови елементи (диоди, транзистори, и т.н.). Прилагат се високи импулсни напрежения, които превръщат полупроводника в проводник.

*Веднъж записани, съдържанието им не може да се променя.

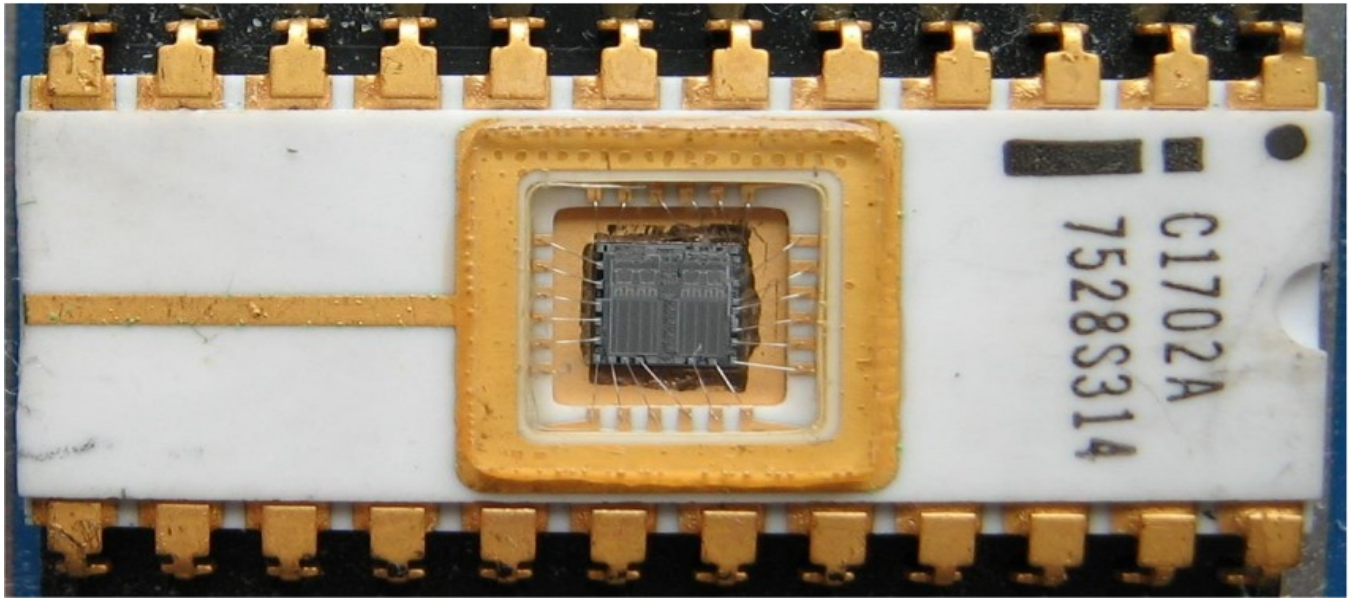
*Среща се и понятието **O**ne **T**ime **P**rogrammable, OTP.

Видове постоянна памет

EPROM (UV **E**rasable **P**rogrammable **R**ead **O**nly **M**emory) – енергонезависима памет, която се записва от потребителя чрез електричество и се трие чрез UV светлина.

*Корпусите на тези чипове имат кварцови прозорчета, позволяващи UV светлината да достигне кристала на чипа и да изтрие битовете.

*Триенето отнема 20 ÷ 40 минути.

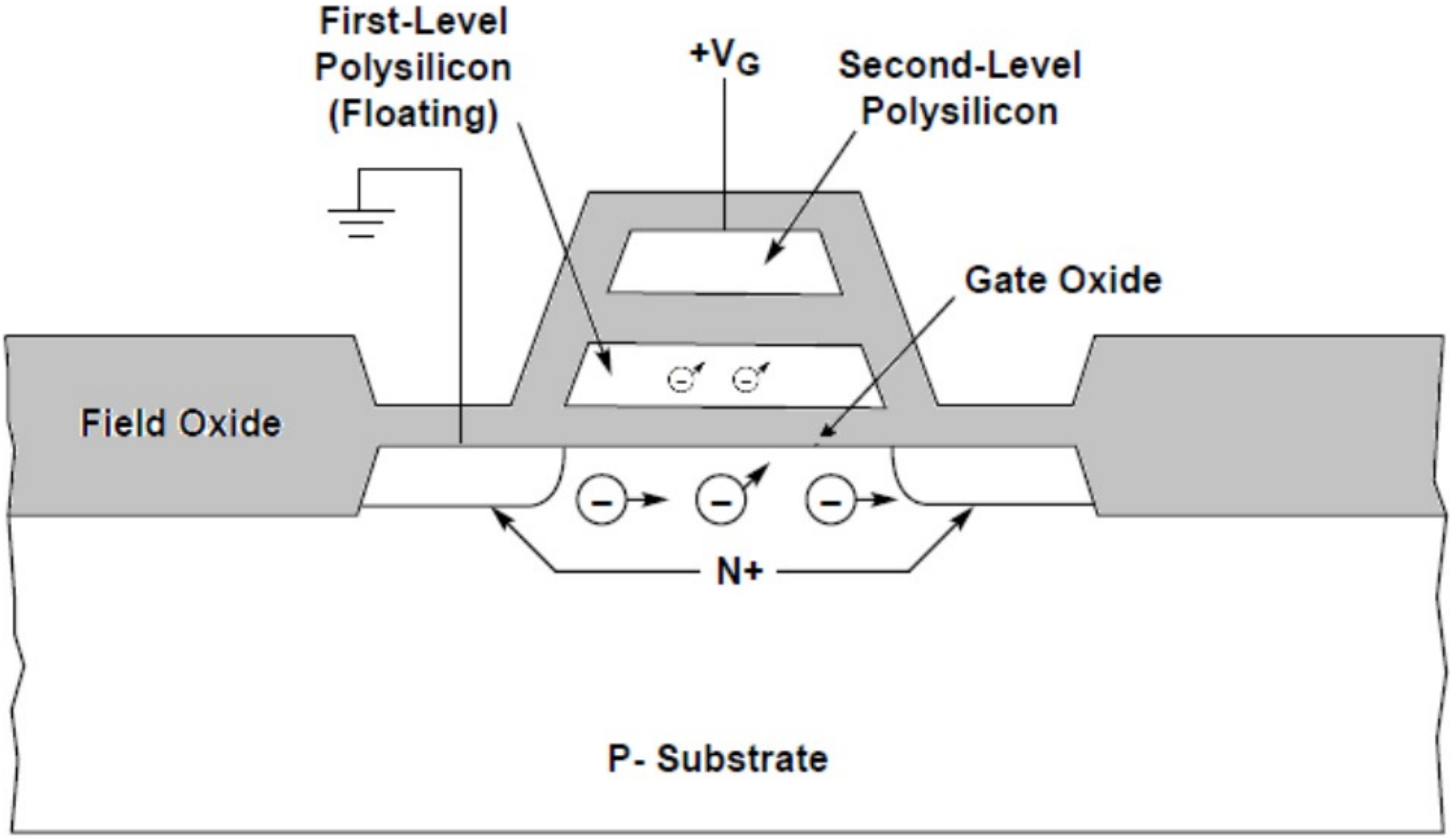


Видове постоянна памет

EPROM съхранява информацията чрез електрически заряд, заключен в т.нар. **плаващ гейт**. Плаващ гейт е полисилициев материал заровен в диелектрика между гейта и канала на MOS транзистор.

Когато се подаде напрежение между гейта и дрейна, електрони с голяма енергия се пренасят към и остават в плаващия гейт. Това води до промяна на праговото напрежение на транзистора, което всъщност се води запаметяване на 0 или 1.

Видове постоянна памет



Source: Intel/ICE, "Memory 1997" 18474

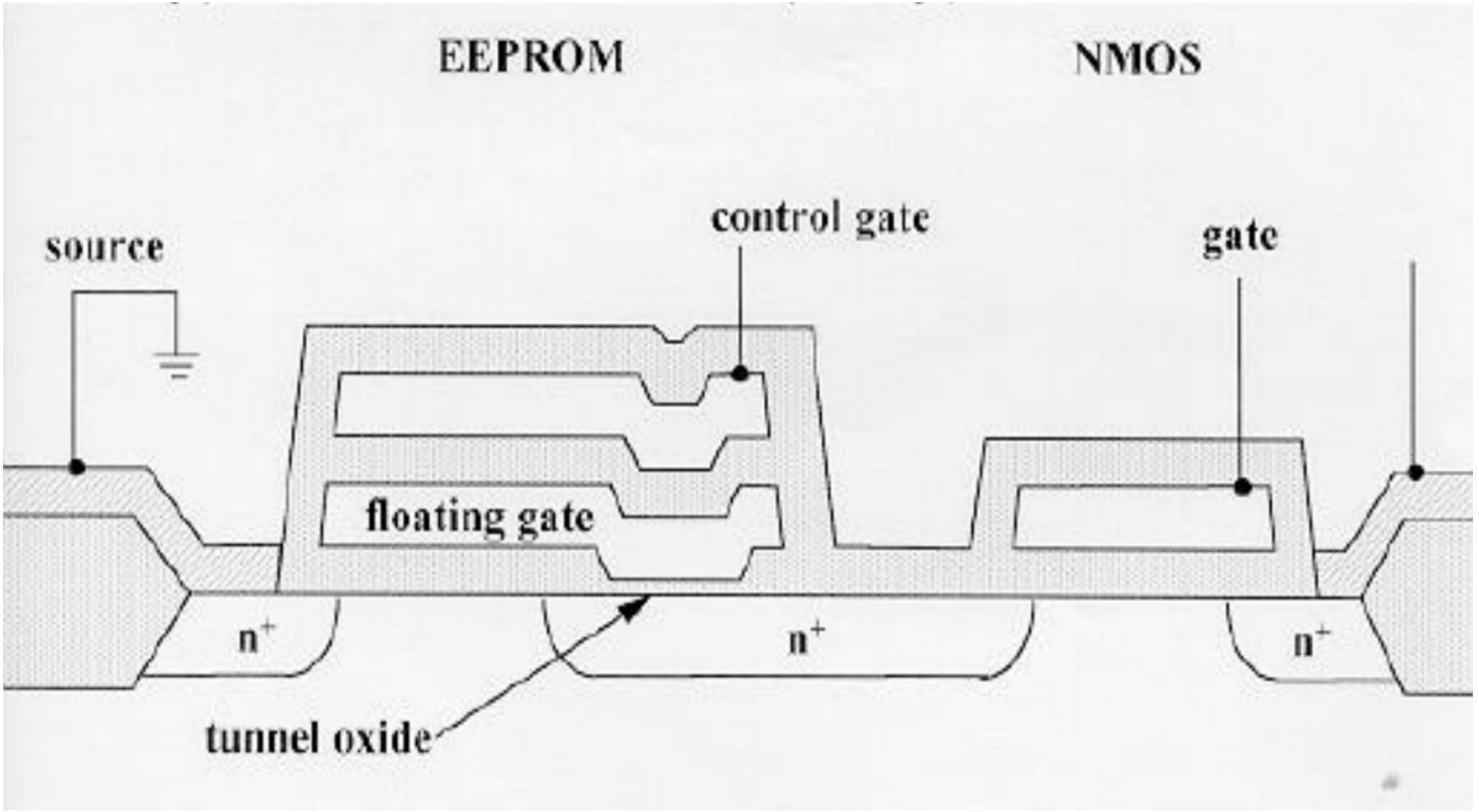
Figure 9-4. Double-Poly Structure (EPROM/Flash Memory Cell)

Видове постоянна памет

EEPROM (Electrically Erasable Programmable Read Only Memory) – енергонезависима памет, която се записва и трие от потребителя чрез електричество.

Вътрешната структура е аналогична на EPROM с една съществена разлика – изолацията на дрейна се прави по-тънка. Това позволява запаметените електрони (заряд) да бъдат разсеяни обратно в дрейна при триене на чипа.

Видове постоянна памет



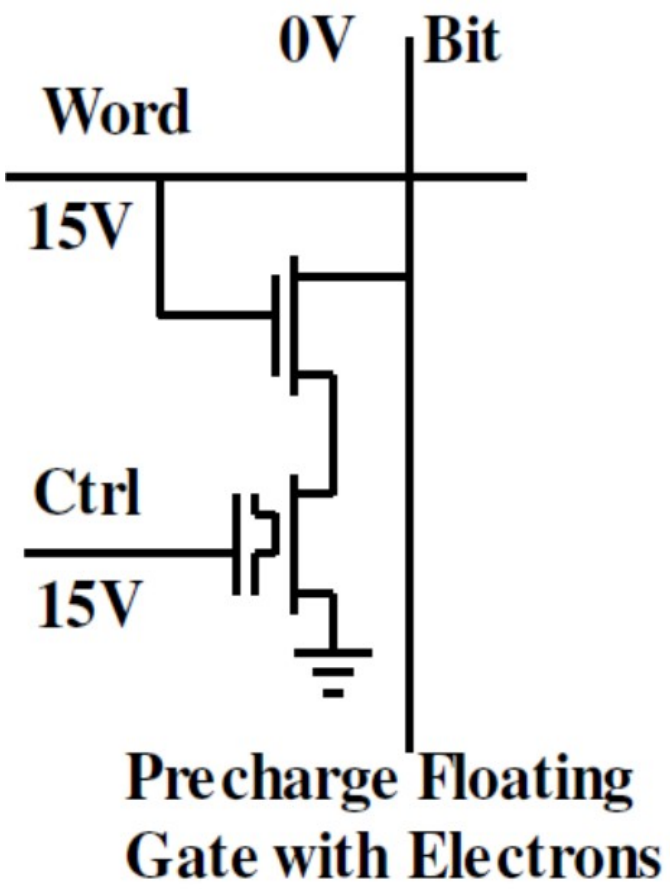
Видове постоянна памет

Триенето се извършва чрез подаване на високо напрежение на дрейна, докато гейта се задържа в 0 V. (Допълнителният транзистор за програмиране се нарича “select” транзистор).

*Обикновено един select-MOS се свързва с други 8 транзистора с плаващ гейт, което позволява триене на ниво **1 байт**.

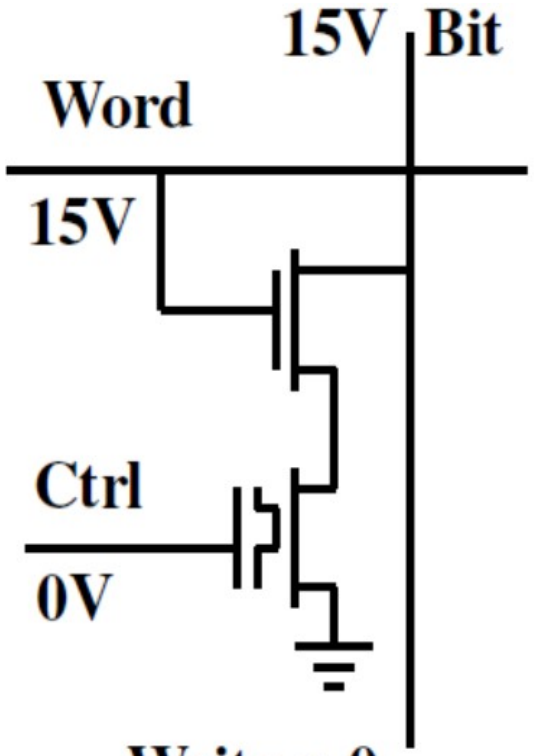
*Циклите запис/презапис в тези памети са в обхвата 100 000 ÷ 1000 000.

Видове постоянна памет



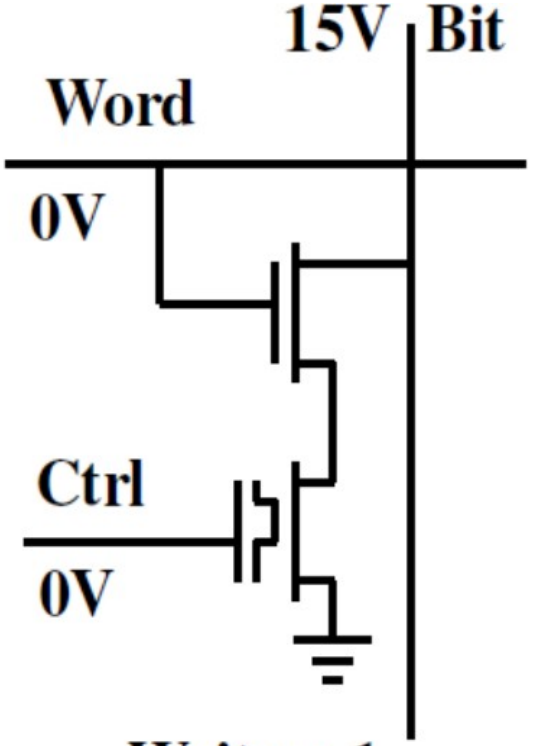
Precharge Floating Gate with Electrons

Първо всички клетки се записват с 1-ци.



Write a 0
Remove Electrons From Floating Gate

Ако битът трябва да е 0:
$V_{GS}=0$, $V_D=V_{HIGH}$.



Write a 1
Leave Electrons on Floating Gate

Ако битът трябва да е 1: $V_{GS}=0$, $V_D=0$.

Видове постоянна памет

Flash - енергонезависима памет, която се записва и трие от потребителя чрез електричество.

Недостатъци **-** използва **същата технология като EEPROM**, но някои от параметрите ѝ са влошени:
*Схемотехниката позволява да се трият само цели **блокове** или **сектори** (а не байт-по-байт като при EEPROM).
*Циклите запис/триене са в обхвата 1000 ÷ 100 000.

Предимства – въпреки, че записа/триенето не може да стане байт по байт, работата с блокове и сектори води до повишаване на бързодействието спрямо EEPROM-а.

Видове постоянна памет

В зависимост от използваната схемотехника, Flash паметите се разделят на два вида:

*__NOR flash__ – клетките (транзисторите) с плаващ гейт са свързани към адресната шина в паралел, което позволява достъп до отделни байтове (като при EEPROM).

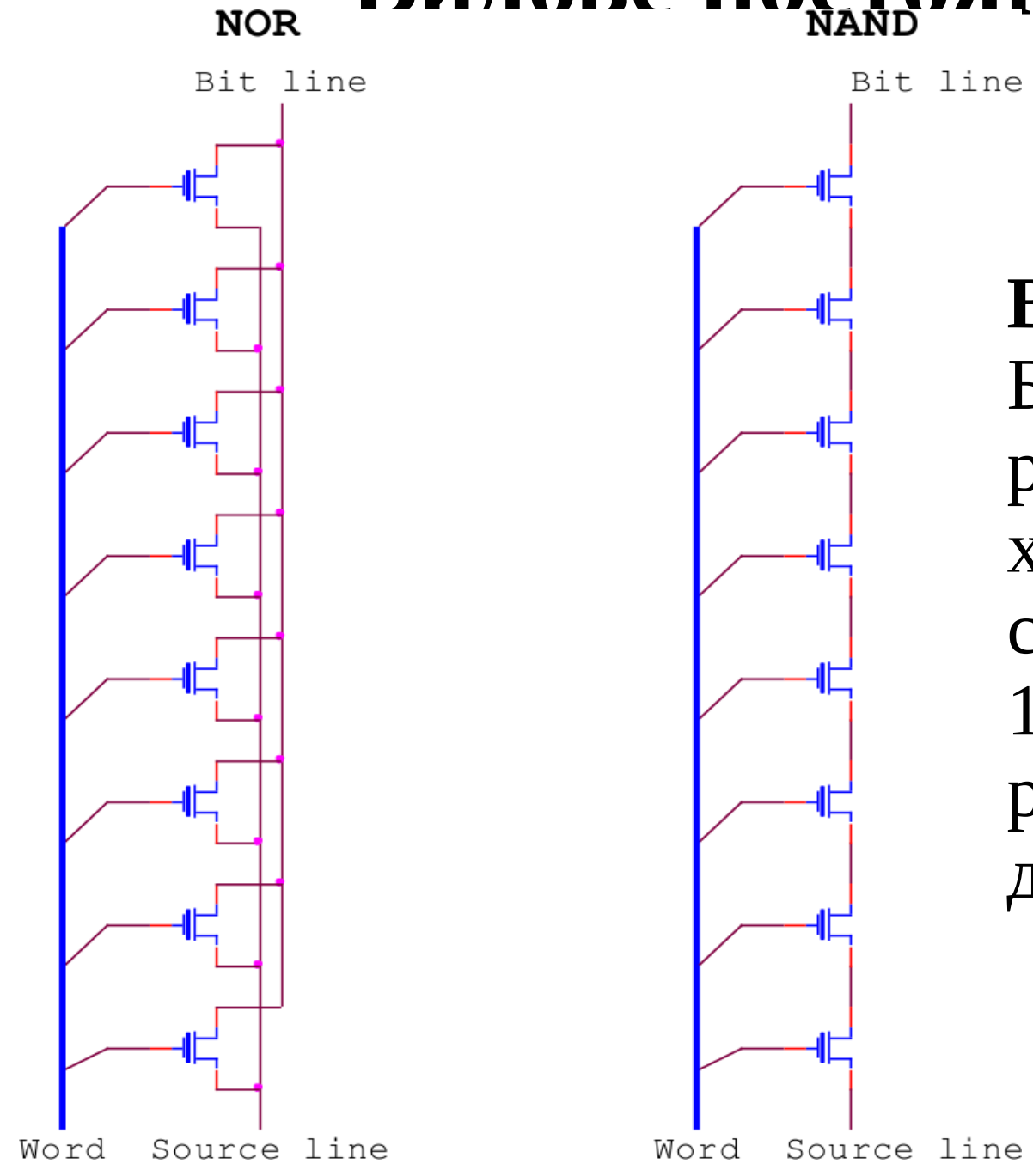
*__NAND flash -__ клетките (транзисторите) с плаващ гейт се свързват към адресната шина последователно, което позволява достъп на блокове и сектори.

Видове постоянна памет

*__NOR flash__ са подходящи за контролни приложения (control) – заемат повече площ на кристала (имат повече метални контакти, [1]) и по-бавно се записват/трият. От друга страна, те се четат по-бързо, т.е. може да се използват като програмна памет.

*__NAND flash__ са подходящи за съхранение на големи обеми от данни (storage), защото заемат малко място на кристала. Записват се и се трият по-бързо от NOR, но се четат по-бавно от NOR.

Видове постоянна памет



ВНИМАНИЕ!
Байтовете са разположени хоризонтално, т.е. схемата показва само по 1 бит от 8 байта разположени един върху друг.

Видове постоянна памет

За да се прочете даден бит в NAND се използва следния **аналогов** метод:

*подава се логическа 1 на сигнала Bit line;

*към транзисторът, който ще се чете, се подава напрежение близко до праговото на гейта;

*на гейтовете на всички останали битове от колоната се подават по-високи напрежения, за да се отпушат на 100% без значение дали е записана 0 или 1 в тях;

*ако в **гейта има заряд, праговото му напрежение ще се е повишило** и транзисторът няма да се отпуши. На Source line = 0V

*ако в **гейта няма заряд, праговото му напрежение ще е ниско** и транзисторът ще се отпуши. На Source line = Bit line - V_{GSth}

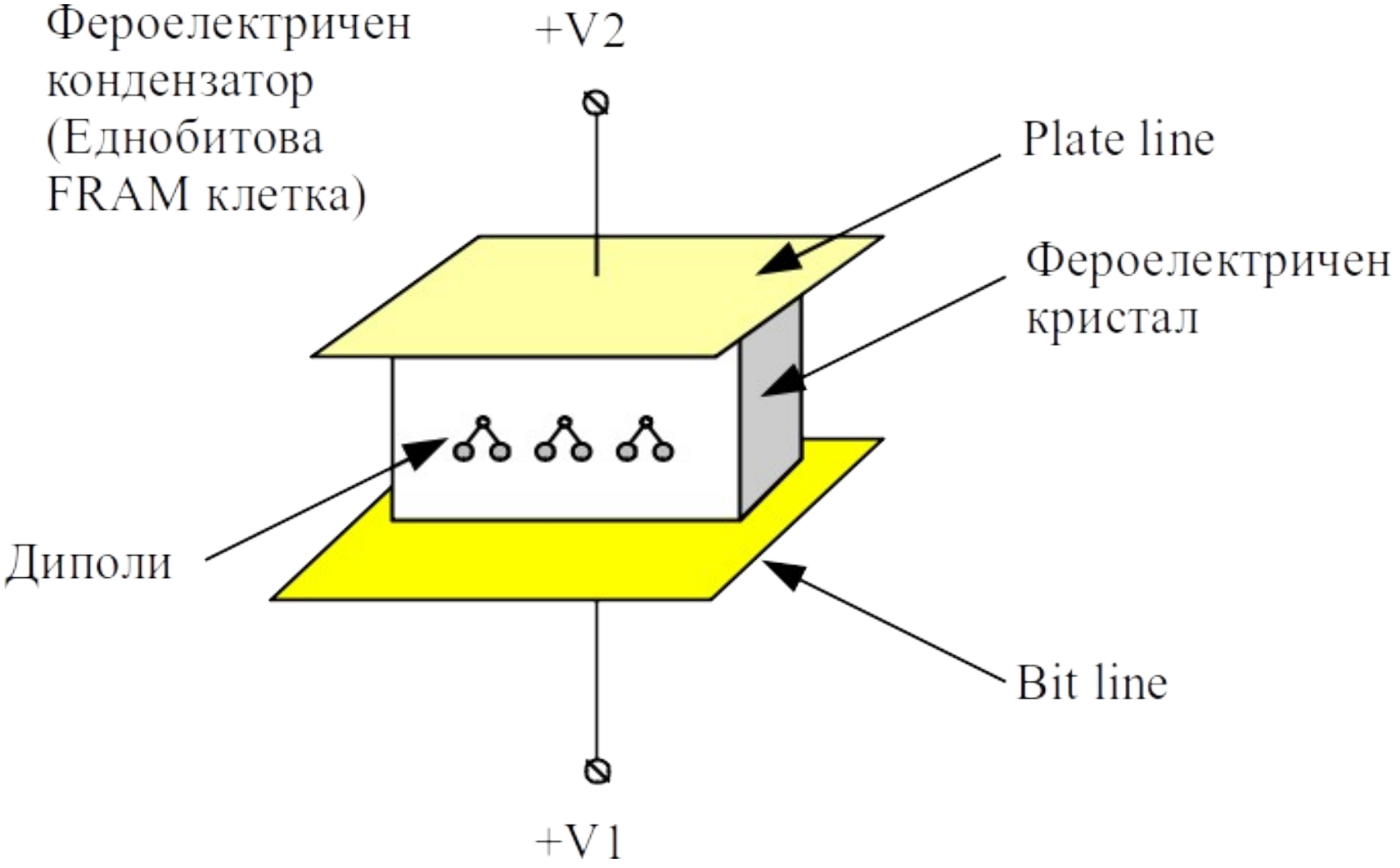
Видове постоянна памет

FRAM (Ferroelectric Random Access Memory) - енергонезависима памет, която се записва и трие от потребителя чрез електричество.

Използва се за **програмна памет**, НО поради големия брой цикли запис/триене, които може да издържи, възможно е приложението ѝ също като **даннова памет** (затова се добавя съкращението RAM).

Информацията се запаметява в керамичен материал, който проявява свойства, подобни на феритните материали.

Видове постоянна памет



Видове постоянна памет

При прилагане на напрежение на електродите V1 и V2, диполите (молекулите) на фероелектричния кондензатор променят ориентацията си.

Сравнителна таблица на Texas Instruments [2]:

Comparison of Memory Attributes

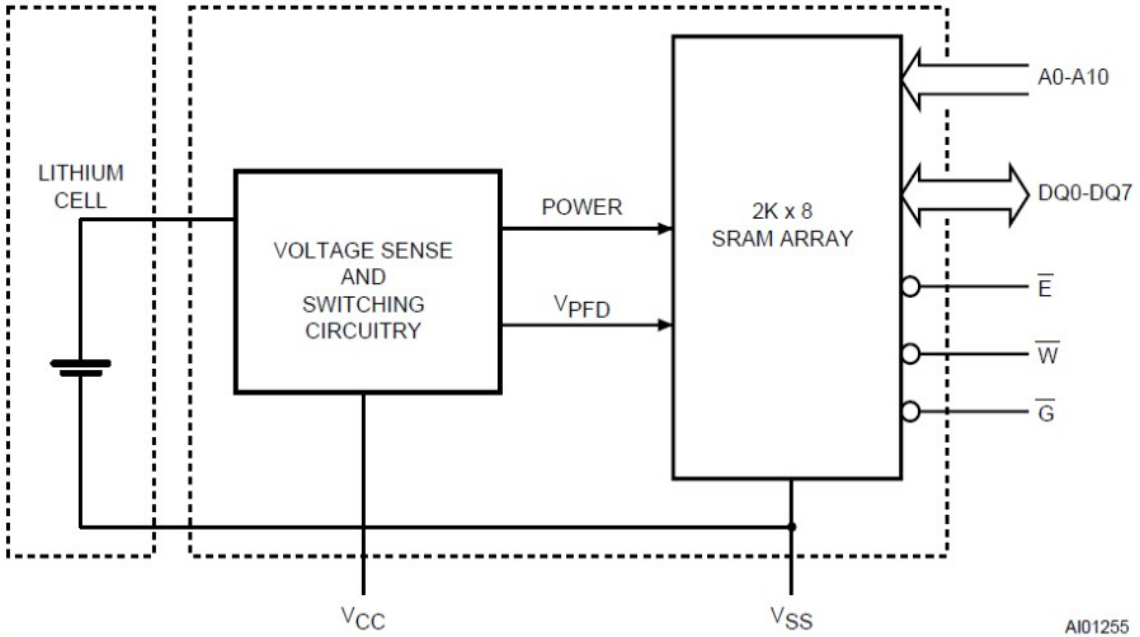
	FRAM	EEPROM	Flash
Time to write 64 bytes to memory	1.6 µs	2,200 µs	6,400 µs
Time to read 64 bytes from memory	1.6 µs	4.5 µs	4.5 µs
Number of write cycles	100 trillion	500,000	100,000
Voltage needed to write	1.5 V	10 to 14 V	10 to 14 V
Manufacturing cycle time	–	>3×	3×
Resistance to gamma radiation	Yes	No	No

Видове постоянна памет

Батерийно-подсигурена RAM (Battery-backed RAM) - енергонезависима памет, която се записва и трие от потребителя чрез електричество [3].

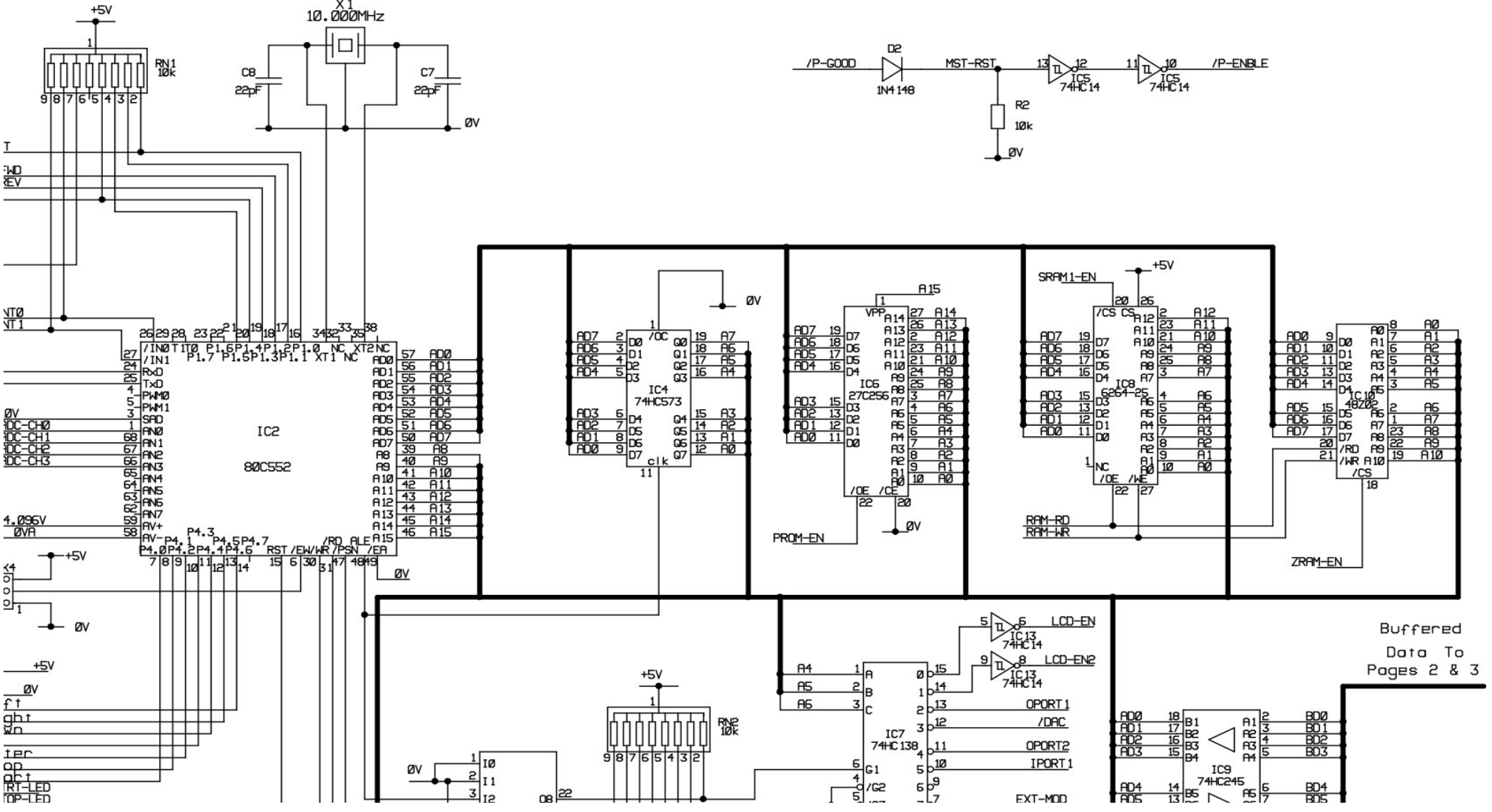
*Това е SRAM с интегрирана батерия в корпуса на чипа
*Ако батерията се изтощи, чипът трябва да се смени
*Среща се още като NVRAM (**N**on-**V**olatile **RAM**) или ZRAM (**Z**eropower **RAM**)
*Има неограничен брой цикли запис/триене
*Кратки времена за четене/запис

Видове постоянна памет



Видове постоянна памет

Пример – машината за катодно разпрашване Emitech K550x използва ZRAM 48Z02 за /най-вероятно/ съхраняване потребителските настройки на уреда.



Видове постоянна памет

Обобщение – паметите с плаващ гейт, показани на миналите слайдове, използват праговото напрежение на гейт-сорс на транзистор, за да запаметят един бит информация. Това означава, че в зависимост от това дали има записана 1 или 0, транзисторът **няма да провежда** или **ще провежда** при подаване на едно и също напрежение на електродите гейт и сорс.

Видове постоянна памет

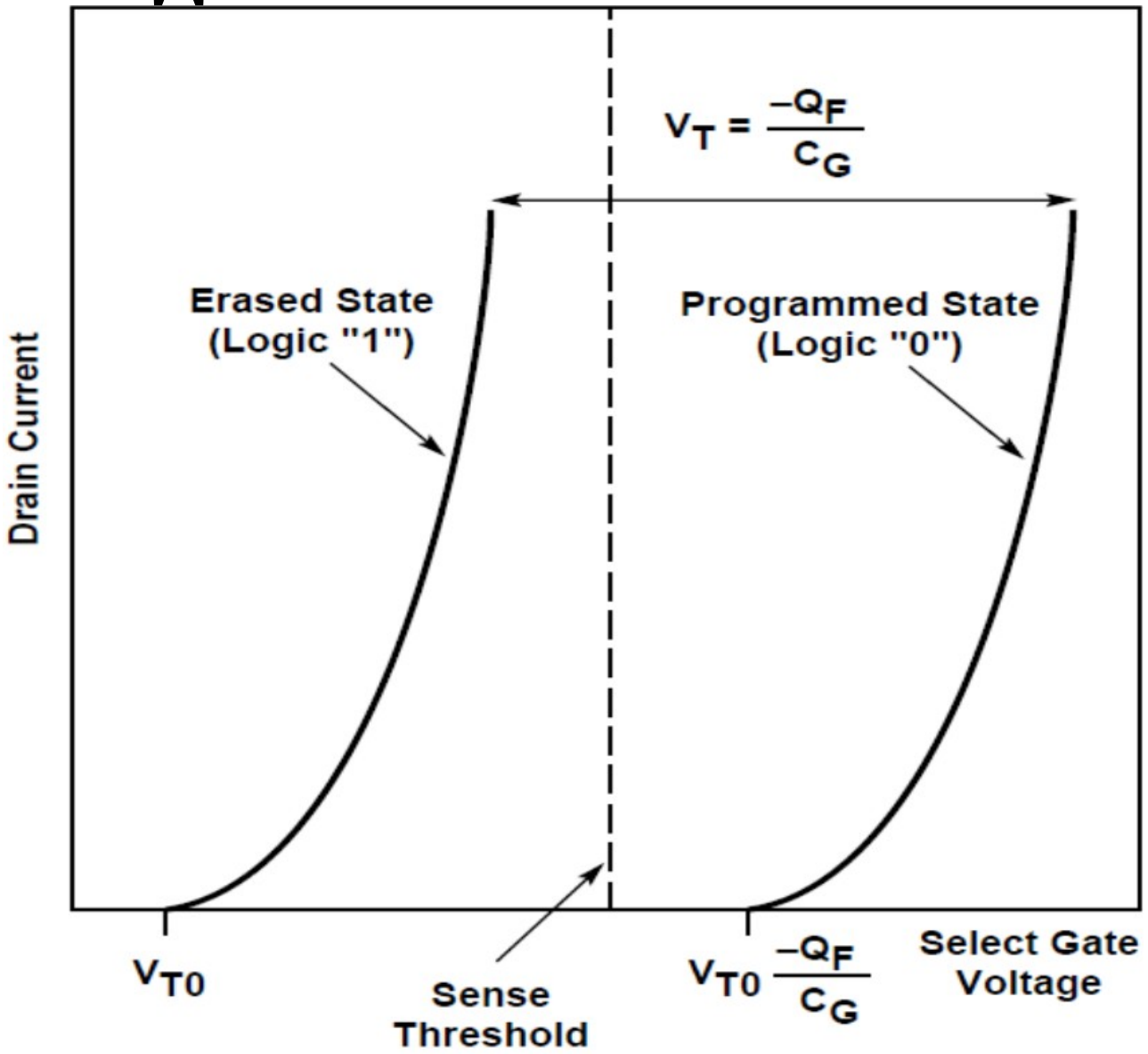
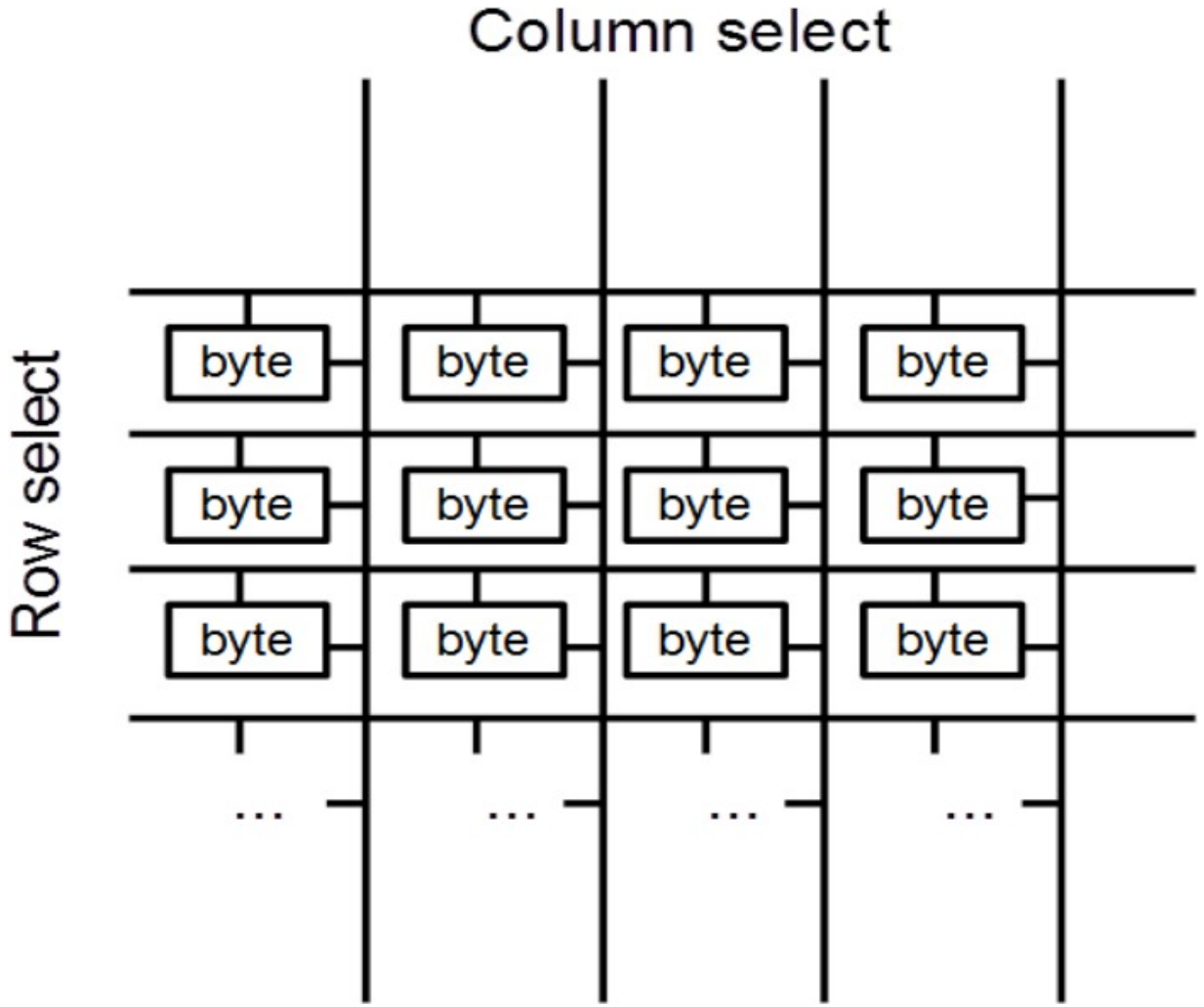


Figure 9-6. Electrical Characteristics of an EPROM

Видове постоянна памет

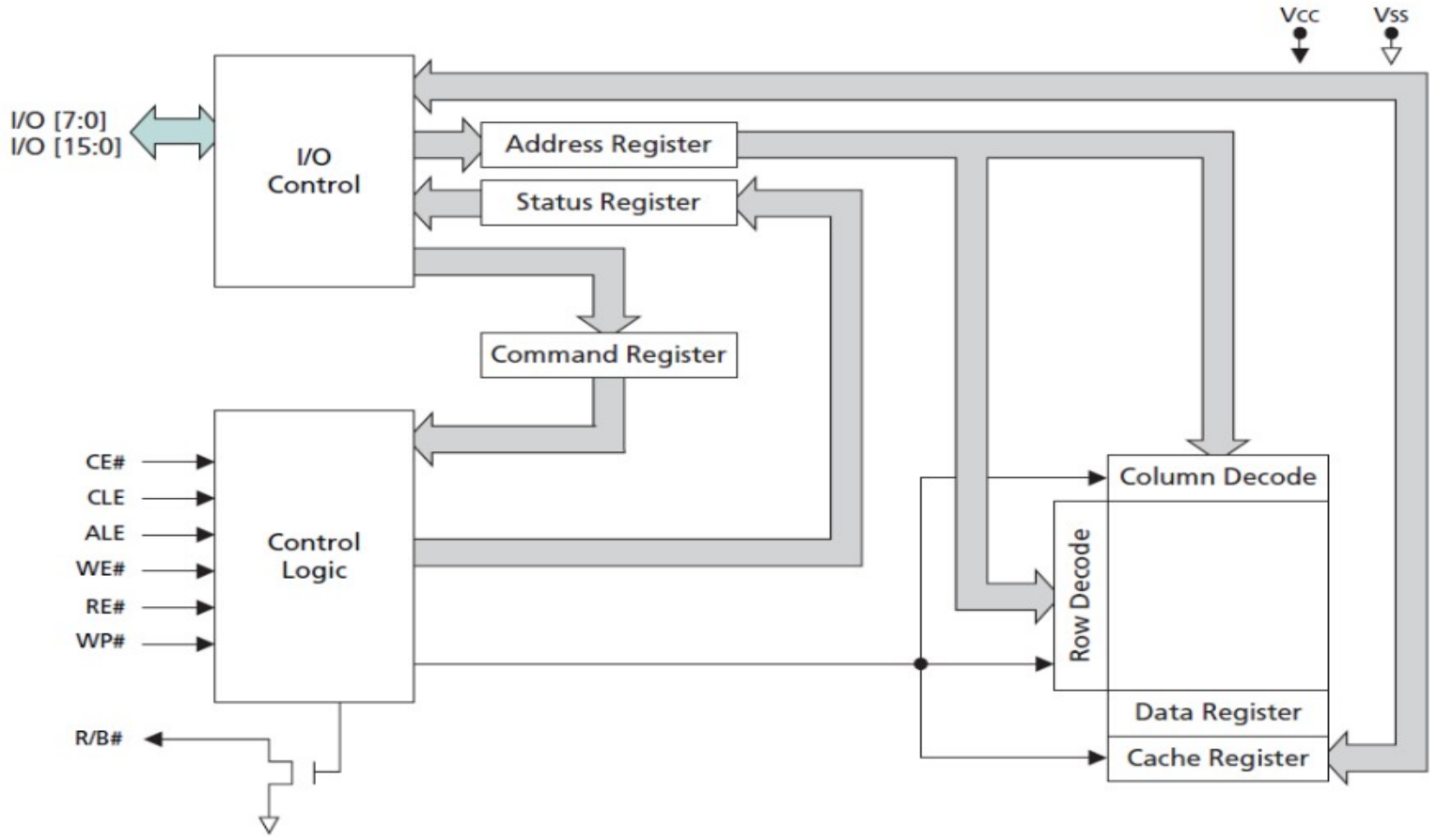
Битовете от паметите се групират в думи, които са свързани в матрица и са адресируеми по колона и ред, както при SRAM и DRAM. Думите може да са по-големи от 1 байт.



Видове постоянна памет

Пример – вътрешна структура на 8 GB NAND Flash чип. Обърнете внимание на декодерите за ред и колона.

NAND Flash Functional Block Diagram



Note: The PRE function is not supported on extended-temperature devices.

Видове постоянна памет

Триенето, записа и четенето на вградени в μCU и външни енергонезависими памети става посредством уред, който се нарича **програматор**.

Чипът, който ще се програмира се поставя в цокъл с изводи повече или равни на чипа, който ще се програмира. Ако изводите на цокъла са повече, чипът трябва **да се постави на точно определено място**. Често се използва ZIF (Zero Insertion Force) цокъл.

Програматорът се свързва към персонален компютър, който използва специализиран софтуер за програмиране на памети и микроконтролери.

Микропроцесорни и ROM емулатори

ROM емулатор (ROM emulator) – уред, който се включва на мястото на външна ROM памет от вградената система и **наподобява поведението на паметта** (=емулира) по такъв начин, че μPU все едно използва оригиналната памет.

ROM емулатора се свързва към персонален компютър посредством стандартен интерфейс (напр. RS232, паралелен порт, и т.н.).

ROM емулатора има **сонда**, която се свързва към цокъла на емулираната памет.

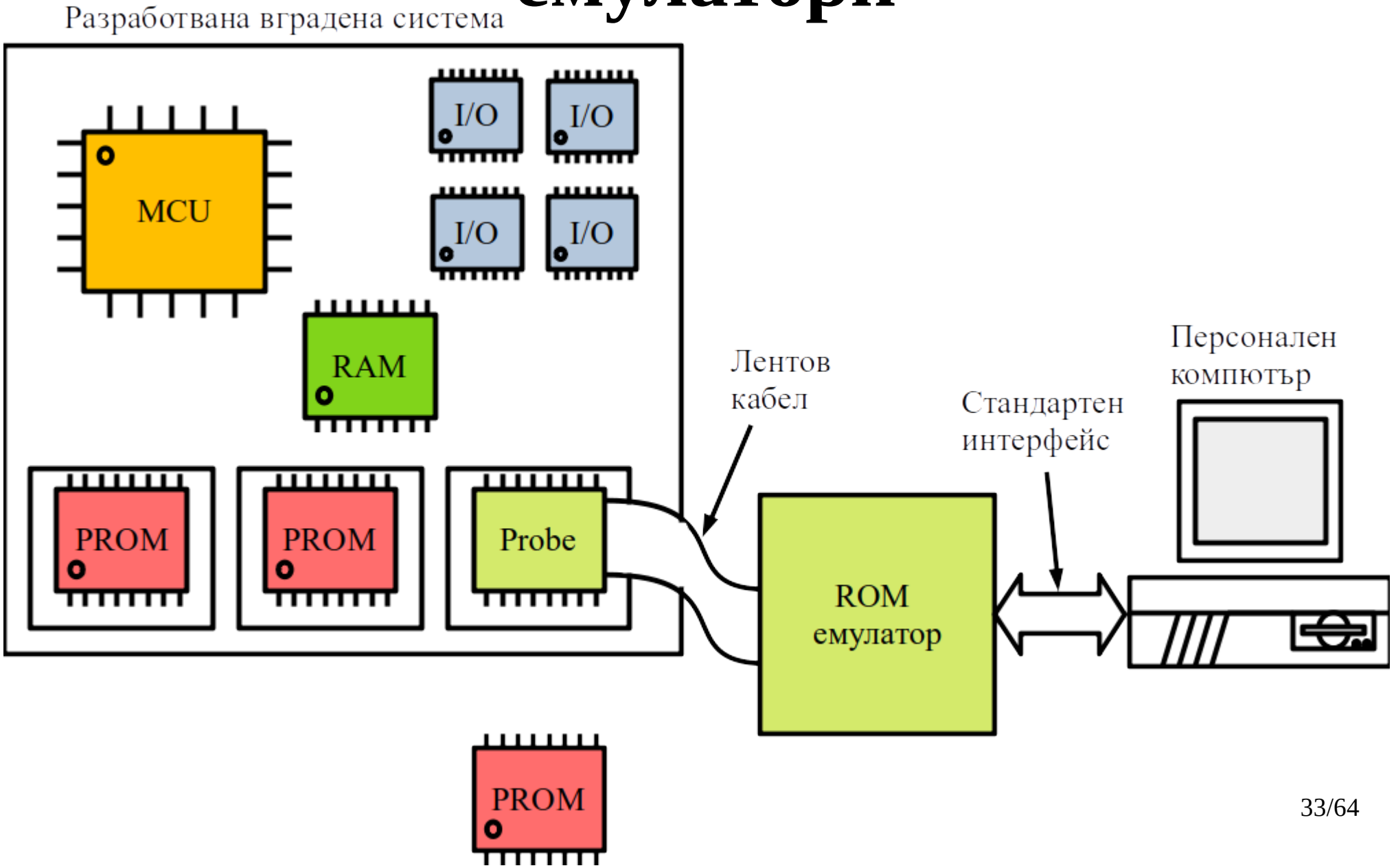
Микропроцесорни и ROM емулатори

На персоналният компютър върви специално приложение, което прехвърля потребителския фърмуер в ROM емулатора (най-често се използва ZRAM).

С помощта на ROM емулатор времето за развой се намалява значително (особено ако се емулира UV EPROM).

ROM емулаторите вече са история (ROM паметите са вградени в микроконтролери, работят с високи тактови честоти).

Микропроцесорни и ROM емулатори



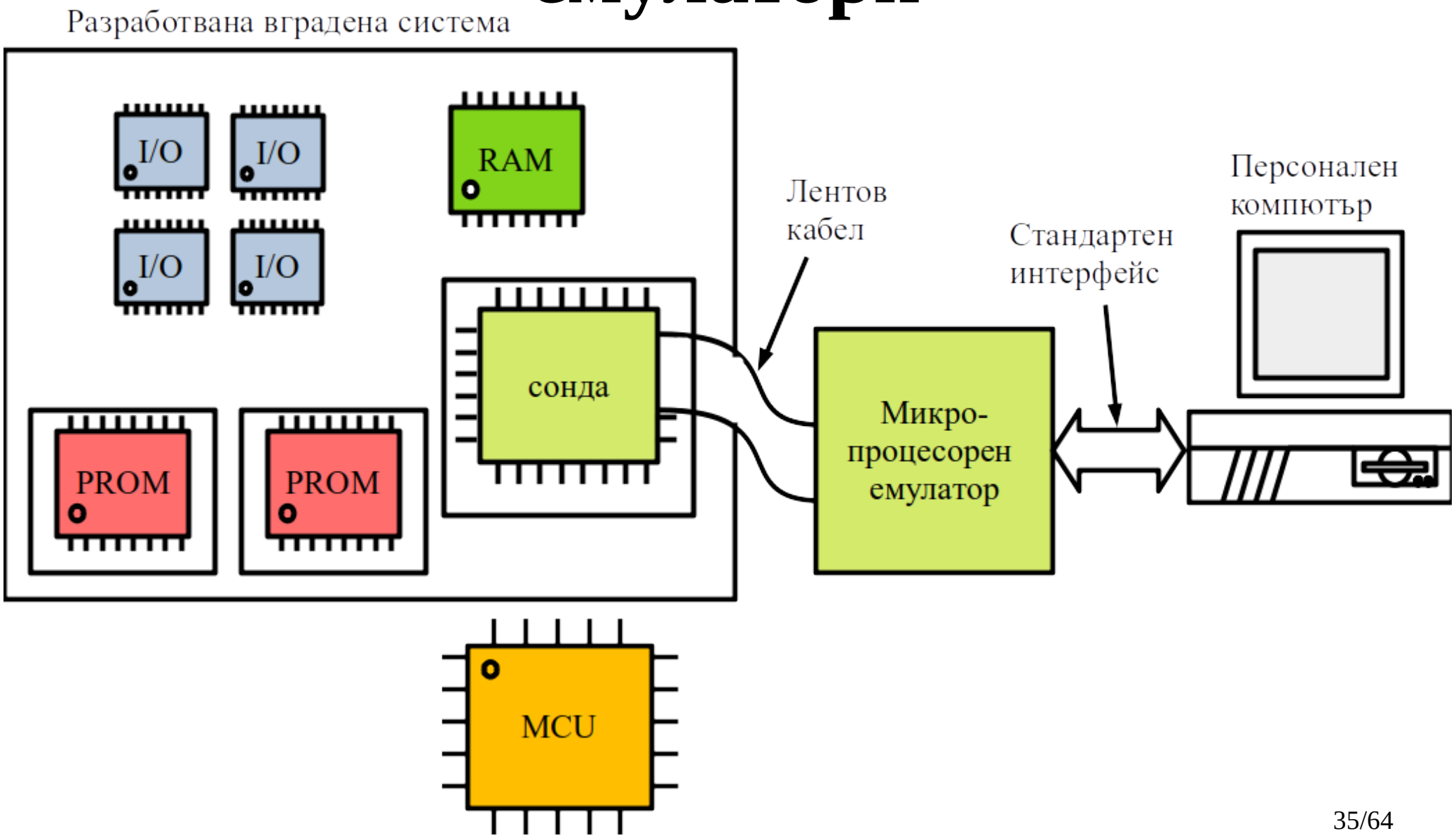
Микропроцесорни и ROM емулатори

Микропроцесорен емулатор (microprocessor emulator) **-** уред, който се включва на мястото на микропроцесора от вградената система (когато той е на отделен чип) и **наподобява поведението на микропроцесора** (=емулира) по такъв начин, че останалата част от системата (периферия и памети) все едно използва оригинала.

С помощта на μPU емулаторите може да се открие дефектирал микропроцесор.

μPU емулатори вече са история (микропроцесорите са вградени в микроконтролери, работят с високи тактови честоти).

Микропроцесорни и ROM емулатори



Микропроцесорни и ROM емулатори

ROM емулаторите и програматорите използват специализиран софтуер, за да прехвърлят потребителския фърмуер във вградената система.

Повечето такива приложения използват следните **файлови формати на фърмуера**:
*.bin (двоичен файл)
*.axf (двоичен файл с дебъгерна информация)
*.elf (двоичен файл с дебъгерна информация)
*.coff (двоичен файл с дебъгерна информация)
*.ihex (Intel hex, текстови файл с ASCII символи)
*.srec (Motorola hex, текстови файл с ASCII символи)
*други

Микропроцесорни и ROM емулатори

Пример – съдържанието на ihex файл е показано по-долу.

```
:0300000020006F5
:03005F0002000399
:030003002006296
:20006200AFA8747F5FF5A81201851201841201761201A43A880439810900002743 3F0E490
:20008200A3F0A3F0A3F0900002E0FCA3E0FDA3E0FEA3E0FF900001ECF043801090002E4FE
:2000A200F0A3F0A3F0A3F0900002E0FCA3E0FDA3E0FEA3E0FFC3EC9488ED9413EE9400EFD4
:2000C2009400501890002E02401F0A3E03400F0A3E03400F0A3E03400F080CBAF8074EFC9
:2000E2005FF580900002E4F0A3F0A3F0A3F0900002E0FCA3E0FDA3E0FEA3E0FFC3EC94884F
:20010200ED9413EE9400EF9400501890002E02401F0A3E03400F0A3E03400F0A3E0340050
:20012200F080CB900002E02401F0A3E03400F0A3E03400F0A3E03400F0900001E0FF3395CE
:16014200E0FE8F828E83120158AE82AF83900001EEF002009B22AC
:20015800AF83E582900006F0EFA3F0900006E0FEA3E0FF740A2EFEE43F8E82F5832275867E
:20017800012290 2F11E4F0902F12F022243981090 2F00E4F0902F08F0902F50F0AF8074C4
:05019800EF5FF580227D
:06003500E478FFF6D8FD9F
:200013007900E94400601B7A009001A1780875A000E493F2A308B8000205A0D9F4DAF275EA
:02003300A0FF2C
:20003B007800E84400600A790175A000E4F309D8FC7807E84400600C7901900001E4F0A3BB
:04005B00D8FCD9FAFA
:0D00060075811F12019DE58260030200 0359
:04019D007582002245
:00000001FF
```

Микропроцесорни и ROM емулатори

Форматът на всеки ред е:
:ccaaaarrddss

: - всеки ред започва с двуеточие
cc – брой байтове на реда
aaaa – адресът на първия байт от реда
rr – вид данни на реда (ако е 00 – потребителски байтове, ако е 01 – край на файла)
dd – потребителски байтове на инструкции/данни, може да са 0 ÷ 255 броя
ss – 8-битова контролна сума на предшестващите байтове (two's complement):
(~[cc + aa_H + aa_L + rr + SUM(dd)]) + 1 = ss

Интерфейс JTAG

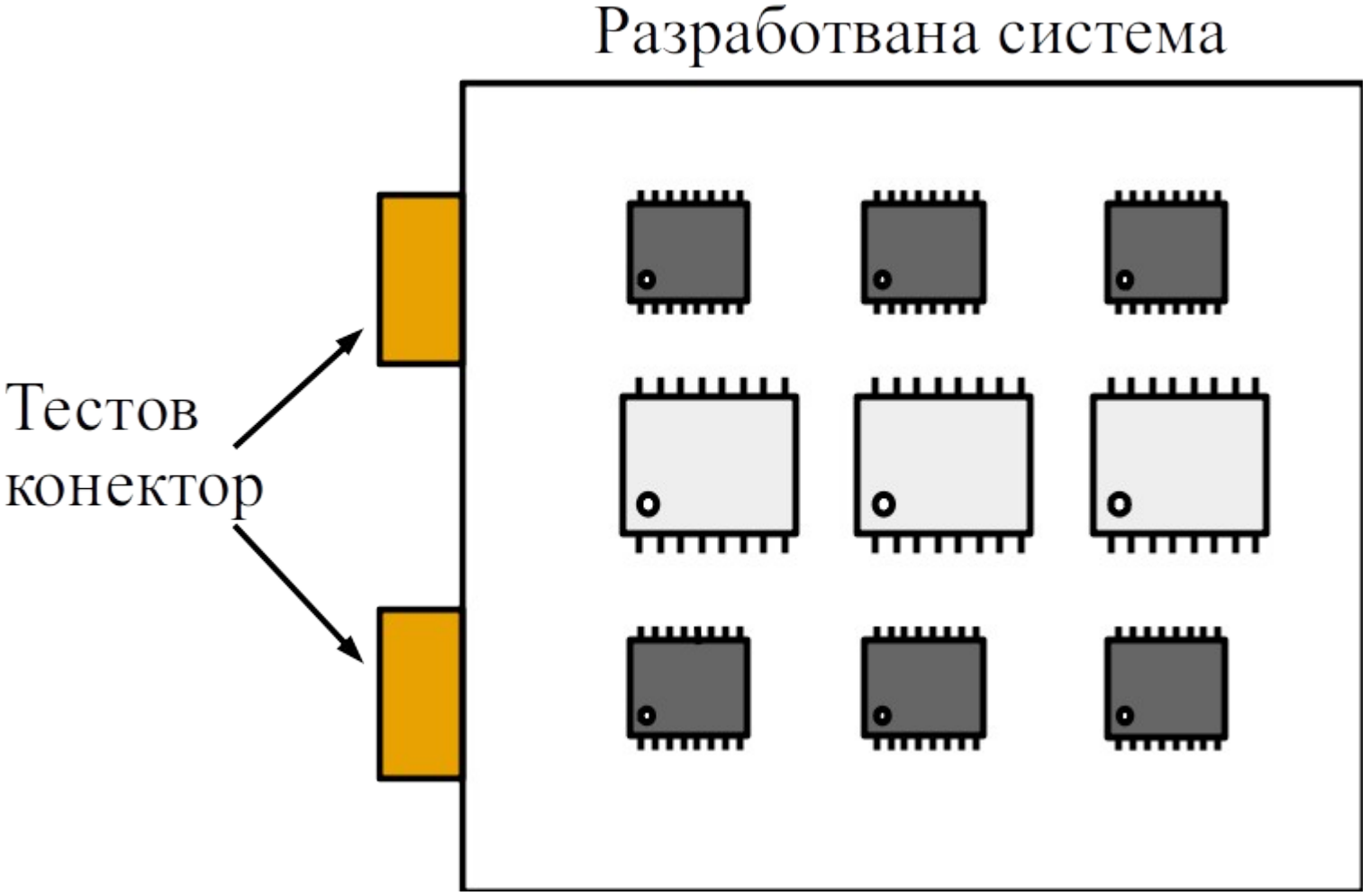
Печатните платки (**P**rinted **C**ircuit **B**oard, PCB) на електронни уреди трябва да се тестват веднага щом слязат от производствената линия.

В миналото PCB са били прости, и съответно – процедурите за тест са били също прости.

За целите на тестването, инженерите вграждали тестови конектори на PCB.

На конекторите се извеждали сигнали от важни възли на схемата.

Интерфейс JTAG



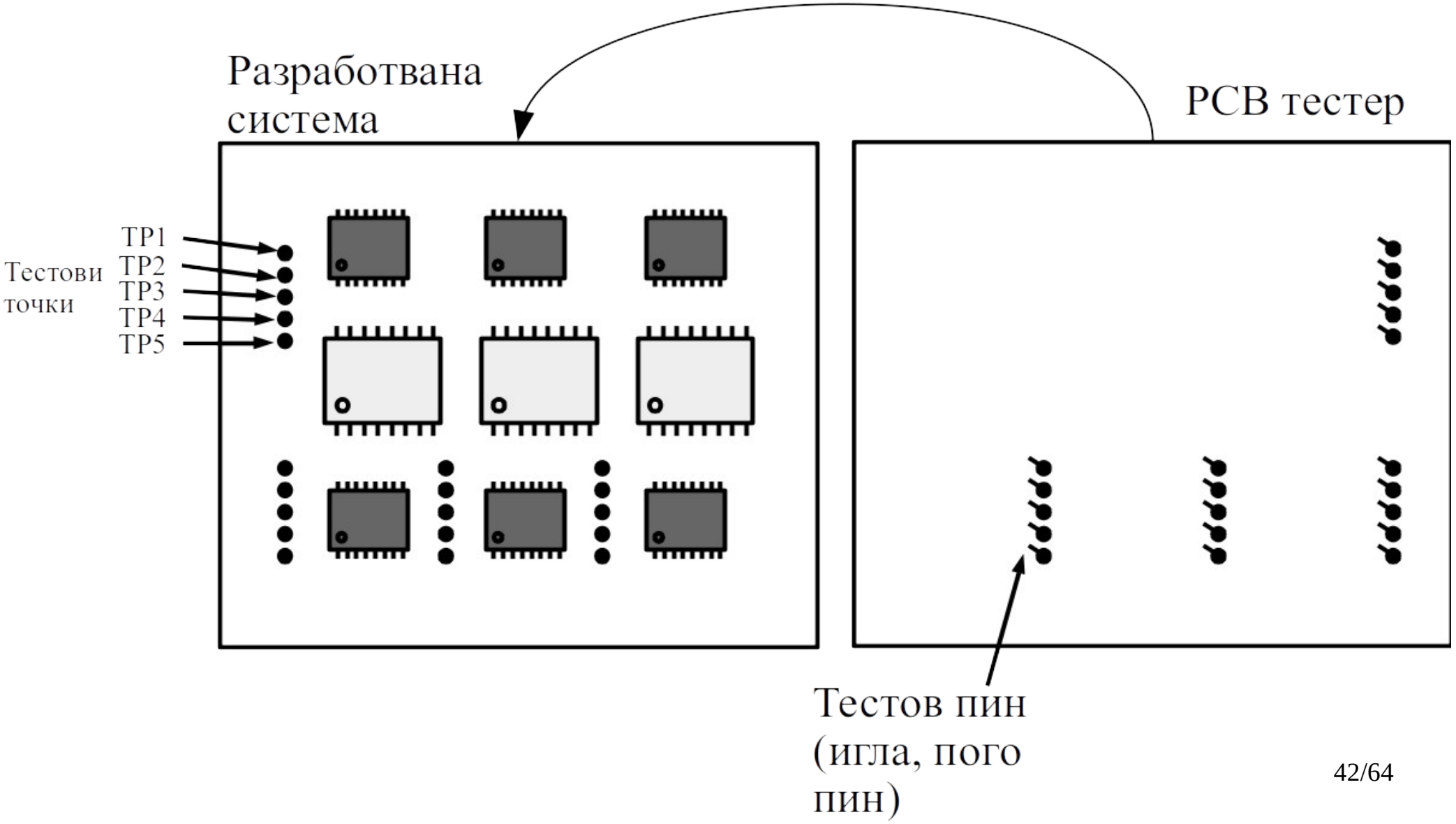
Интерфейс JTAG

С увеличаване сложността на проектираните платки, инженерите започнали да слагат тестови точки на самата платка [5].

*По този начин по-голям брой сигнали е било възможно да се тестват.

*Тестовите точки се достъпвали със специално разработени за целта тестери, оборудвани с контактни игли (пого пинове).

Интерфейс JTAG



Интерфейс JTAG

Сложността на платките продължила да расте, и тестовите точки вече не били достатъчни за пълен тест.

Производителите на чипове решили да добавят т.нар. **гранична сканираща логика** (**B**oundary **S**can **L**ogic, BSL), която се използва за четене на входни и изходни точки на интегралната схема.

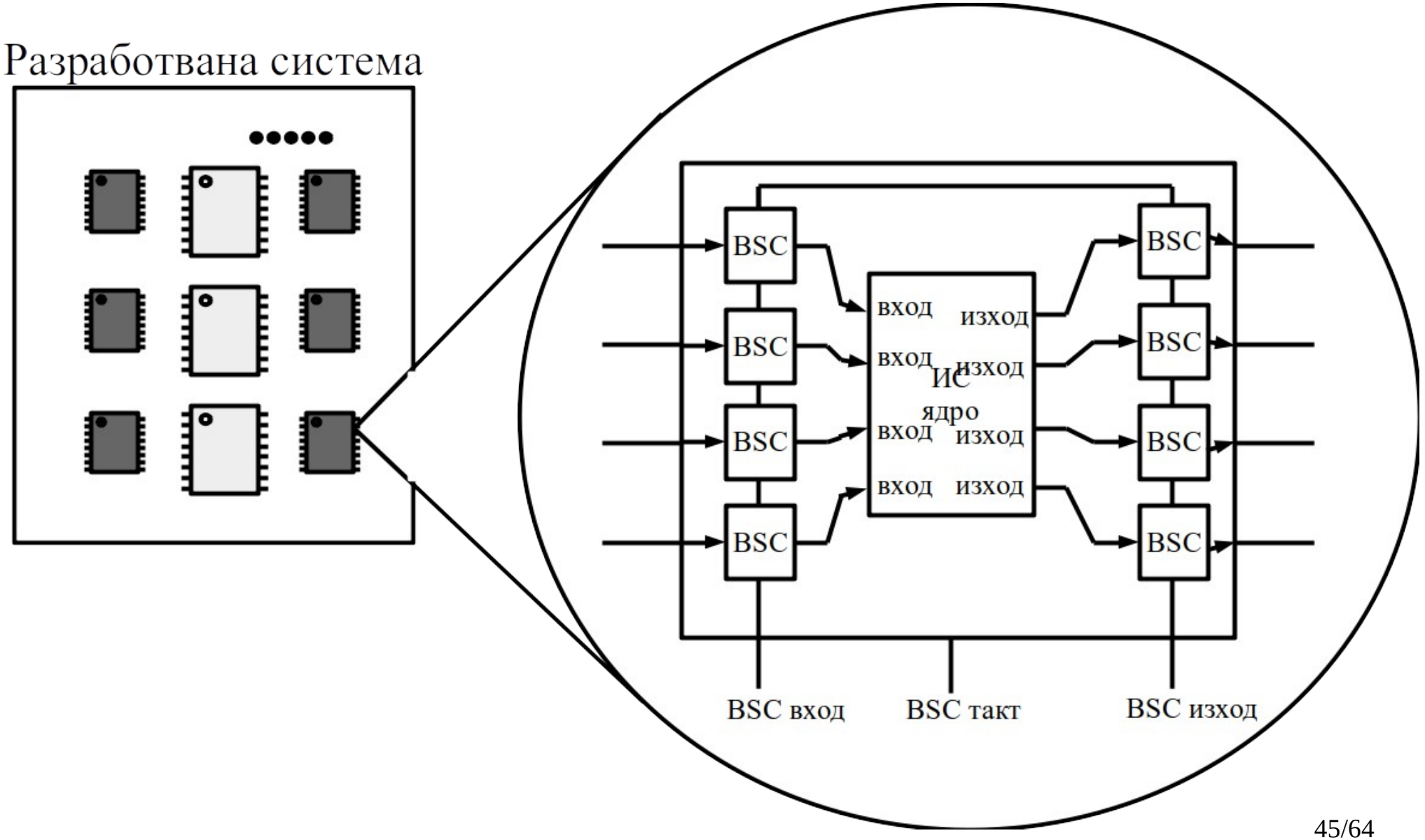
Ядрото на BSL са **граничните сканиращи клетки** (**B**oundary **S**can **C**ell, BSC), които представляват еднобитови преместващи регистри, свързани на входовете и изходите на ИС. Тези регистри са свързани последователно и формират един многоразреден преместващ регистър, който може да бъде записван и четен.

Интерфейс JTAG

Данните в BSC се записват/четат под управлението на **тактов сигнал**.

Понеже **образуваният регистър е сериен**, то и изводите, необходими за достъпа му са **малко на брой.** Оттук идва драстичното намаляване на тестовите точки върху самата платка.

Интерфейс JTAG



Интерфейс JTAG

Към BSC трябва да се добави управляваща логика:
***TAP контролер** (Test Access Port) – машина на състоянието, която контролира цялата вътрешна BSL, така че дадена тестова операция да се изпълни.

TAP контролерът включва в себе си два вида регистри:
***Регистър на инструкцията** – 8-битов регистър, съдържащ число, което отговаря на:
→ тестовата операция, която трябва да се изпълни;
→ режим на работа на BSL
→ регистри, които да бъдат включени в BSL веригата
→ източници на данни за данновите регистри

Интерфейс JTAG

Даннови регистри – набор от регистри, специфични за всеки чип, но винаги съдържащи общите регистри BSR, BYPASS, IDCODES. Понякога към общите регистри се включва и BCR регистъра.

***BSR** (**B**oundary **S**can **R**egister) – преместващ регистър, изграден от всички BSC на чипа. Отразява настоящото състояние на входовете/изходите на чипа, както и осигурява възможността за промяната им от потребителя.

***BYPASS регистър** – използва се за съкращаване на BSR регистъра, когато е необходимо. Той просто свързва входа и изхода на BSR през един еднобитов преместващ регистър.

Интерфейс JTAG

IDCODES регистър – съдържа информация за чипа, като например име на производител, номер на чип, версия, и т.н [6].

BCR регистър (**B**oundary **C**ontrol **R**egister) – 2-битов регистър, използван за разрешаване на допълнителни тестови операции, показани в таблицата по-долу:

Table 3. Boundary-Control Register Opcodes

BINARY CODE BIT 1 → BIT 0 MSB → LSB	DESCRIPTION
00	Sample inputs/toggle outputs (TOPSIP)
01	Pseudo-random pattern generation / 16-bit mode (PRPG)
10	Parallel-signature analysis/16-bit mode (PSA)
11	Simultaneous PSA and PRPG/8-bit mode (PSA/PRPG)

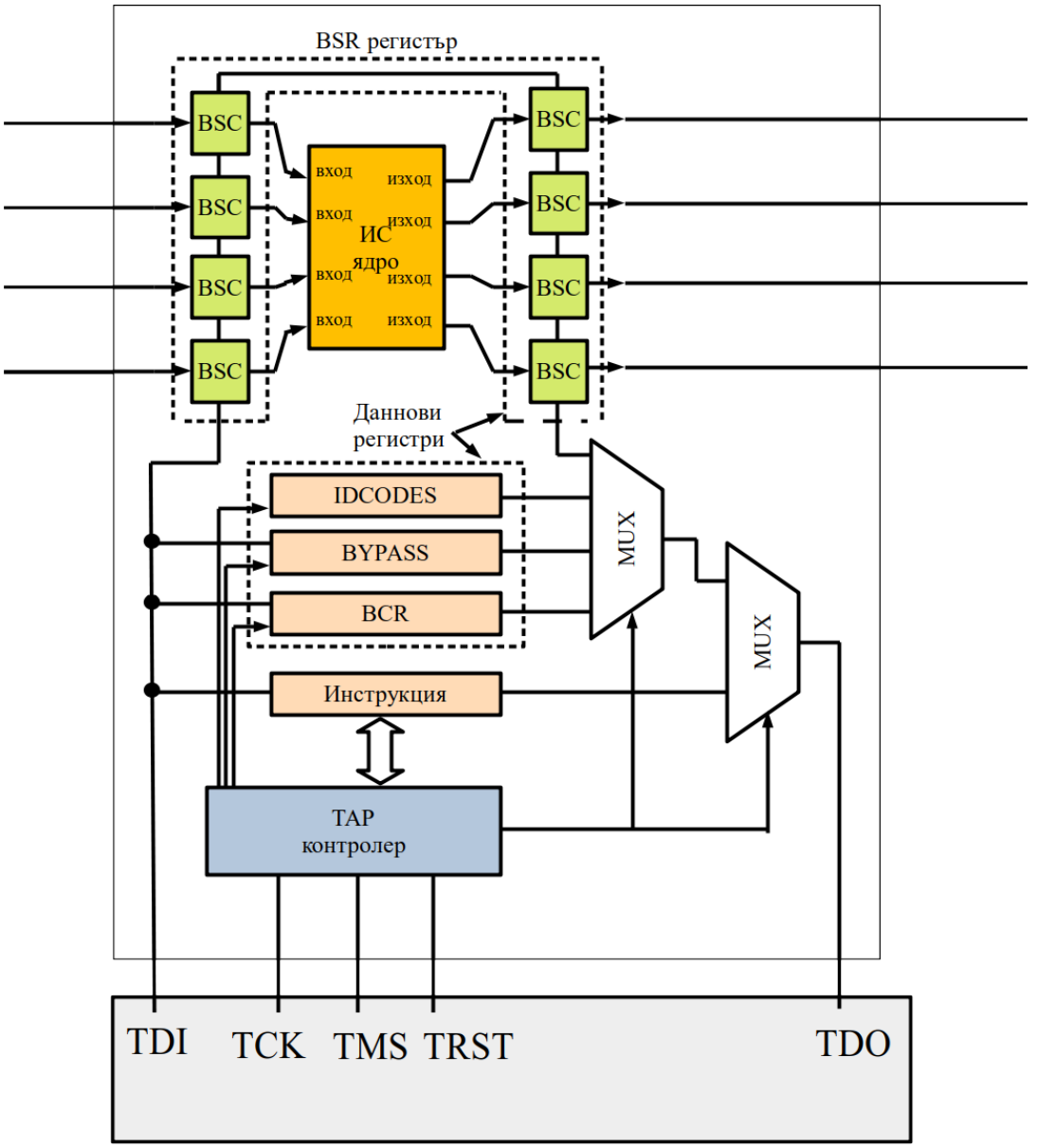
Интерфейс JTAG

JTAG интерфейс (Joint Test Action Group, IEEE 1149.1) – стандартизиран от индустрията интерфейс, използван от инженерите за тестване и програмиране на логически елементи (виж SN74BCT8244A), специализирани чипове (ASIC), програмируеми матрици (FPGA), микропроцесори (μPU), микроконтролери (μCU), цифрови сигнални процесори (DSP) и др.

JTAG е съвкупността от управляващи сигнали на TAP контролера и данновите сигнали на BSR регистъра.

Интерфейс JTAG

Интегрална схема



JTAG интерфейс

Интерфейс JTAG

TCK (Test Clock Input) – тактов вход, използван за синхронизация на влизащите в и излизащите от интерфейса данни и инструкции. Максималната стандартизирана честота е 20 MHz.

TDI (Test Data Input) – вход за данни и инструкции, които се възприемат (latch) по нарастващ фронт на тактовия сигнал. Към извода има свързан издърпващ към захранване резистор.

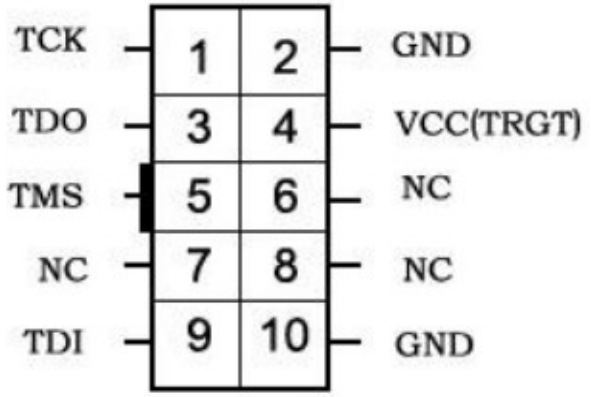
TDO (Test Data Output) – изход за данни и инструкции, които се изработват (shift out) по падащ фронт на тактовия сигнал.

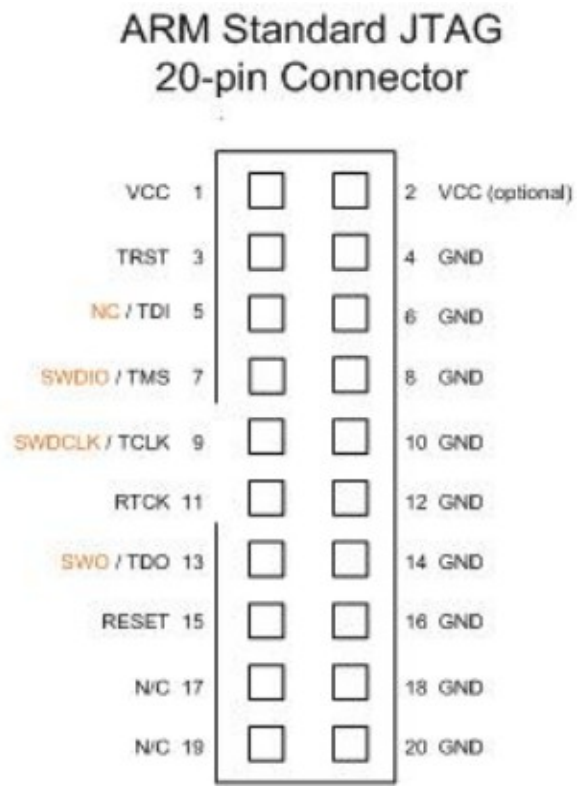
TMS (Test Mode Select) – регулира превключването на машината на състоянията на TAP контролера.

TRST (Test Reset) – извод за ресетиране на TAP контролера. Изводът не е задължително да го има.

Интерфейс JTAG

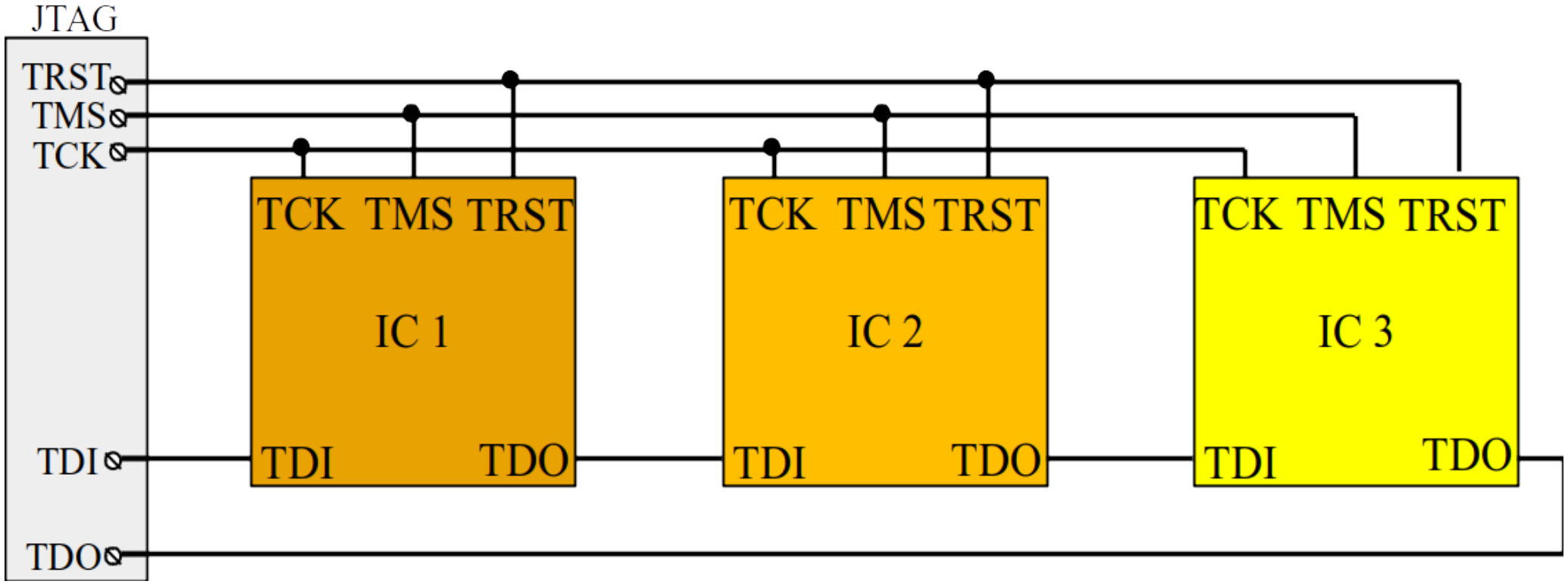
На печатната платка обикновено се поставя 10-изводен или 20 изводен JTAG конектор, който може да включва и други сигнали освен тези на JTAG интерфейса.





Интерфейс JTAG

Методът на използване на преместващ регистър позволява няколко еднакви или различни чипа да бъдат свързани в един голям кръгов буфер, съставен от BSR регистрите на отделните ИС. Нарича се още JTAG верига (JTAG chain).



Интерфейс JTAG

BSDL (Boundary Scan Description Language) – език за описание на хардуер за електронно тестване на устройства посредством JTAG. Езикът е подмножество на VHDL.

Информацията, която един BSDL файл включва е:
*използвани сигнали (TMS, TCK, TDI, TDO, TRST)
*използвани регистри в пътя на JTAG веригата
*поддържани инструкции от TAP контролера

BSDL файловете са необходими на тестващите уреди, за да разпознаят всички чипове във веригата и да настроят параметрите на интерфейса.

Интерфейс JTAG

JTAG дебъгери (JTAG debugger, JTAG adapter) – устройства, които позволяват да се откриват и отстраняват грешките на системния софтуер (фърмуер), както и да се зарежда в паметта на разработваната вградена система.

Затова програматорите са вече история.

JTAG дебъгерите правят транслирането на нивата на сигналите, които отиват към разработваната система, както и преобразуването на комуникационния протокол от един в друг. Целта е да се използва стандартен PC интерфейс, който след дебъгера да стане на JTAG.

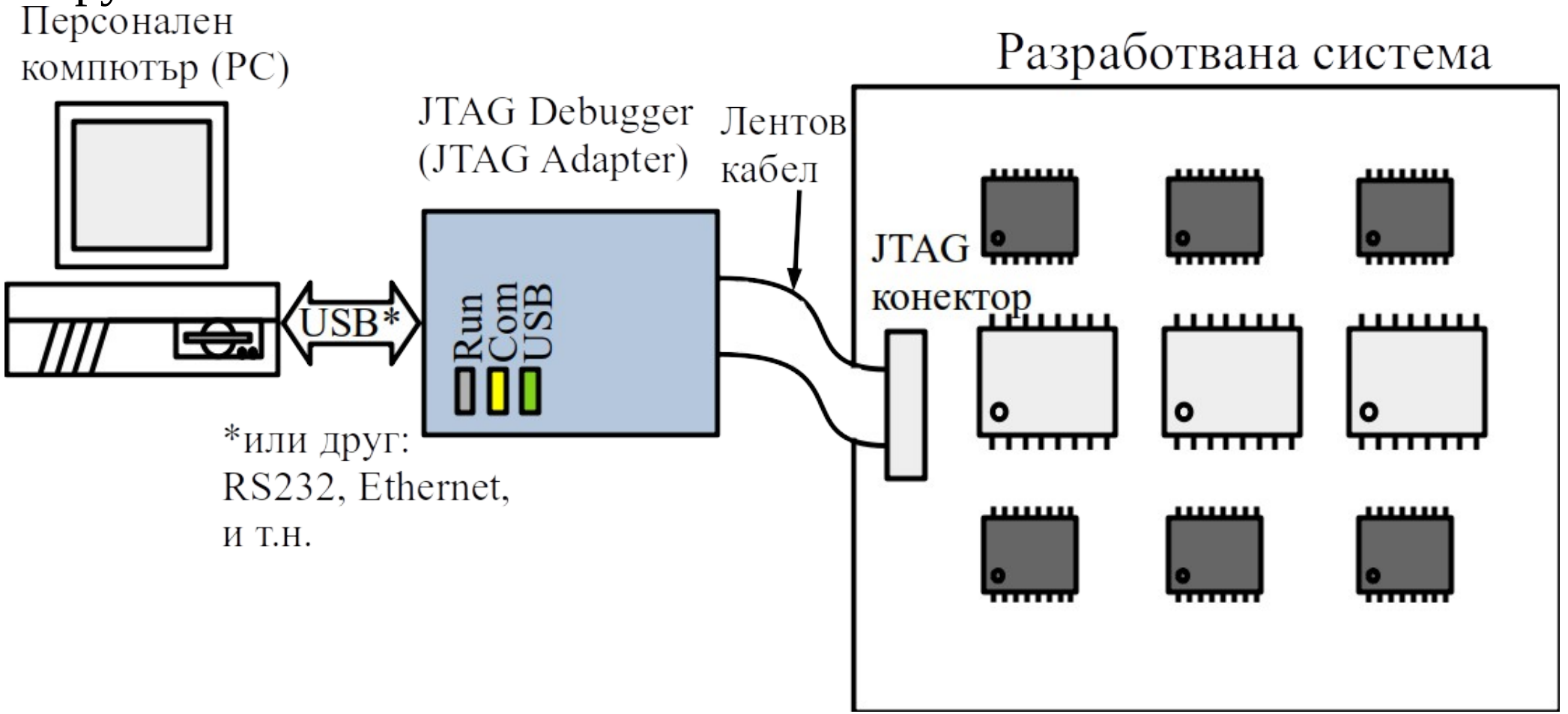
Интерфейс JTAG

JTAG дебъгерите се предлагат като отделни уреди. Това най-често означава, че:

*може да програмира **много различни µCU**, на **различни фирми**;

*може да тества с **високи скорости** (20 MHz);

*струва **30 ÷ 2000 $**.



Интерфейс JTAG

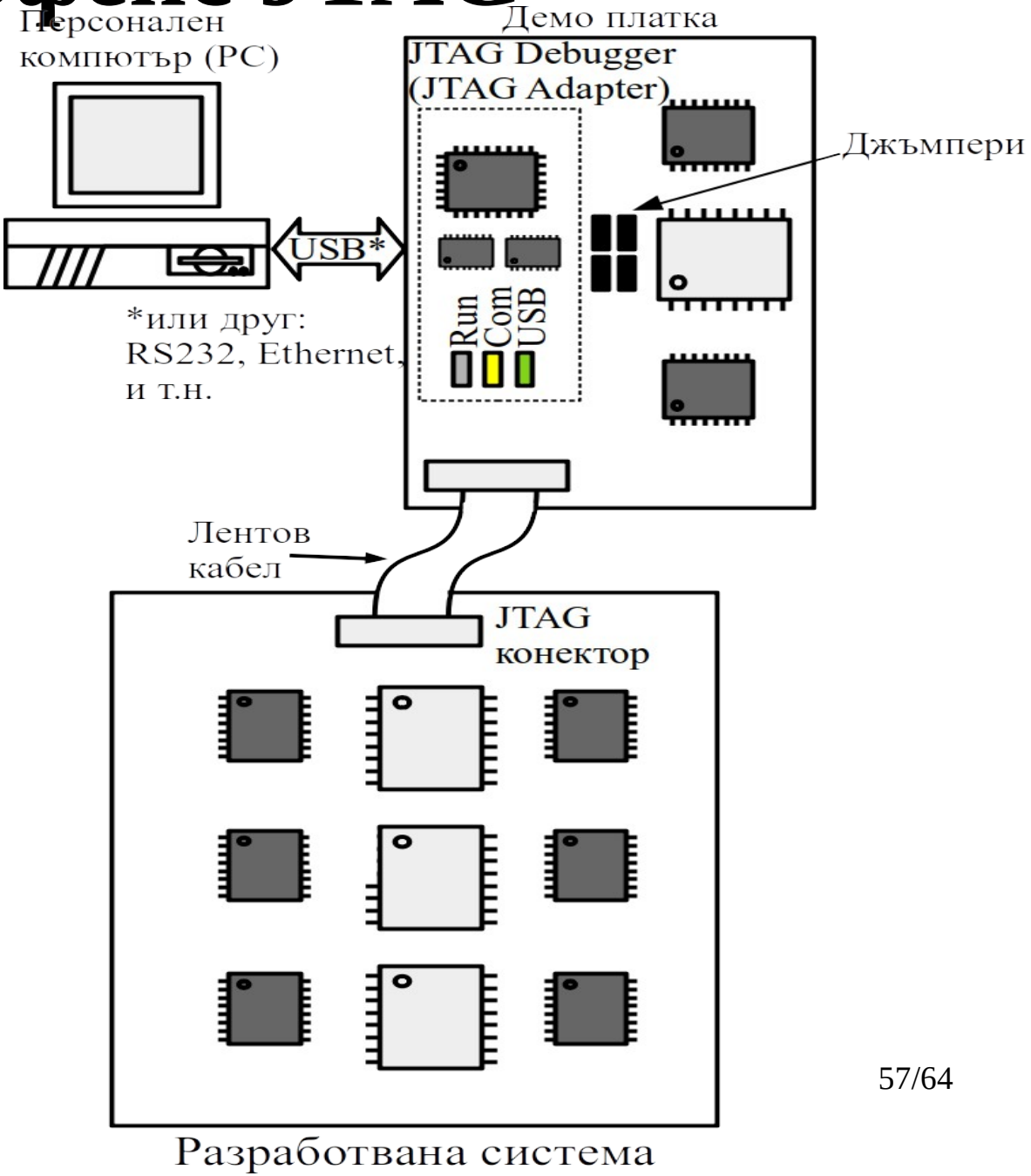
JTAG дебъгерите се предлагат като вградени уреди на демо платки. Това най-често означава, че:

*може да програмира **няколко μCU от една фамилия**, на **една фирма производител**;

*може да тества с **ниски скорости** (0.1 ÷ 4 MHz);

*струва **5 ÷ 30 $**.

*понякога има JTAG куплунг, към който може да се свърже разработваната система.



Интерфейс JTAG

Дебъгване на програма – процес на откриване и отстраняване на грешки в програмата.

За дебъгване на вградена система са необходими:

*JTAG адаптер, който да направи конвертирането “стандартен PC интерфейс ↔ JTAG интерфейс”

*софтуерен дебъгер (GDB) – програма, която да прочете и зареди обектовия код в паметта на системата и да задава команди на микропроцесора през JTAG-а.

Интерфейс JTAG

Командите, които μPU изпълнява под управлението на софтуерен дебъгер, са:

*(**program**) зареждане на фърмуера в паметта на μCU (системата);

*(**halt**) временно спиране изпълнението на фърмуера при поискване;

*(**run**) пускане на безкрайното изпълнение на фърмуера (така, че все едно няма дебъгер свързан);

*(**run to line**) временно спиране изпълнението на фърмуера при достигане на предварително зададена точка (адрес) от програмата;

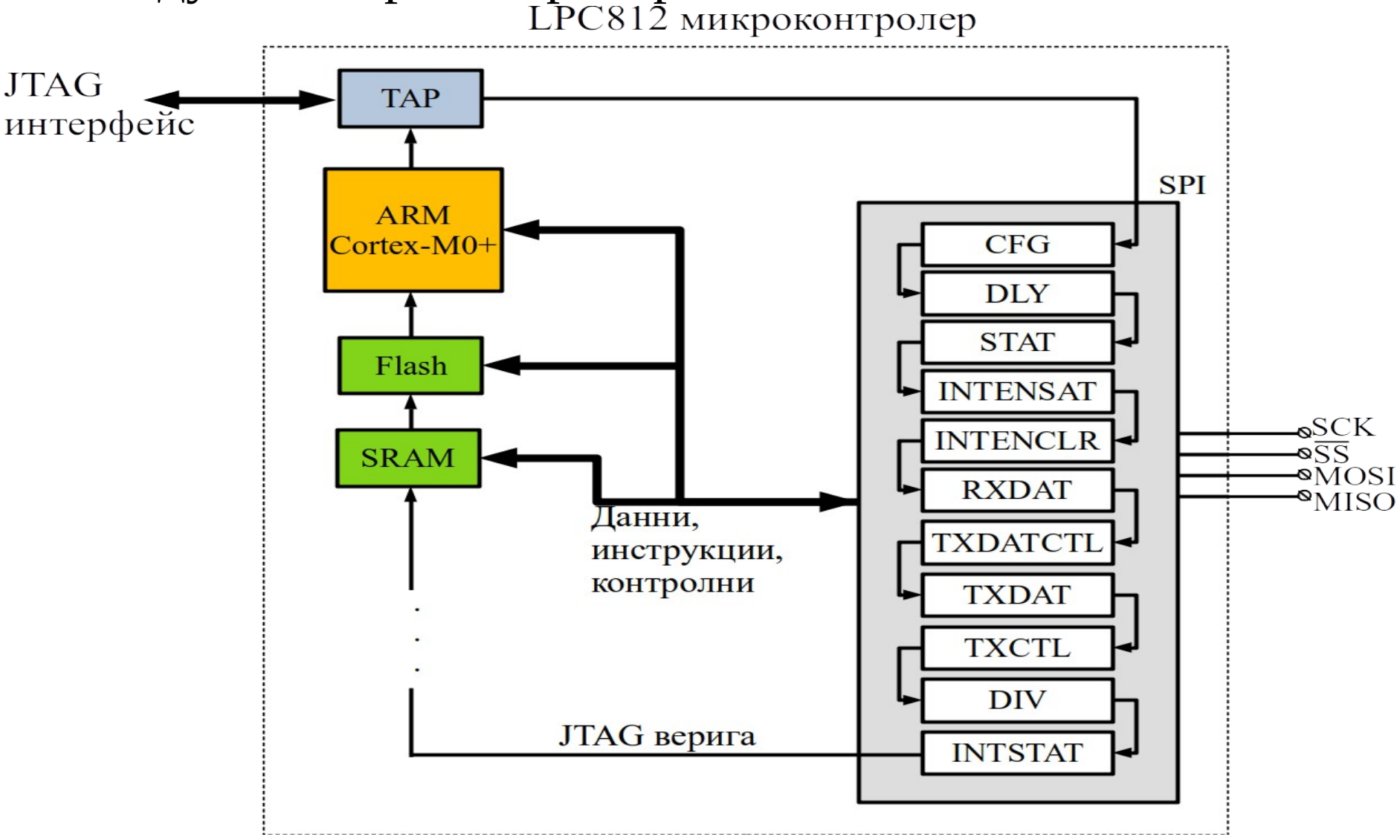
*(**single step**) изпълнение на фърмуера ред по ред;

*(**watch**) наблюдаване регистрите на памет и периферия;

*други.

Интерфейс JTAG

Пример – използване на JTAG за наблюдение на регистрите на SPI модул в микроконтролера LPC812.



Интерфейс SWD

SWD интерфейс (**S**erial **W**ire **D**ebug) – алтернативен интерфейс за програмиране и дебъгване, създаден от фирмата ARM. Изисква по-малко на брой сигнали от JTAG.
*Да не се бърка със SBW (**S**py-**B**i-**W**ire) на Texas Instruments.*

Използва следните сигнали:
***SWCLK** (**S**erial **W**ire **C**lock) – тактов сигнал
***SWDIO** (**S**erial **W**ire **D**ata **I**nput / **O**utput) – входно/изходен сигнал за данни

Незадължителни:
*SWO (Serial Wire Output) – за изпращане на printf съобщения от дебъгвания μCU, през SWD дебъгера, до персоналния компютър
*TRACEDATA0 ÷ 3 – 4-битов паралелен интерфейс за трасиране на програми [9]
*TRACECLK – тактов сигнал за трасиращия интерфейс

Интерфейс SWD

Трасиране на програми – извличане на инструкциите, изпълнявани от микропроцесора и изпращането им по широколентов интерфейс без да се прекъсва програмата.

Това позволява да се направи лог от инструкции, които описват много точно историята на програмата и така може да се изчистят много сложни грешки (бъгове).

Дебъгери с възможност за трасиране струват от порядъка на 700 ÷ 10000 $.

*

Интерфейс SWD

*SWD може да работи до 4 MHz, а трасиращият порт [8]:

Микропроцесор	Трасиращ порт (TPIU)
1 x ARM Cortex-M4 @ 200 MHz	4-бита @ 40 MHz DDR
2 x ARM Cortex-A55 @ 1333 MHz	16-бита @ 105 MHz DDR
4 x ARM Cortex-A55 @ 1333 MHz	16-бита @ 210 MHz DDR

*SWD позволява трасиране с помощта на няколко допълнителни сигнала.

*Периодично може да се наблюдават регистри, без да се спира изпълнението на програмата (watchpoint).

*Позволява да се пускат броячи на събития.

*Позволява до 8 хардуерни точки на прекъсване в програмата.

*Позволява да се използват микротрасиращи буфери.

Литература

[1]”NAND Flash 101: An Introduction to NAND Flash and How to Design It In Your Next Product”, Technical Note TN-29-09, Micron, 2006.
[2]”FRAM – New Generation of Non-Volatile Memory”, SZZT014A, 2009.
[3] M48Z02 Zeropower SRAM, Datasheet, ST Microelectronics, 2003.
[4] MT29F4G08FABWP Datasheet, Micron, 2004.
[5] Г. Михов, “Настройка и диагностика на микропроцесорни системи”, ТУ-София, 2005.
[6] “IEEE Standard 1149.1 (JTAG) in the SX/RTSX/SXA/eX/RT54SX-S Families”, Application Note AC160, Microsemi, 2012.
[7]”SN74BCT8244A Scan Test Devices with Octal Buffers”, Datasheet, Texas Instruments, 1996.
[8]”Trace port type and maximum width and data rate”, Article ID KA001391, ARM Ltd, 2021.
[9] “CoreSight 20 connector”, online, 2021.
https://developer.arm.com/documentation/100893/0100/Target-interface-connectors/CoreSight-20-connector