Таймерни модули

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

Въведение в таймерните модули

Във вградените системи почти винаги се налага генерирането и измерването на времеви интервали.

Пример - системата трябва да измерва периодично с АЦП-то си изхода на датчик и да изпраща данните към базова станция.

Пример – трябва да генерира правоъгълни импулси с определен коефициент на запълване и период, които да се използват за управление на различни изпълнителни устройсва, например електродвигатели.

Въведение в таймерните модули

Пример - измерване параметрите на импулси → някои датчици и периферни схеми генерират честота или времетраене на импулс, пропорционално на измерваната величина.

Ултразвуковият датчик HC-SR04 генерира правоъгълни импулси с времетраене, пропорционално на измереното разстояние до обект.

Инфрачервеният датчик MLX90614 генерира правоъгълни импулси, чиито период и коефициент на запълване са пропорционални на безконтактно измерената температура на обект.

Въведение в таймерните модули

Таймерен модул – хардуерен блок от микроконтролера, с помощта на който:

----------*измерва*----------

*се измерват честота/период и коефициент на запълване на правоъгълни импулси от външни схеми/датчици;

*регистрира се броят на постъпили импулси от външни схеми/датчици;

----------*генерира*----------

*генерират се правоъгълни импулси с определена честота/период и коефициент на запълване за управление на външни схеми/актуатори;

*генерират се времеинтервали за спомагане работата на фърмуера на микроконтролера.

Въведение в таймерните модули

За генерирането на времеви интервали има два подхода:
*да се реализира **софтуерно**, чрез цикъл;
*да се реализира **хардуерно** – чрез таймер.

Софтуерно:

```
        mov   #100, r5 ;Зареди 100 в r5, 2-такта
L1:     sub   #1, r5    ;Намали r5 с едно, 2-такта
        jnz   L1        ;Сравни и се върни в L1, ако е ненулев
                        ;резултат, 2-такта
```

Въведение в таймерните модули

Ако микропроцесорът работи с тактова честота 8 MHz, то периода на един такт ще е T = 1 / 8 MHz = 125 ns.

От горната програма може да изчислим колко време ще отнеме цялото изпълнение на цикъла:

t_{delay} = [2 + 100*(2+2)]*125 ns = 50250 ns = 50.25 μs.

Чрез промяна на първия ред можем да променим това време зануждите на приложението.

Въведение в таймерните модули

```
volatile uint8_t i;
for(i = 0; i < 100 ; i++){ }
```

и ако знаем времето, за което се изпълнява една итерация на цикъла, може да изчислим закъснението на цялата for-конструкция.

Предимства – софтуерното генериране на времеви интервали не изисква специални хардуерни модули.

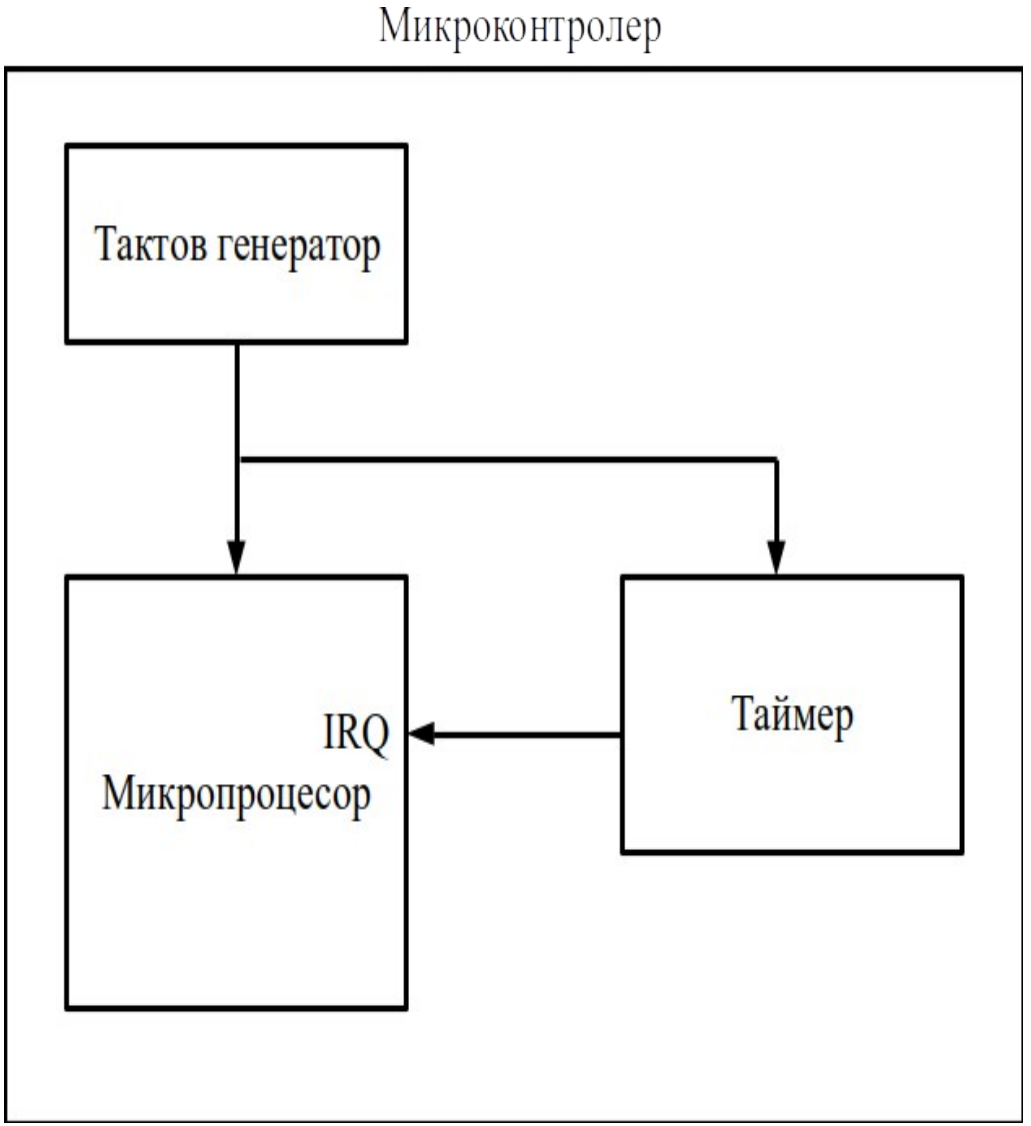
Недостатъци – докато микропроцесорът изпълнява кода на цикъла, **нищо друго не може да извършва**.

Конвейерното изпълнение на инструкцията и зануляването на конвейера **затрудняват изчислението на времето**.

Въведение в таймерните модули

Хардуерно генериране на закъснение – може да се извърши със таймерен модул, вграден в микроконтролера. Неговата цел е да отмери определен брой тактови импулси и да сигнализира на процесора, когато е достигнат максимума.

Предимства – докато този отделен модул отброява, микропроцесорът може да продължи изпълнението на програмата си.



Въведение в таймерните модули

Броячи – цифрови последователностни логически схеми, с паралелен изход, при който всеки разред се установява в логическо състояние, така че да отговаря на числото, съответстващо на изброените импулси,
постъпили на входа [1]. Разредността на брояча зависи от броя на изходните му сигнали, т.е. на максималното число, което може да бъде изведено на тях.

4-битови броячи – максималното число, до което могат да броят е $2^4 - 1 = 15$

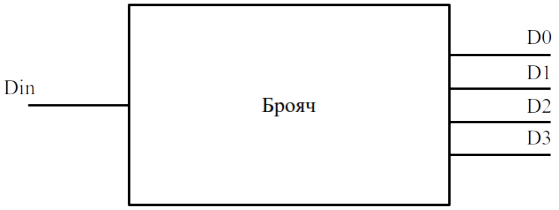
8-битови броячи – максималното число, до което могат да броят е $2^8 - 1 = 255$

16-битови броячи – максималното число, до което могат да броят е $2^{16} - 1 = 65535$

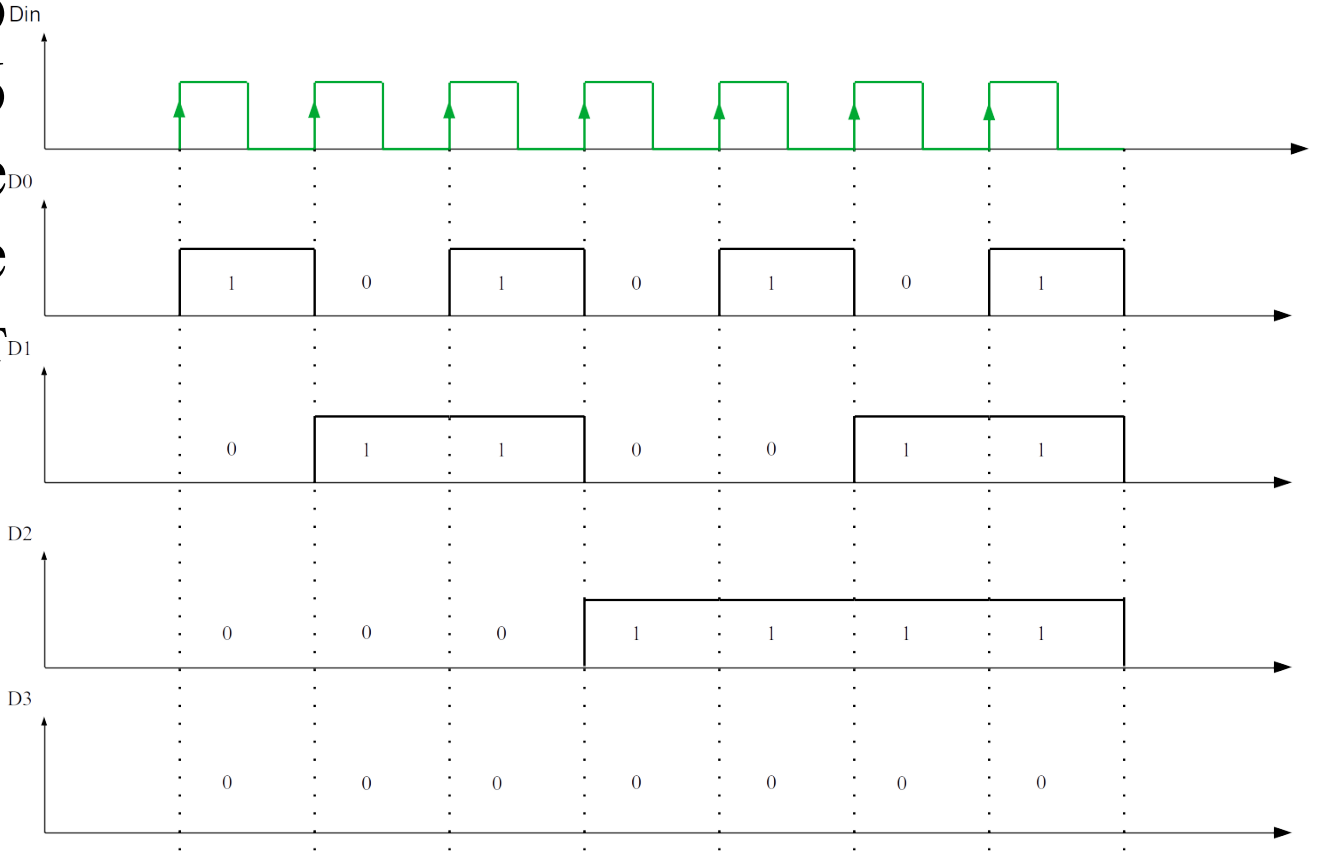
32-битови броячи – максималното число, до което могат да броят е $2^{32} - 1 = 4\ 294\ 967\ 295$

Въведение в таймерните модули

Пример – 4-битов брояч. За по-лесно изобразяване е показано броенето на 7 импулса. На 16 импулс броячът ще се препълни и ще започне да брои от 0.



Din	D3	D2	D1	D0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1



Въведение в таймерните модули

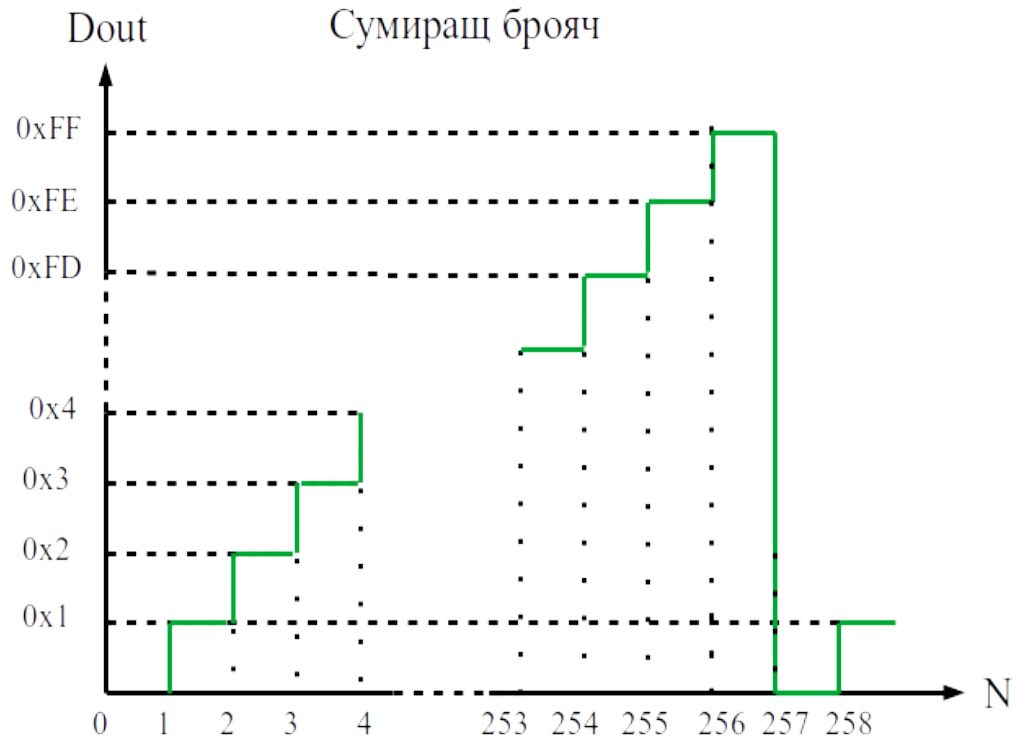
Ако един брояч брои от 0 към максималното си число, той се нарича **сумиращ** (up counter). При препълване броячът се установява в 0.

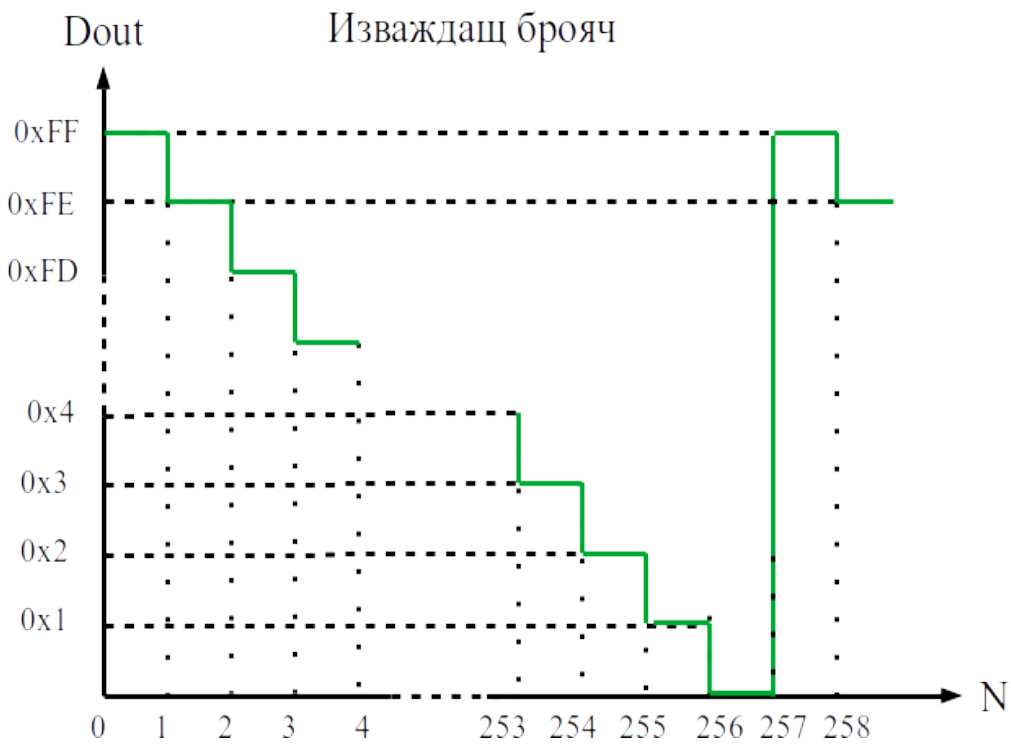
Ако един брояч брои от максималното си число към 0, той се нарича **изваждащ** (down counter). При препълване броячът се установява в максималното си число.

Ако един брояч брои от 0 към максималното си число и след това при препълване започне да брои от максималното си число до 0, той се нарича **реверсивен** (up/down counter).

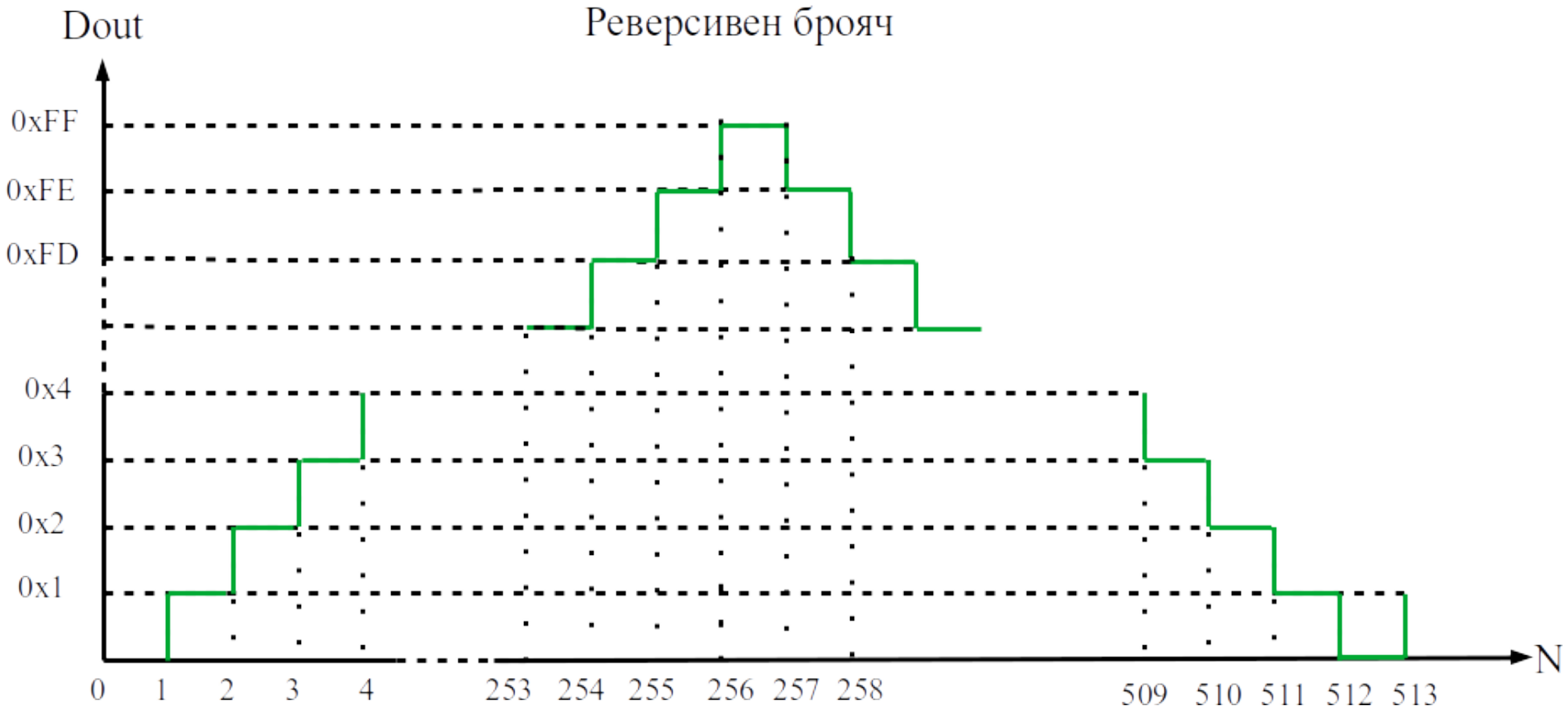
На фигурите на следващия слайд е демонстрирана работата на 8-битови броячи.

Въведение в таймерните модули





Въведение в таймерните модули



Въведение в таймерните модули

Делителите на честота не се различават съществено от броячите. Това са едни и същи структури, като двете понятия се използват в зависимост от приложението им [1].

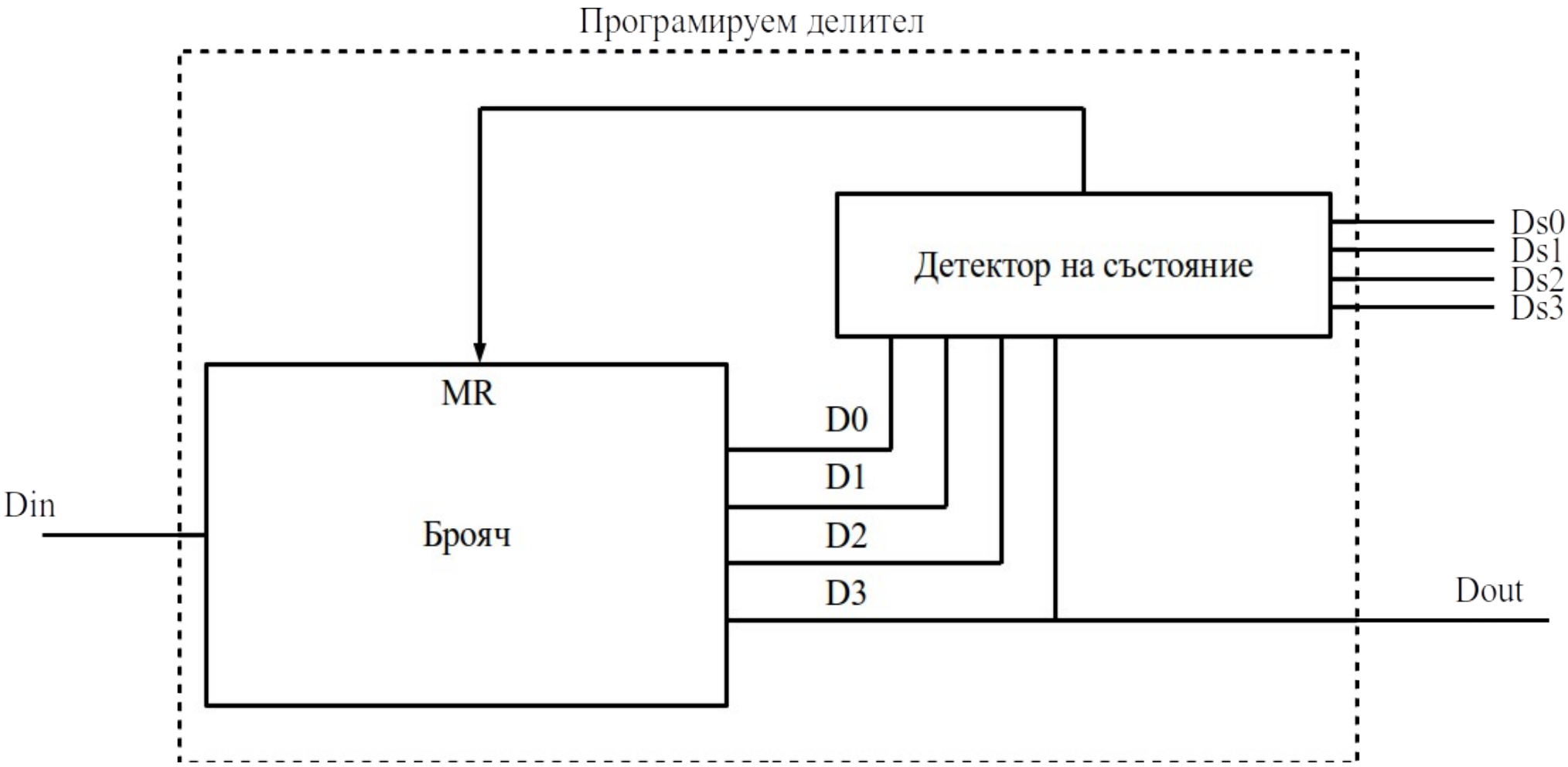
Делителите на честота притежават само един изход и при тях е от значение коефициента на делене, който осигуряват и коефициента на запълване на изходните импулси.

Въведение в таймерните модули

На следващия слайд е представена блокова схема на програмируем делител на честота. **Детекторът на състояние** е свързан последователно на изходите от брояча. На входовете Ds0 ÷ Ds3 се задава числото, до което броячът трябва да брои. Когато това число бъде достигнато, детекторът рестартира брояча чрез входа Master Reset (MR).

Програмируемите делители намират широко приложение в микроконтролерите, тъй като от една фиксирана системна честота могат да се получат други по-ниски такива. Това позволява повече възможности за конфигуриране на различните периферии.

Въведение в таймерните модули



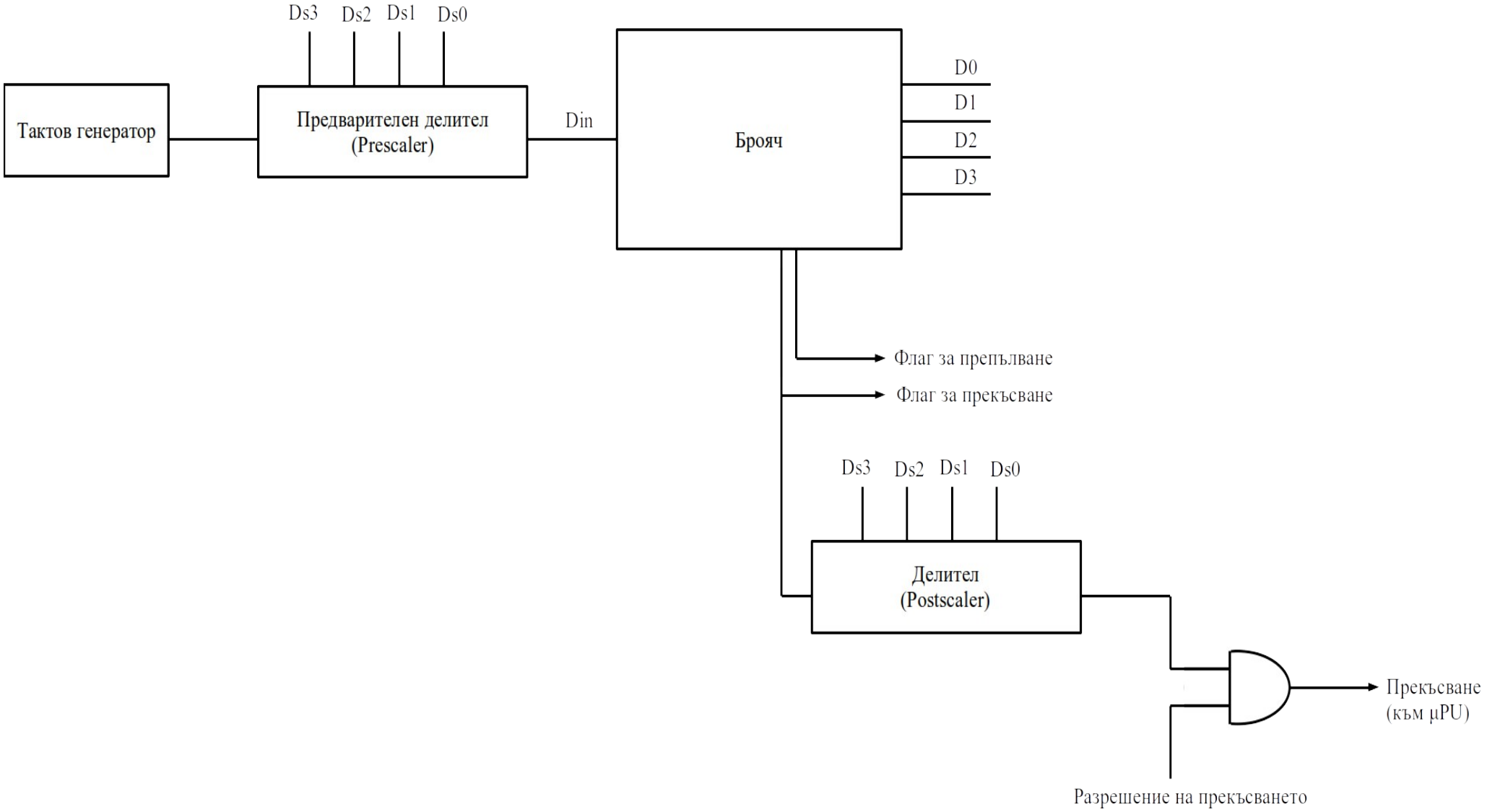
Въведение в таймерните модули

Понякога делителите се използват не само за понижаване на входната честота (наричат се предделител, prescaler), но и за понижаване на изходната (наричат се постделител, postscaler).

На следващия слайд е демонстрирана работата на брояч с предделител и постделител.

Вижда се, че благодарение на постделителя може да се конфигурира прекъсване на всяко второ, трето, четвърто и т.н. прекъсване.

Въведение в таймерните модули



Режими на работа на таймерните модули

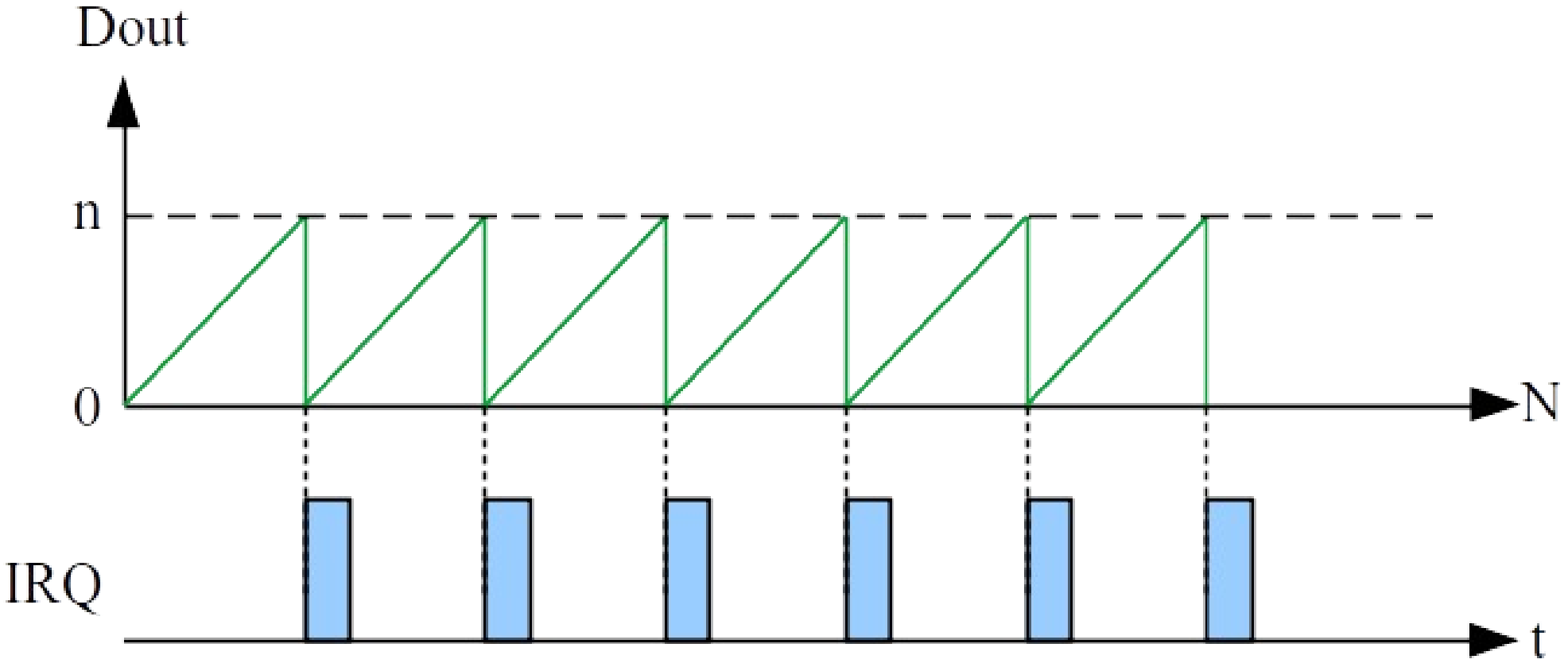
Таймерите се състоят от поне четири цифрови схеми:
*брояч
*цифров компаратор
*потребителски регистър
*детектор на фронт

Таймерите могат да работят в 4 режима:
***свободно броящ** (free running)
***измерващ** (Capture)
***генериращ** (Compare)
***ШИМ** (PWM)

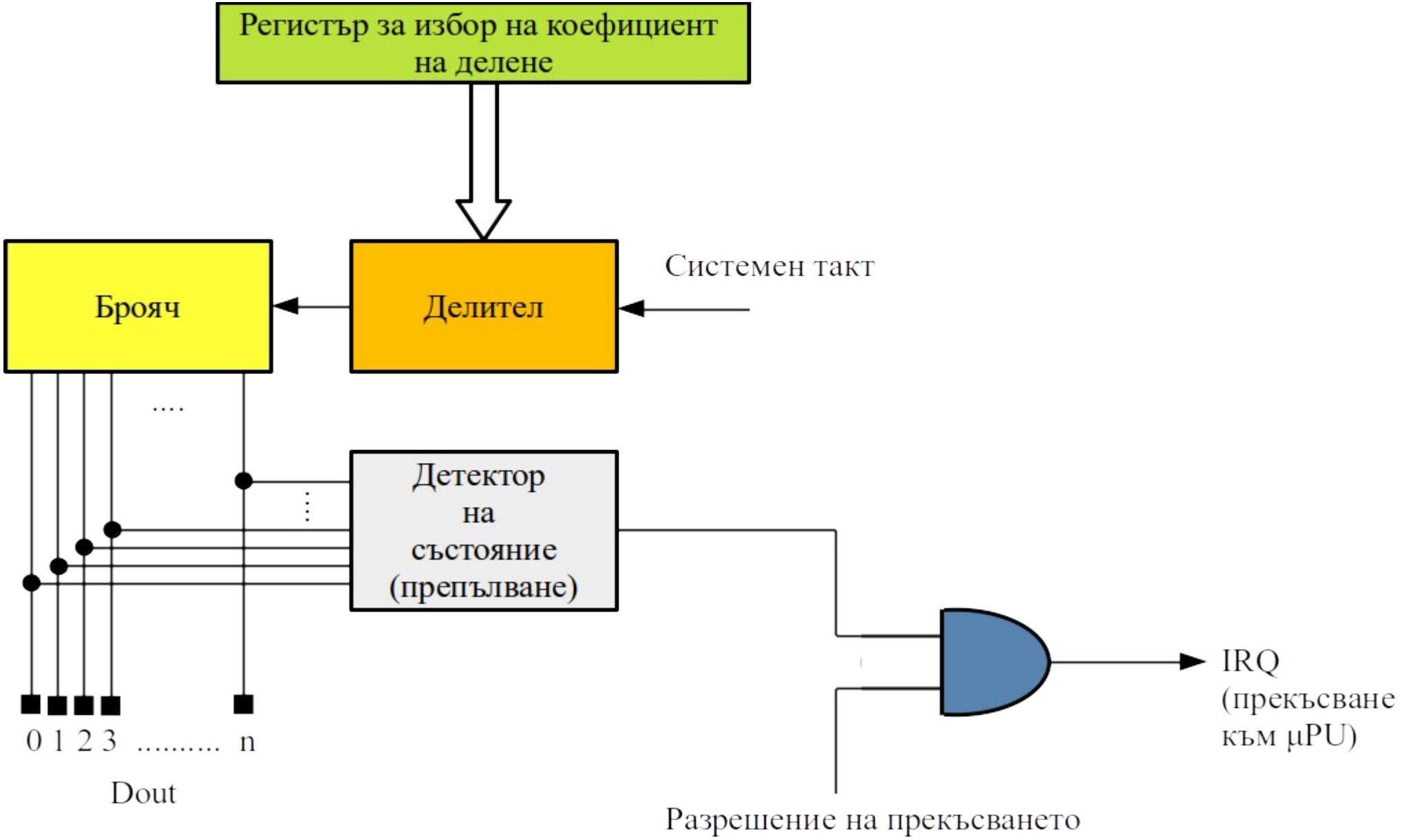
Capture/Compare/PWM = CCP

Режими на работа на таймерните модули

***свободно броящ** (free running) – таймерът брои от 0 до максималното си число (ако е сумиращ). При препълване генерира прекъсване.



Режими на работа на таймерните модули



Режими на работа на таймерните модули

***измерващ** (capture) – стойностите на брояча се извличат при настъпване на събитие. μPU запаметява тези стойности и след това ги обработва. В този режим може да се мери период/честота или коефициент на запълване.

Видове събития:

*постъпил нарастващ фронт на входа
*постъпил спадащ фронт на входа
*постъпил или нарастващ, или спадащ фронт

Режими на работа на таймерните модули

Препълване на брояча - повечето μCU имат таймери с възможност за детектиране на препълване (overflow) по време на измерващия режим. За това събитие се добавя бит в статус регистъра на таймера. При изчисляване на периода той трябва да бъде взет под внимание:

T_{in} - период на измервания сигнал

N_1 - стойност на брояча при първия фронт

N_2 - стойност на брояча при втория фронт

T_c - период на тактовия сигнал на брояча

n – максимално число, до което може да брои брояча

$$T_{in} = (N_2 - N_1).T_c \text{ (без препълване)}$$

$$T_{in} = [(n - N_1) + (N_2 + 1)].T_c \text{ (с препълване)}$$

Валидни за сумиращ брояч

Режими на работа на таймерните модули

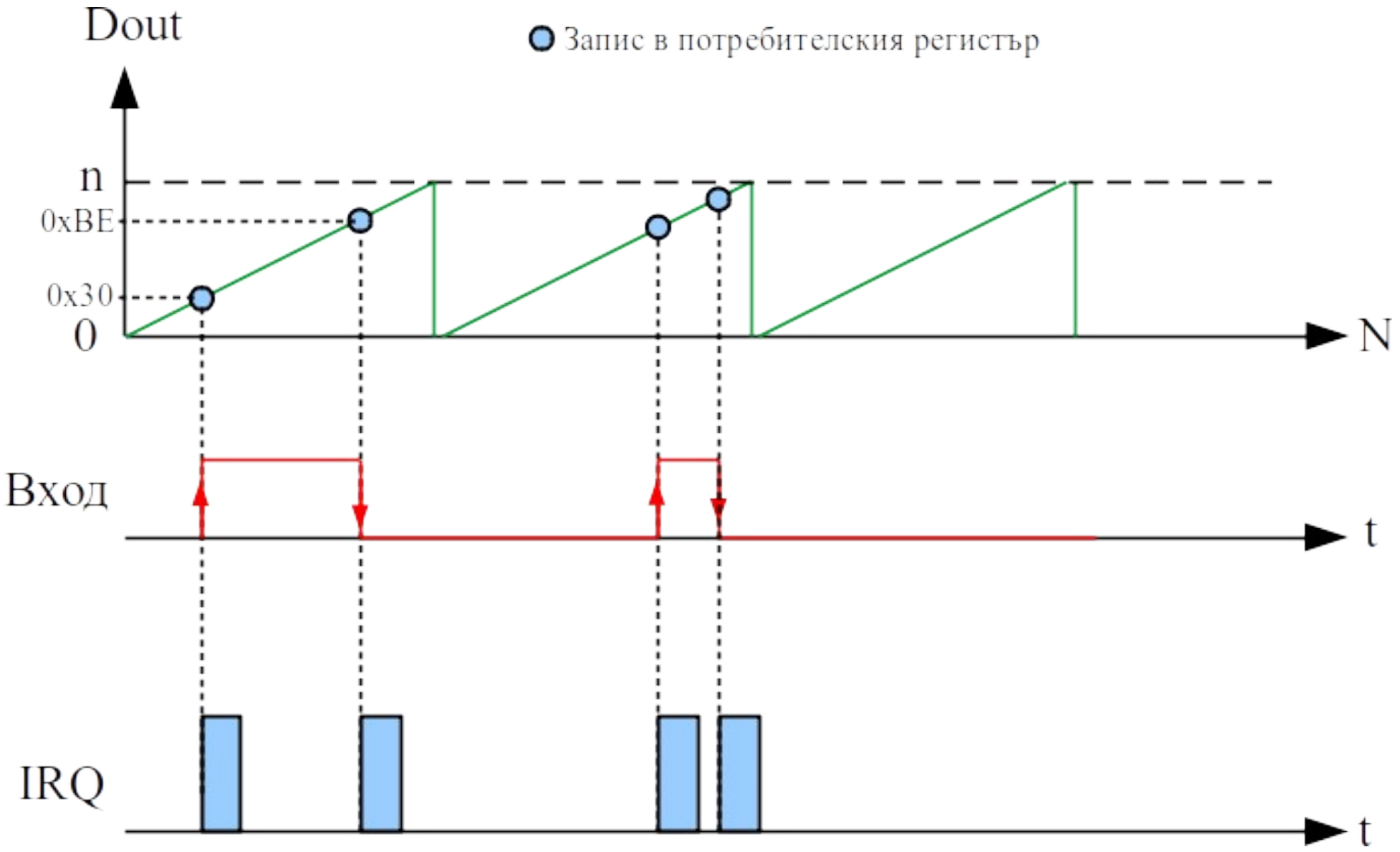
Пример - на извод на микроконтролера се подава правоъгълен сигнал.

Вътрешно изводът е свързан към детектор на фронт.

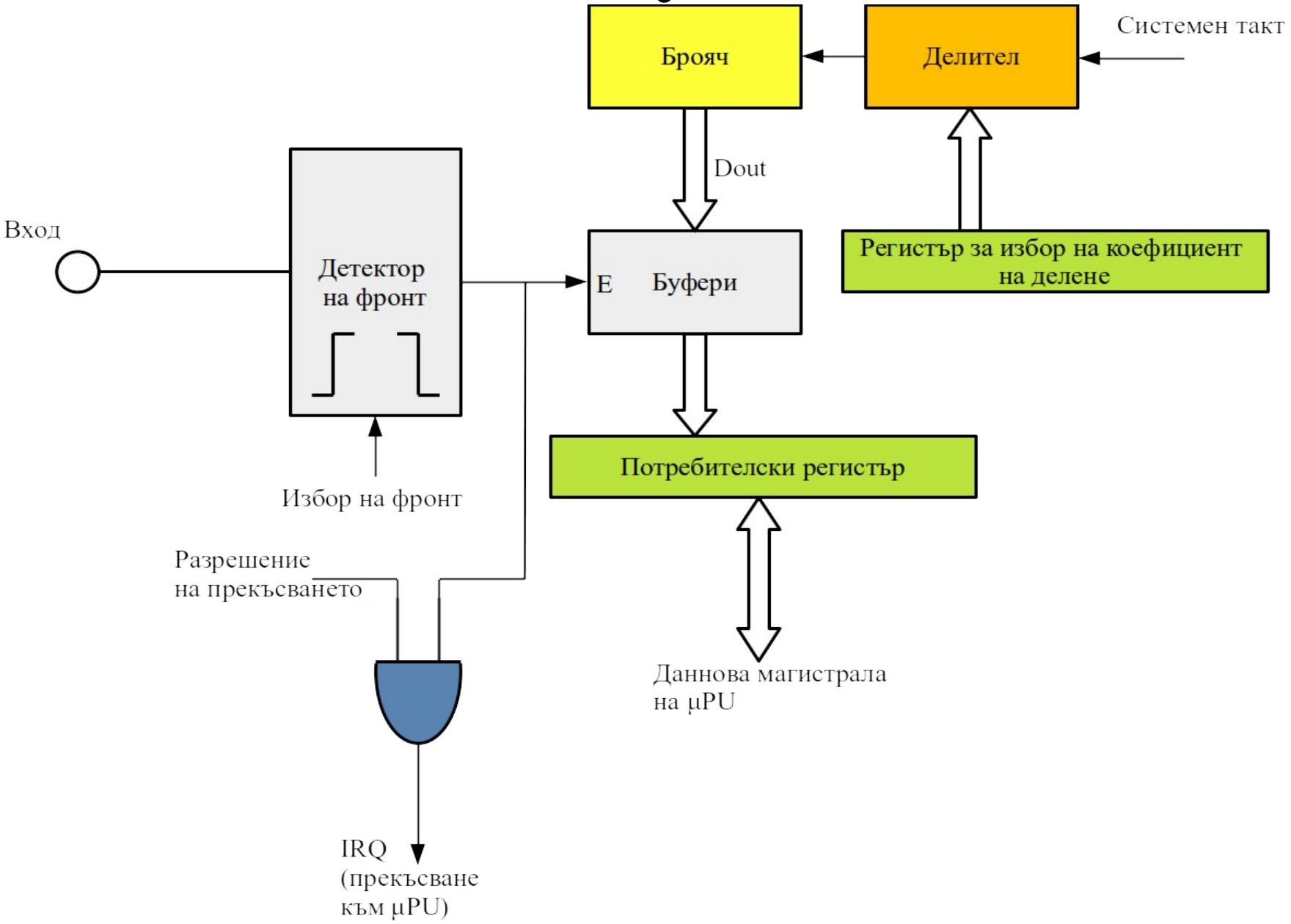
Когато настъпи събитие, микропроцесорът получава прекъсване и чете настоящата стойност на брояча.

Настоящата стойност на брояча е записана автоматично през един буферен блок в потребителския регистър.

Режими на работа на таймерните модули



Режими на работа на таймерните модули

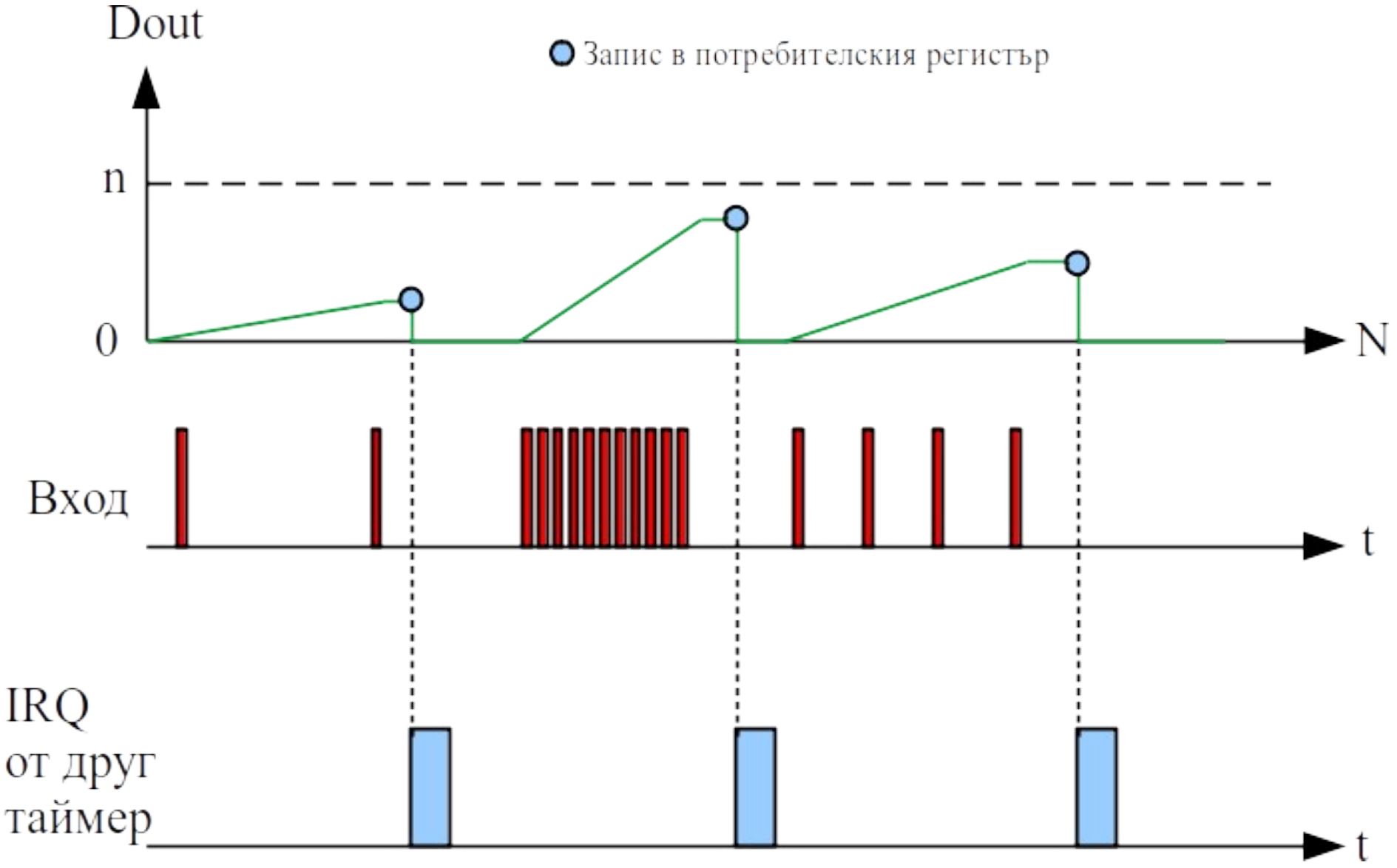


Режими на работа на таймерните модули

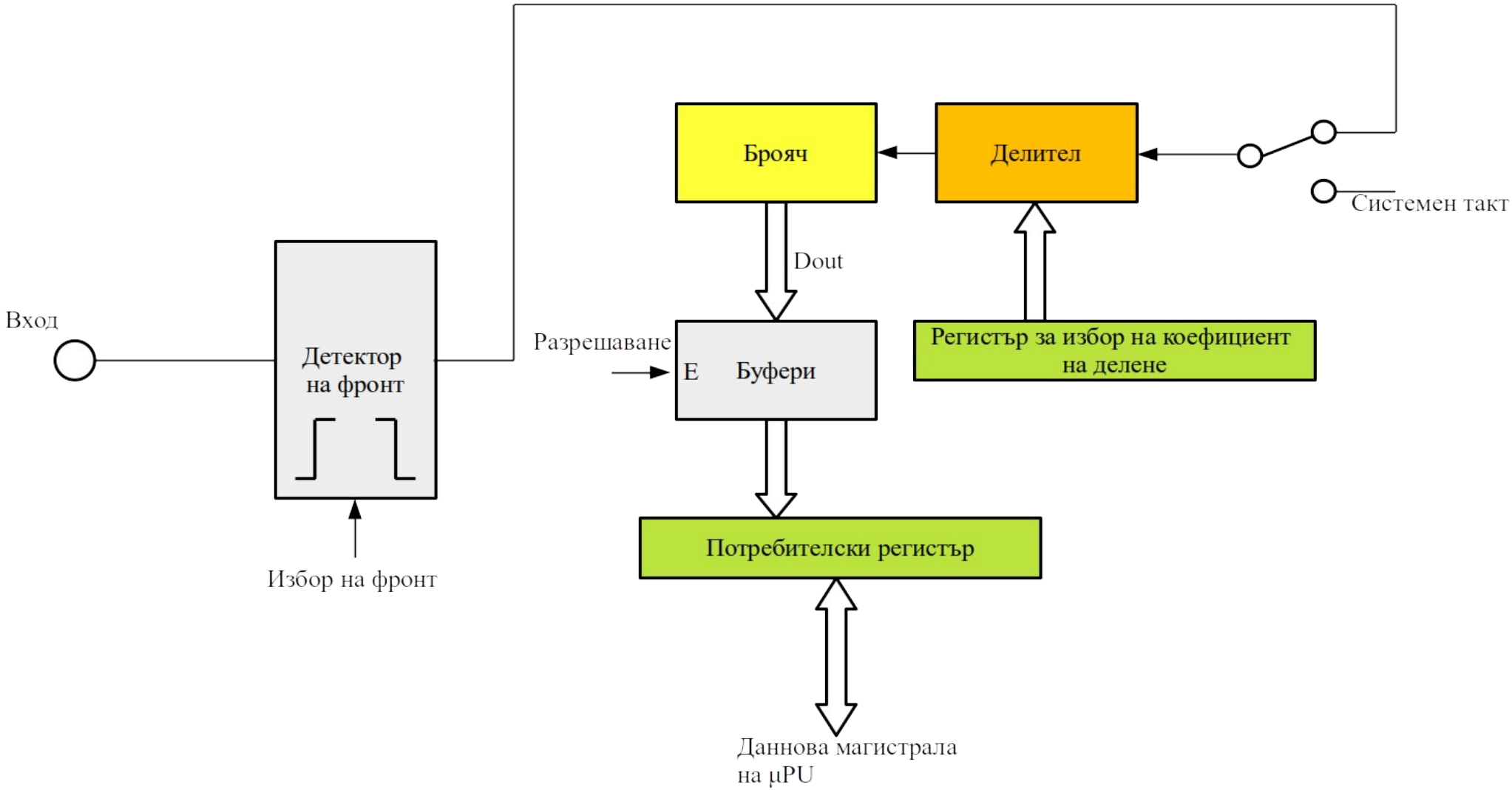
***измерващ** + режим броене на импулси – частен случай на измерващия режим, при който броячът се тактува от външен сигнал. Ако се пусне втори таймер, и периодично се чете стойността на брояча, може да се измери честотата на сигнала или да се броят импулси.

Пример – машина за навиване на трансформаторни намотки. Оборотомер на кола. Броячи на импулси и др.

Режими на работа на таймерните модули



Режими на работа на таймерните модули



Режими на работа на таймерните модули

***генериращ** (compare) – генерира се единичен правоъгълен импулс на извод на μCU. Стойността на брояча се сравнява със стойността на потребителския регистър и при съвпадение изводът може да се установи:

*в единица (set)

*в нула (clear)

*да се преобърне (toggle)

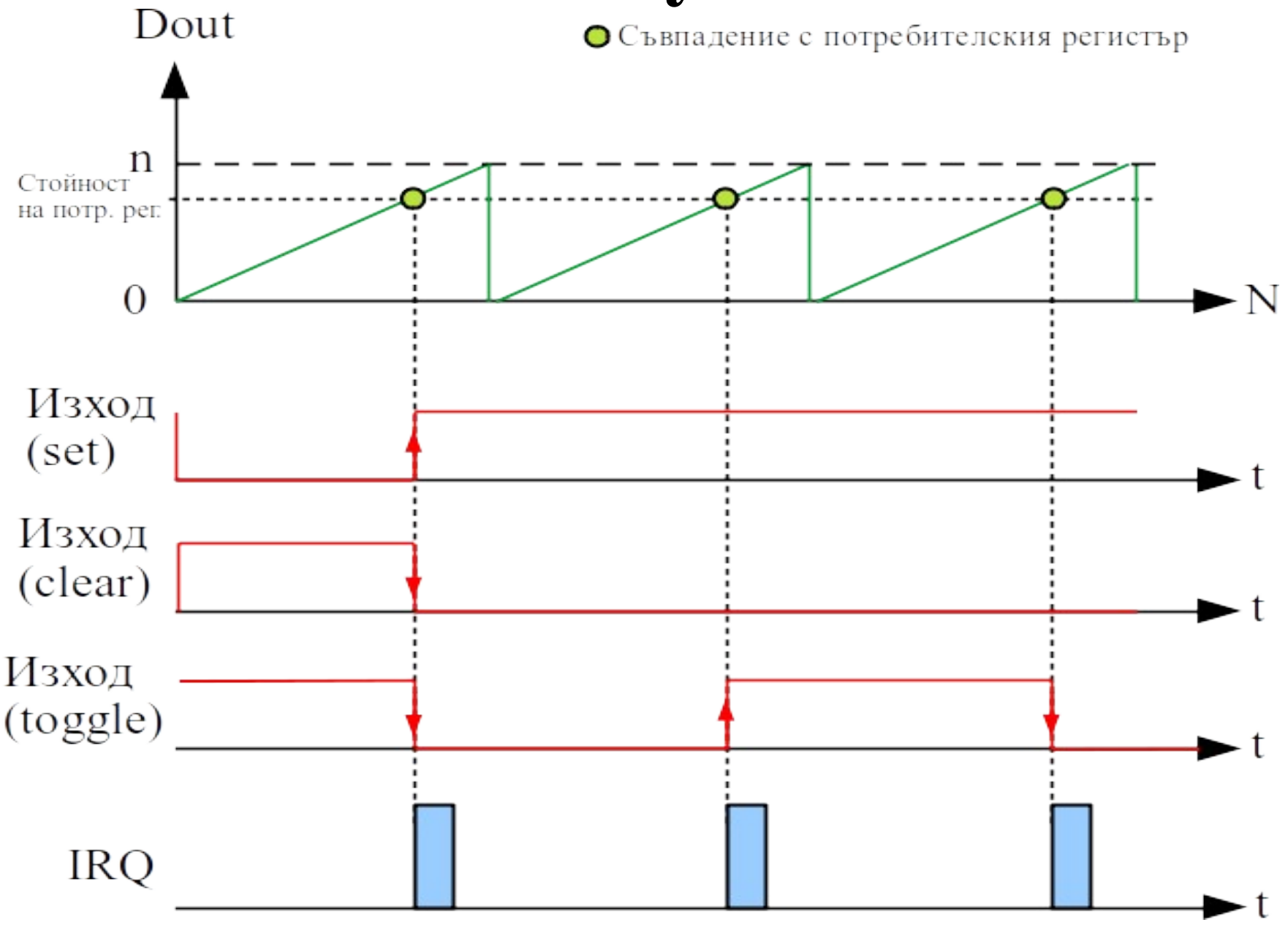
Когато е избрана функцията преобръщане, на извода се генерира ШИМ сигнал, чийто коефициент на запълване не може да се променя (винаги е 50%), нито честотата му (определя се от разредността на брояча).

Режими на работа на таймерните модули

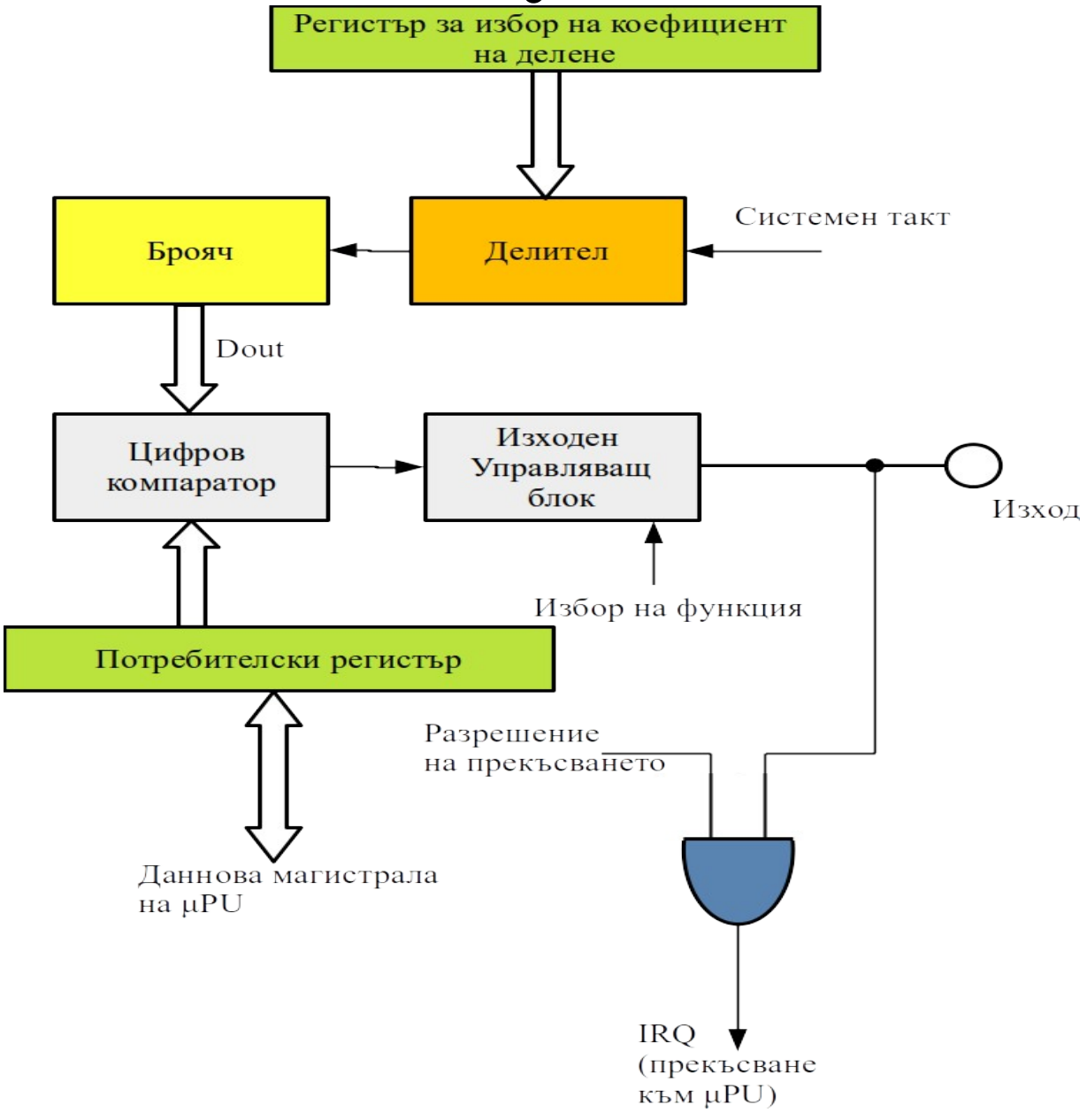
Когато са избрани функциите за установяване в нула или едно, таймерът може да се оприличи на моновибратор.

В генериращия режим сигналът за съвпадение може да не се извежда към извод на μCU. Тогава се **използва за генериране на прекъсване** към μPU като събитие за отмерване на изтекло време (**time out**). Последното е необходимо в много случаи на фърмуера.

Режими на работа на таймерните модули



Режими на работа на таймерните модули

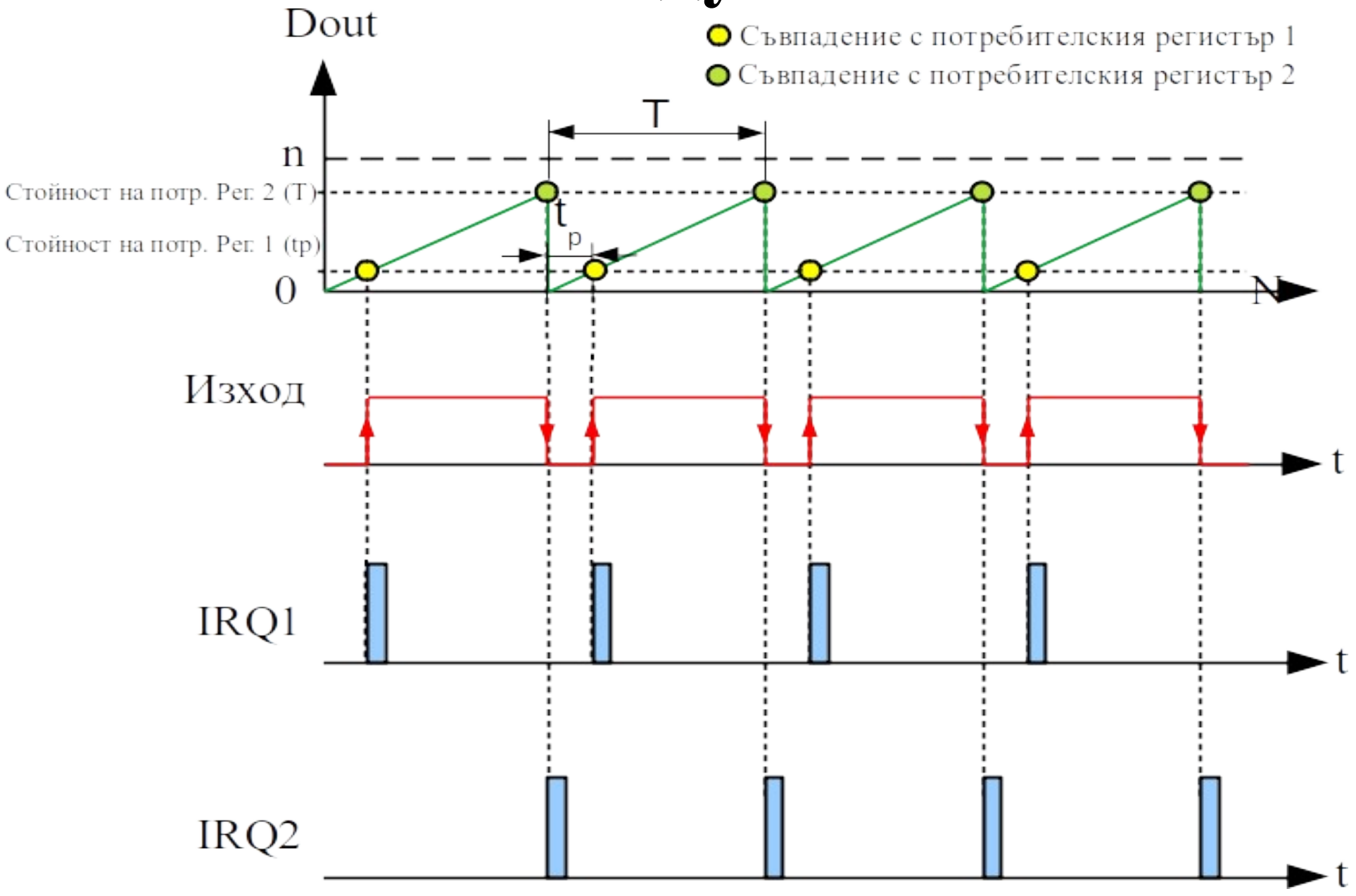


Режими на работа на таймерните модули

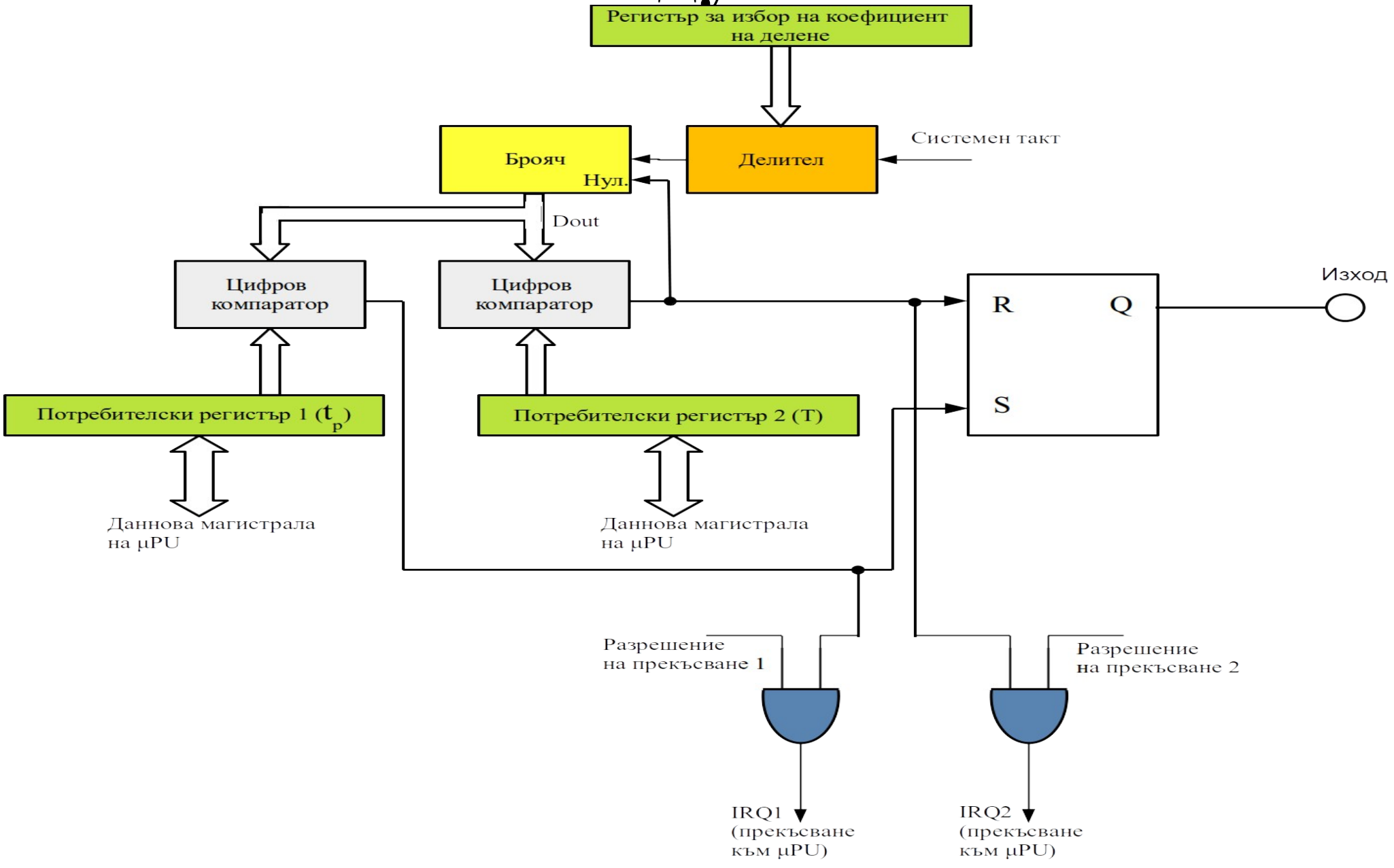
***ШИМ** – режим за генериране на сигнали с **ш**ирочинно-**и**мпулсна **м**одулация. Това е режим, аналогичен на генерирaщия, с разликата, че се използват два потребителски регистъра и два компаратора. Тази архитектура позволява да се променя както **коефициента на запълване** (t_p), така и **периода** на сигнала (T).

Изходите на компараторите се подават към двата входа на RS-тригер. Броячът се нулира при започването на всеки нов период (затова сигналът на входа S се подава на входа Нул.).

Режими на работа на таймерните модули



Режими на работа на таймерните модули



Режими на работа на таймерните модули

Ако броячът на даден PWM модул работи в сумиращ или изваждащ режим, то изходният сигнал може да се намира в началото или в края на периода на броене на брояча. Казва се, че се генерира **едностранно подравнена ШИМ.** Тя може да е**:**

***ляво подравнена** (left-aligned PWM)– ако цифровият компаратор сработва при стойности на брояча по-малки от стойността, записана в потребителския регистър;

***дясно подравнена** (right-aligned PWM)- ако цифровият компаратор сработва при стойности на брояча по-големи от стойността, записана в потребителския регистър.

Режими на работа на таймерните модули

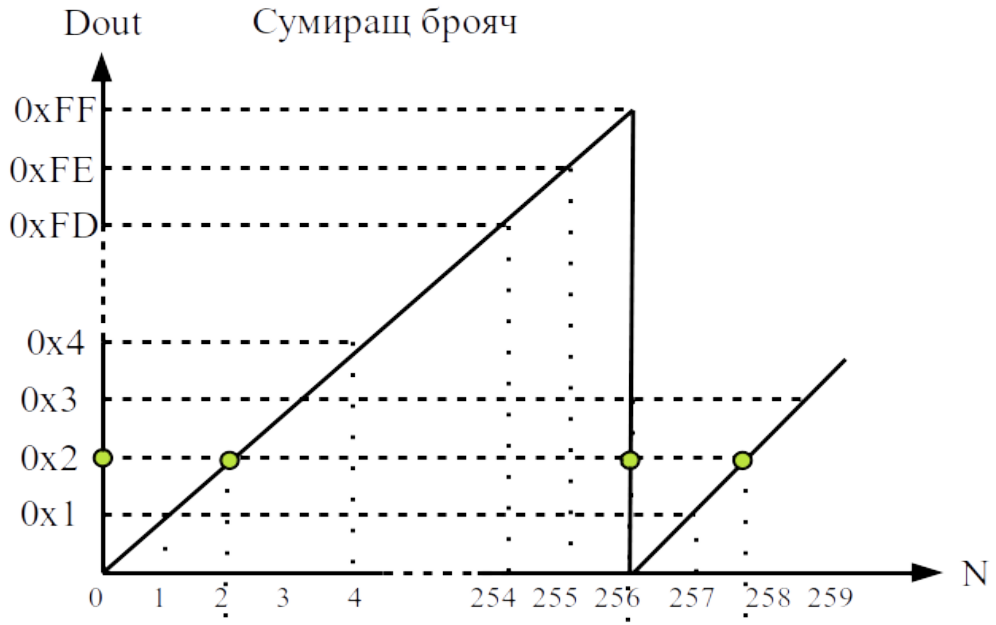
Ако броячът на даден PWM модул работи в реверсивен режим, то изходният сигнал се генерира със **средно подравнена** модулация (center-aligned PWM).

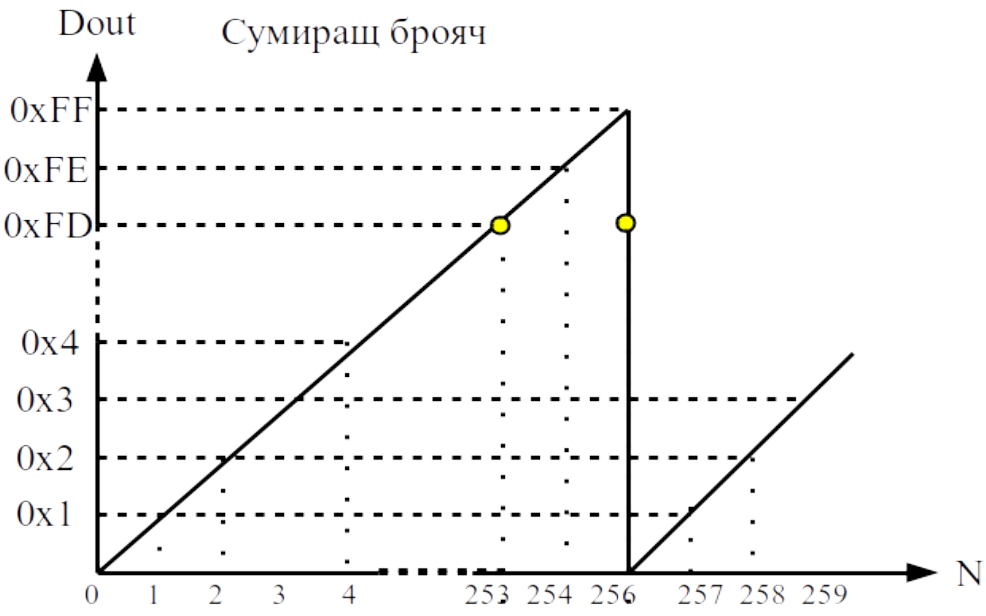
При нея, цифровият компаратор сработва при стойности на брояча по-големи от стойността, записана в потребителския регистър. Компараторът сработва в средата на периода на броене (оттам идва и наименованието на модулацията).

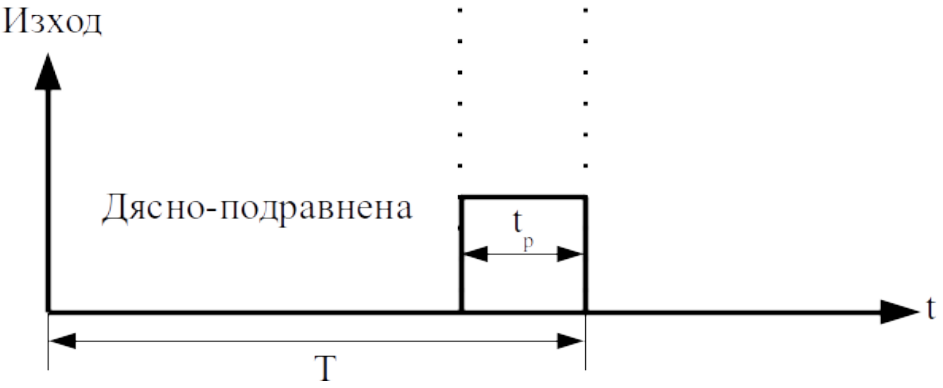
Режими на работа на таймерните модули

По-малко от потребителския регистър

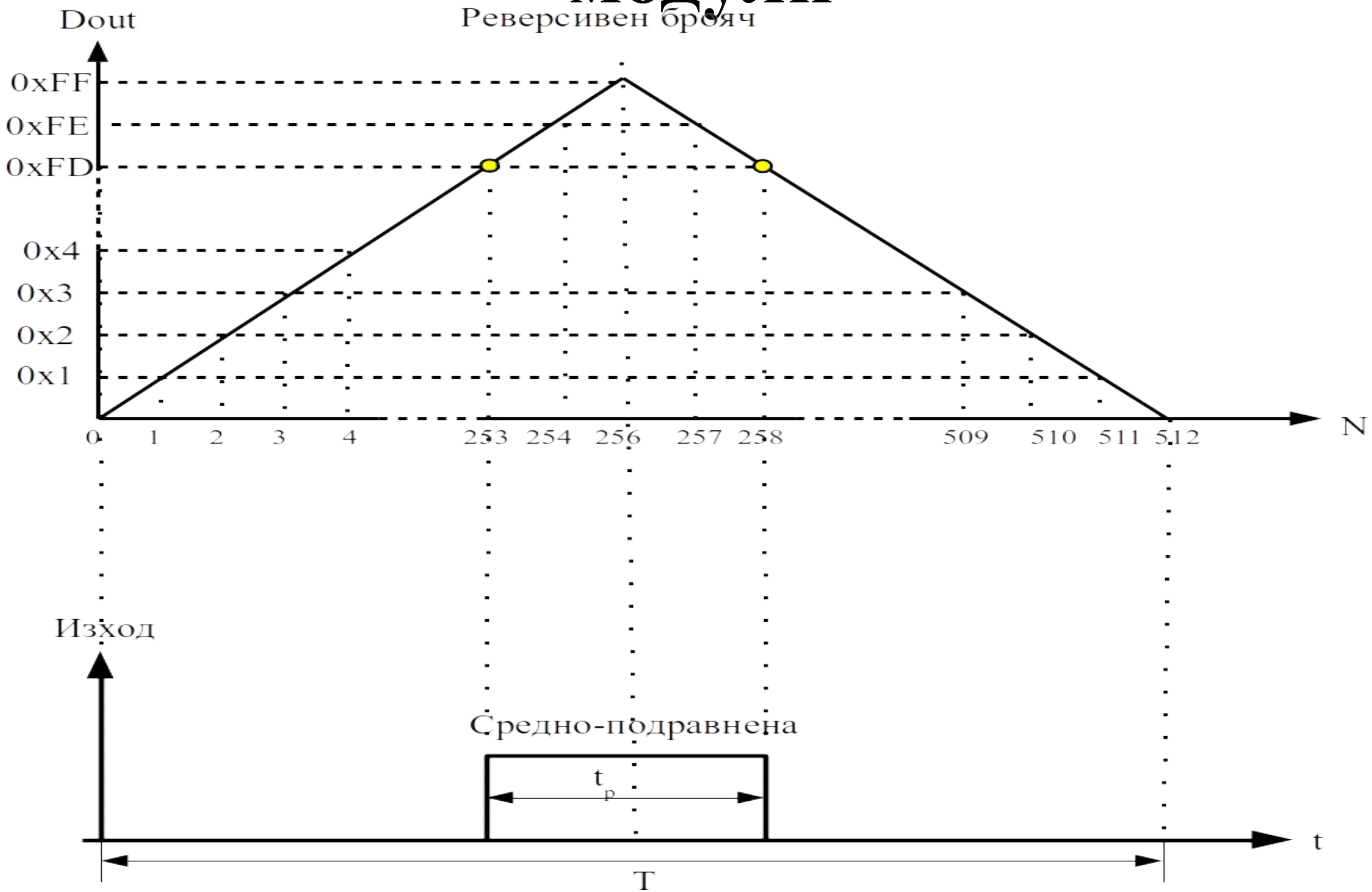
По-голямо от потребителския регистър







Режими на работа на таймерните модули



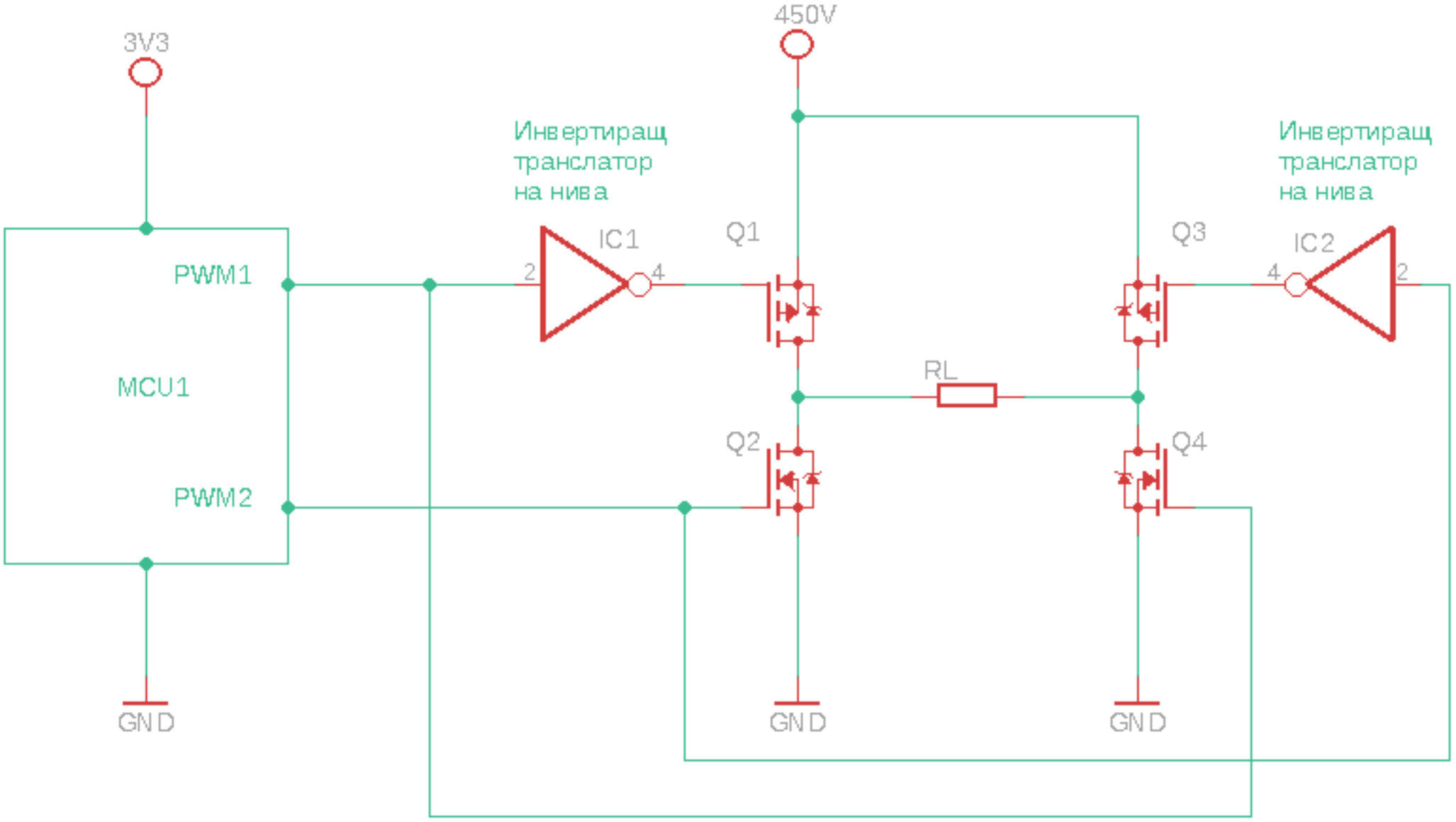
Режими на работа на таймерните модули

Подравняването на модулацията има значение при задаване на двойки управляващи сигнали в **силови схеми**.

В много от тях трябва да се въведе т.нар. **мъртво време**, което представлява разминаване на сигналите във времето, така че два (или повече) силови ключа никога да не бъдат включвани едновременно. Така се подобрява енергийната ефективност и се улеснява охлаждането.

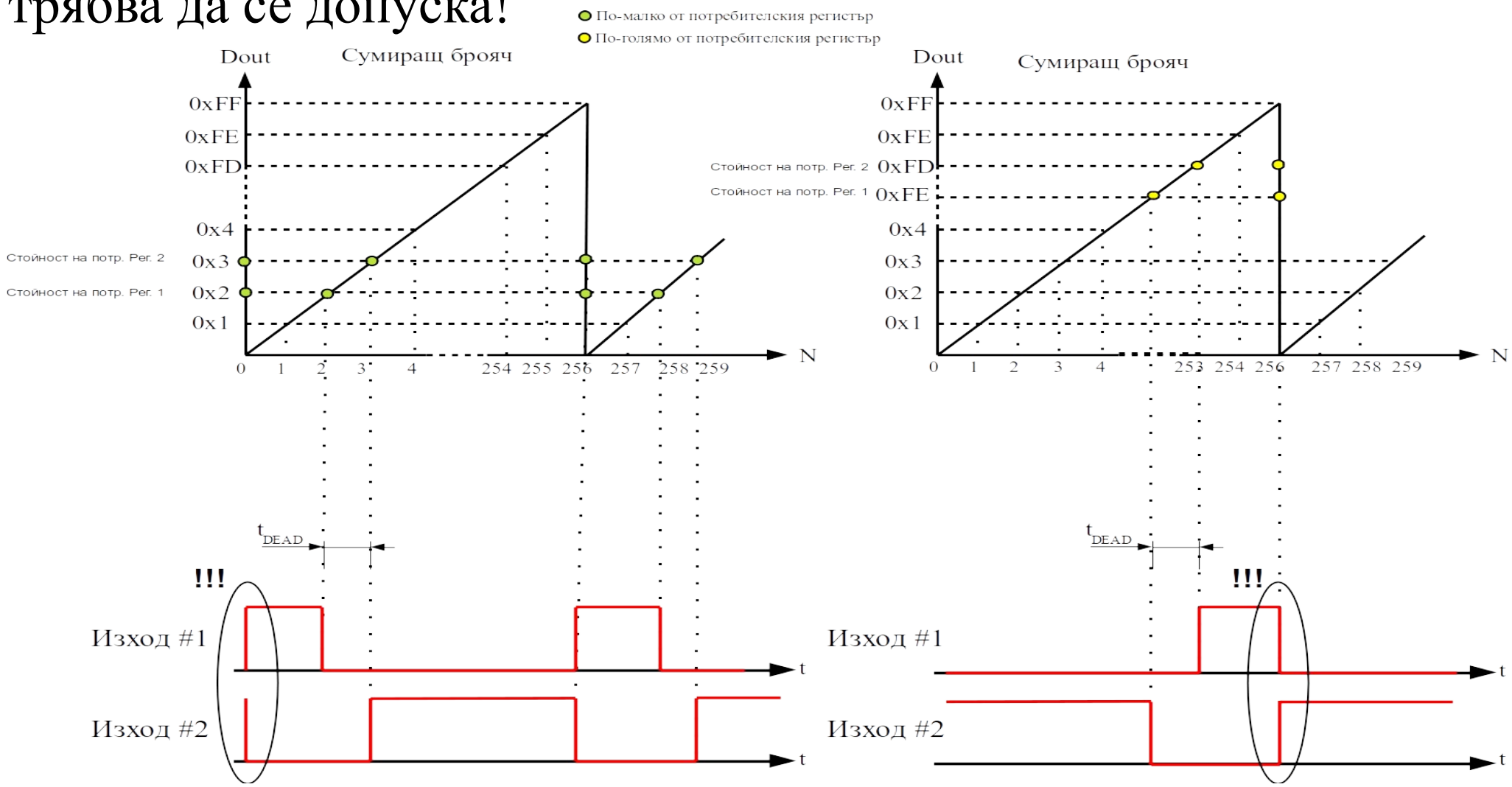
Мъртво време по два фронта може да се реализира **само** с брояч в **реверсивен режим**!

Режими на работа на таймерните модули

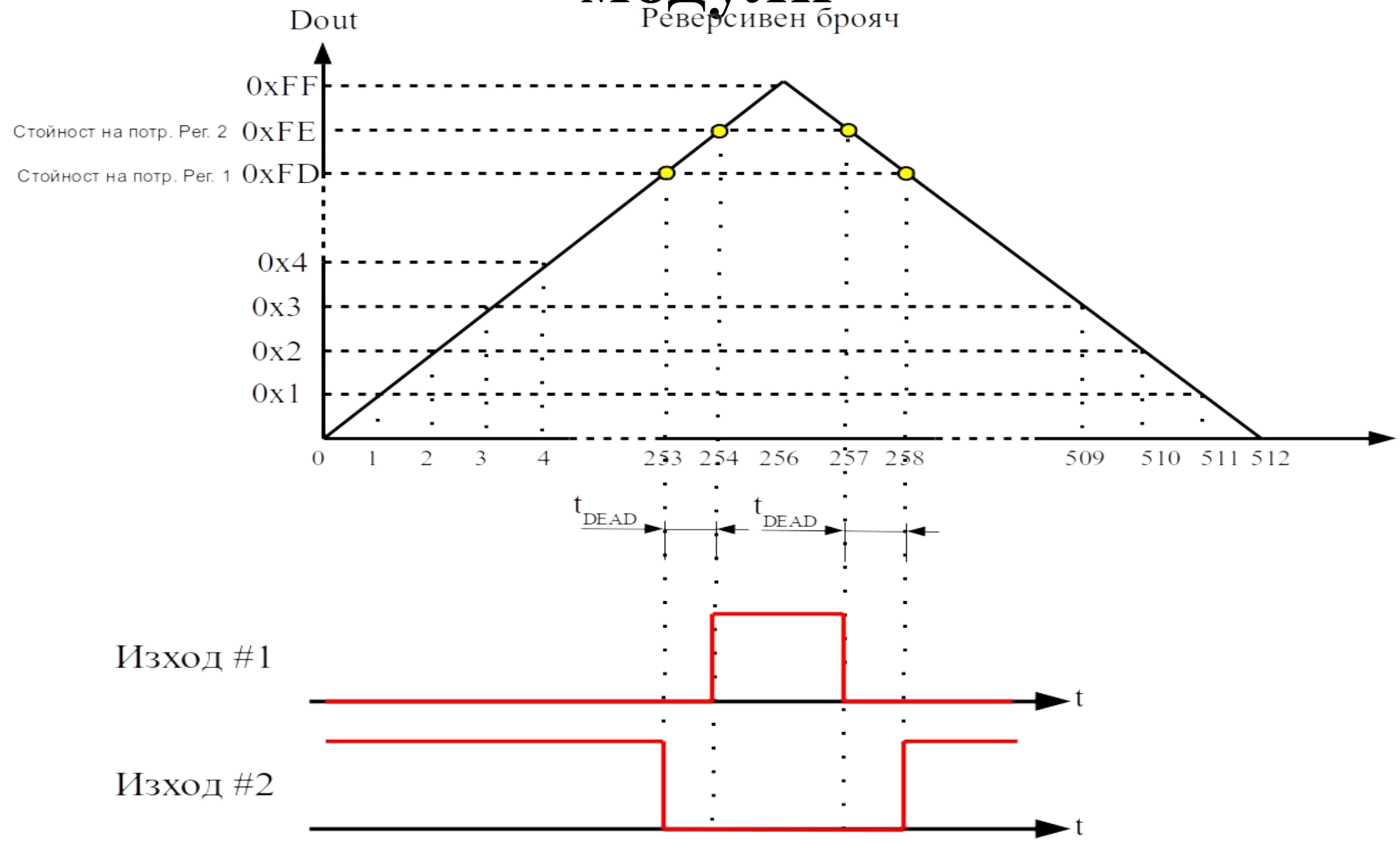


Режими на работа на таймерните модули

Първият или вторият фронт ще се застъпват, ако се използва сумиращ или изваждащ брояч на таймера. За силови схеми не трябва да се допуска!



Режими на работа на таймерните модули



Режими на работа на таймерните модули

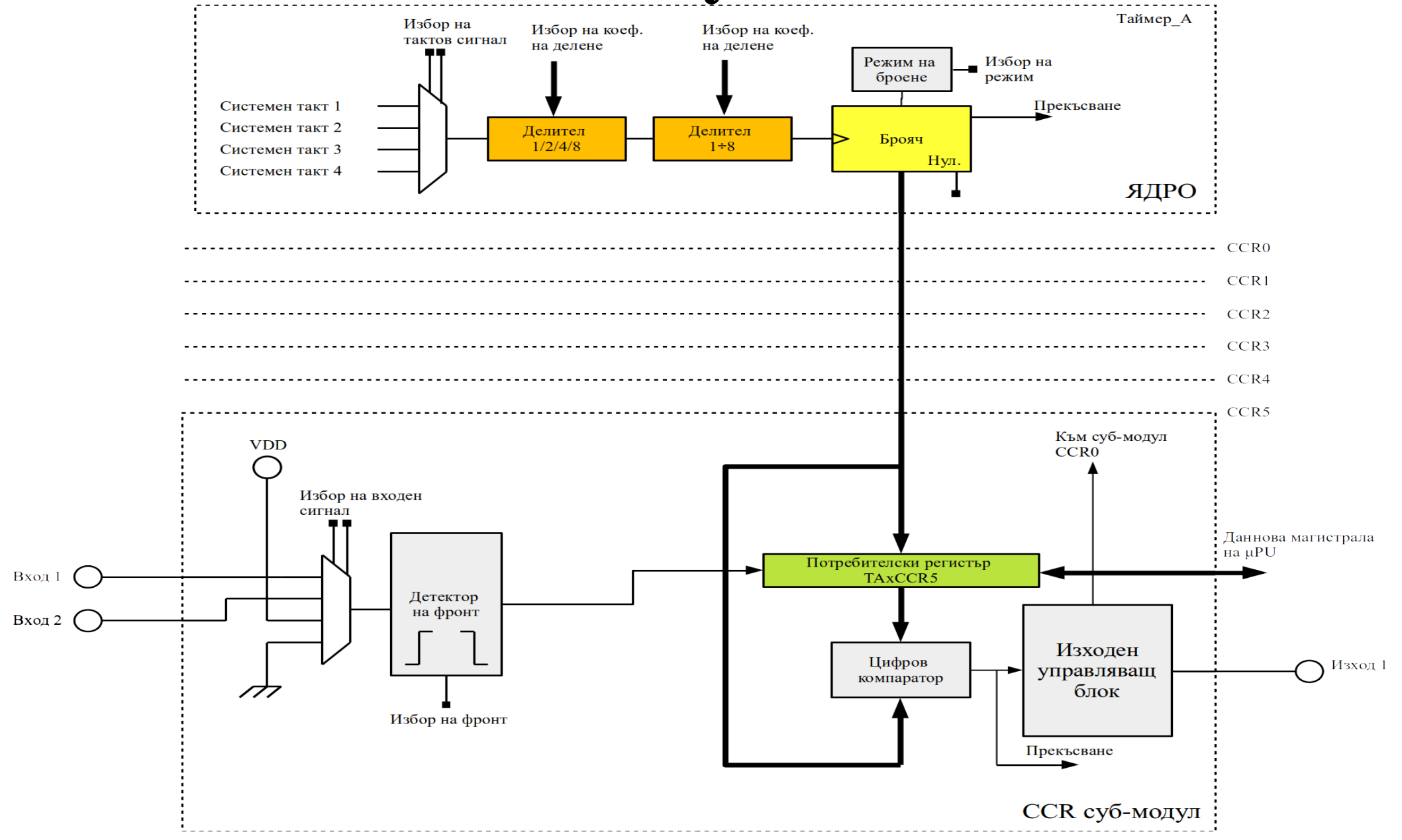
Пример – MSP430-базираните μCU имат еднотипни таймери, чиято обобщена структура е показана на следващия слайд.

Ядрото на модула се състои от един 16-битов комбиниран (сумиращ или реверсивен) брояч, който може да се тактува от 4 различни източника. Тактовият сигнал може да се дели от два предделителя.

Към изходната шина на брояча са закачени до 7 **суб-модула,** работещи в режим измерване-генериране. ШИМ режимът се реализира чрез комбиниране на 2 суб-модула. Суб-модул CCR0 задава периода, всички останали модули CCR1 ÷ CCR7 (или CCR1 ÷ CCR5 конкретно за блоковата схема).

=> максималният брой ШИМ сигнали, които могат да бъдат генерирани от 1 модул е 6 (или 4 конкретно за блоковата схема). **Всички ШИМ са с еднаква честота**!

Режими на работа на таймерните модули



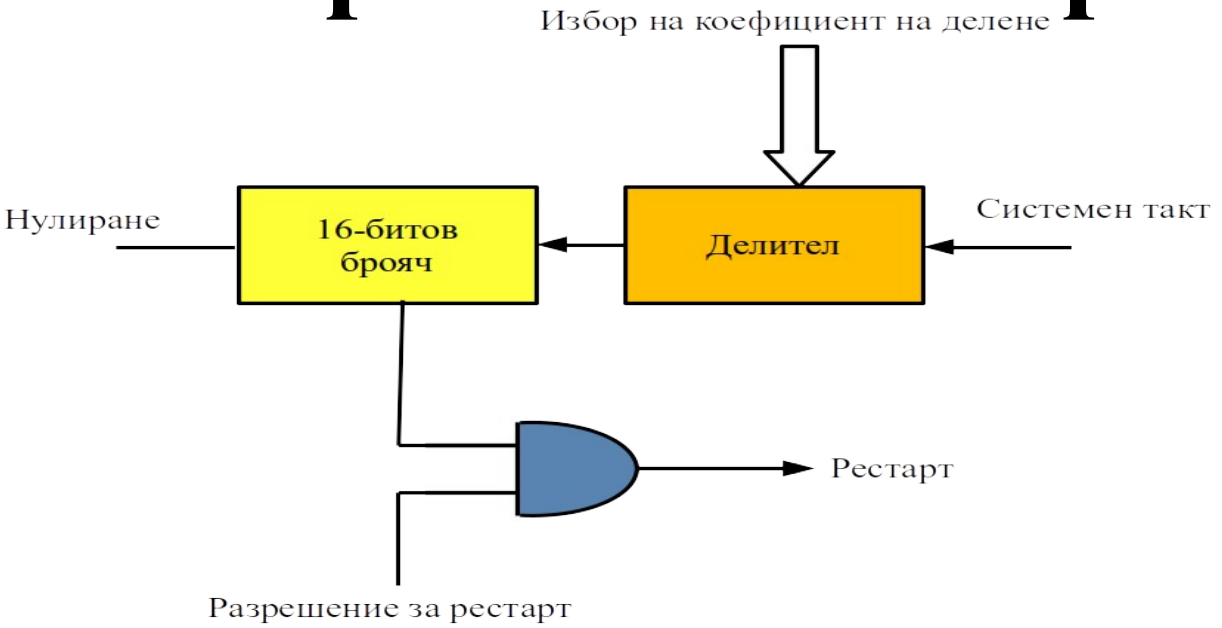
Стражеви таймери

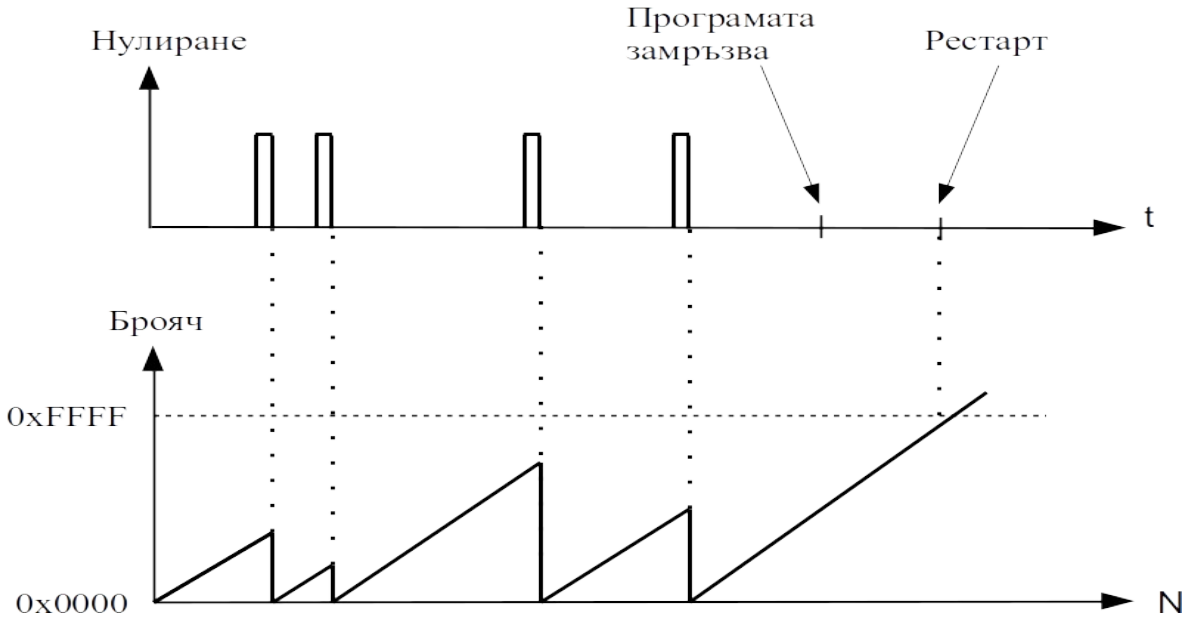
Стражеви таймери (WDT – **W**atch **D**og **T**imer) **-** модули, които се използват за автоматично рестартиране на микроконтролерите в случай, че управляващата програма замръзне (увисне) в някой нейн клон.

WDT е брояч, при препълването на който се рестартира микроконтролера. Задължение на управляващата програма е **да се нулира този брояч преди той да се препълни**. По този начин, ако програмата замръзне, WDT няма да бъде нулиран и ще рестартира системата след изтичане на определен интервал от време (x10 ms ÷ x1000 ms).

На фигурата на следващия слайд е показано действието на един 16-битов WDT.

Стражеви таймери





Стражеви таймери

Конфигурацията на стражевите таймери, чрез запис в контролните им регистри, **винаги е защитена** по някакъв начин. Най-често се използва парола за достъп.

Някой WDT имат схеми, които избират автоматично **алтернативен тактов сигнал** за модула, ако основният спре.

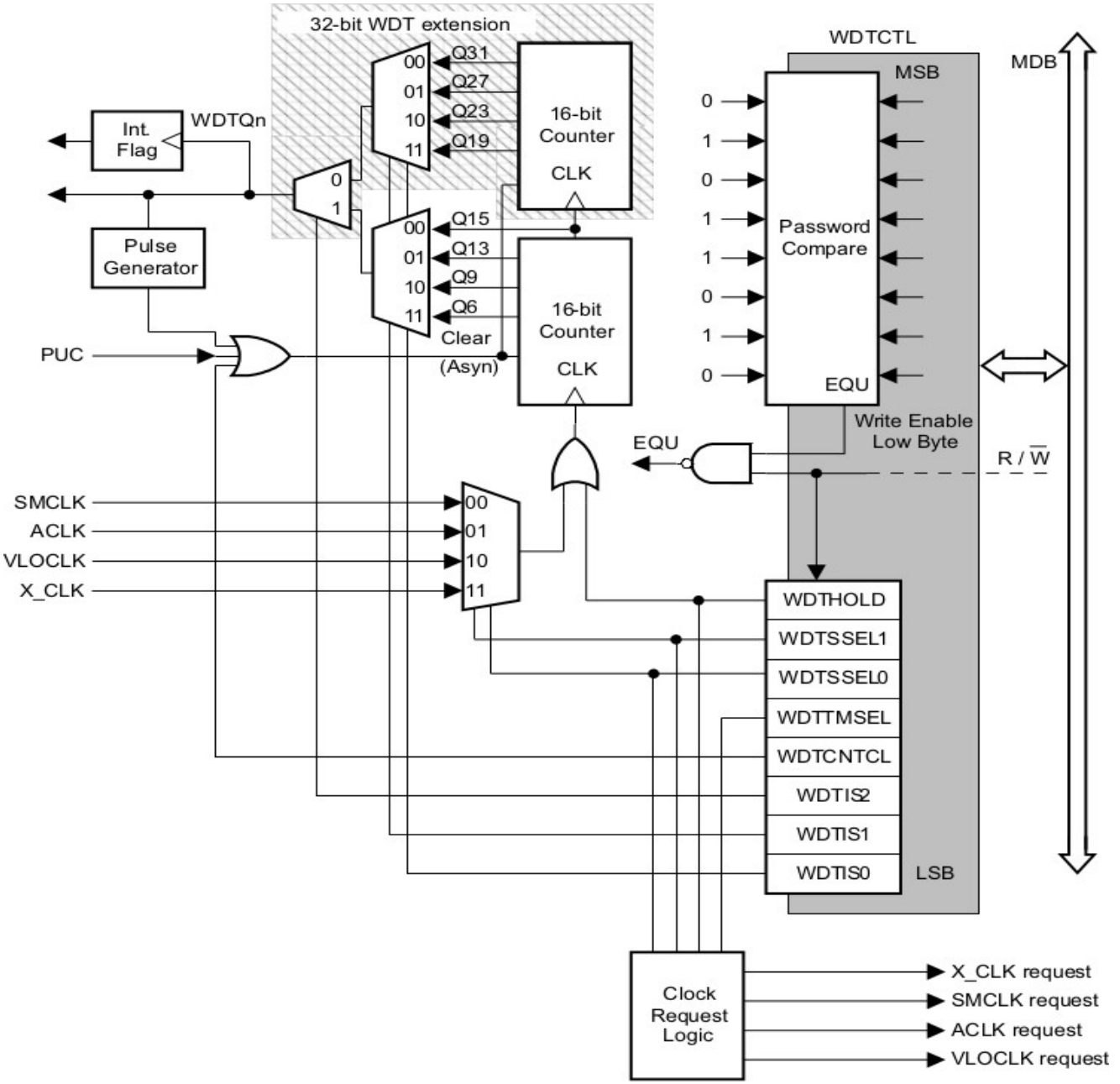
Почти всички μCU имат регистър, в който се записва **причината за рестартирането**. Той може да бъде прочетен и ако причината е рестарт от стражевия таймер, потребителският фърмуер може да опита да се справи с аварийната ситуация.

Стражеви таймери

Стражеви таймер на MSP430.

Спиране на таймера. Паролата се записва едновременно с конфигуриращия бит. Тя е числото 0x5A.

WDTCTL = 0x5A | WDTHOLD;



Стражеви таймери

Стражеви таймер на STM32L011.

1.Трябва да се разреши таймера с

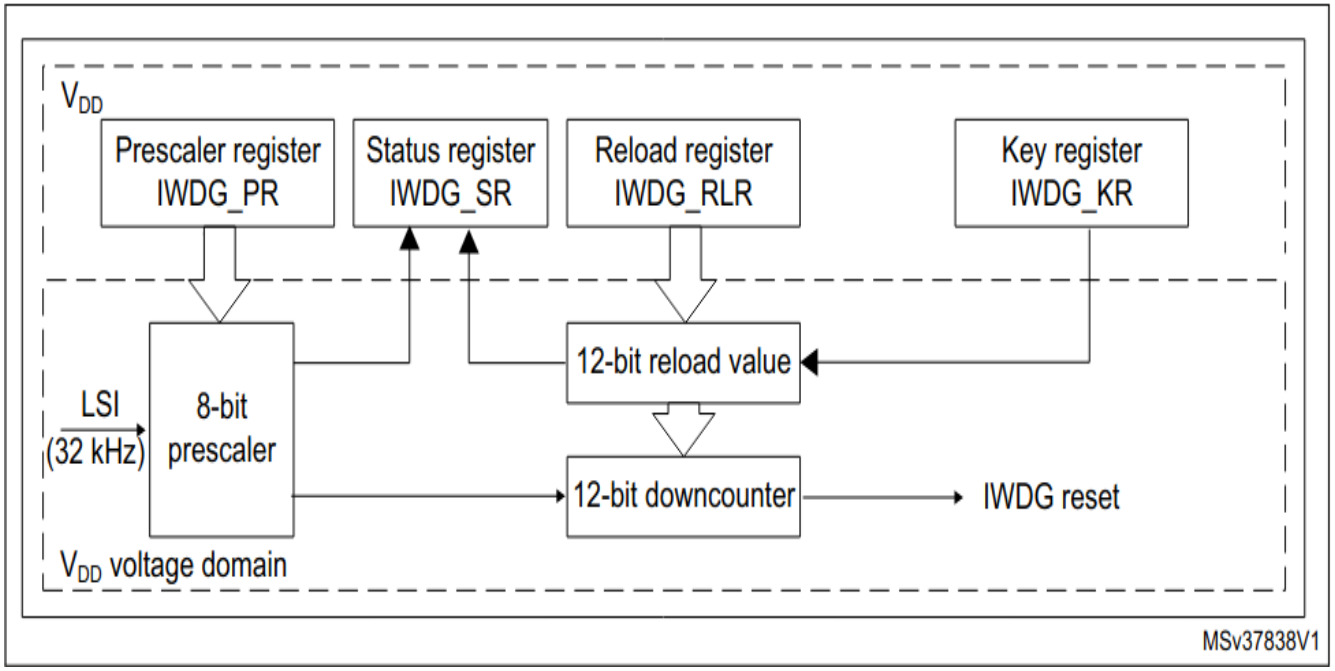
IWDG_KR = 0x0000CCCC;

2. Трябва да се въведе парола (в същия регистър):

IWDG_KR = 0x00005555;

3. Запис в контролен регистър.

Figure 178. Independent watchdog block diagram



Стражеви таймери

```
int main(void){
 init();
 prog_params_manual_prog();
 start_low_power_timer();

    while(1){
        HAL_PWR_EnterSTOPMode(PWR_LOWPOWERREGULATOR_ON, PWR_STOPENTRY_WFI);
        if(...){
        ...
        }
        if(...){
                if(...){
                        start_pause_timeout();
                }
                if(...){
                        play_beep();
                }
        }
        else{
                if(...){
                        stop_pause_timeout();
                }
                if(...){
                        battery_check();
                }
        }

        watchdog_refresh();
    }
}
```

Пример – STML011 в приложение с ниска енергийна консумация и стражеви таймер. Отделен таймер генерира прекъсвания на всеки 20 ms, което “събужда” while-цикъла в main( ) програмата и занулява стражевия таймер.

Стражеви таймери

```
void init(void){
    uint8_t reset_cause;

    HAL_Init();
    ...
    init_watchdog();
    reset_cause = watchdog_check_reset();

    if(reset_cause){
        //Force pinger turn on
        ...
    }
    else{
        //Normal start-up
            ...
    }
}
```

Пример – STML011 в уред, който трябва да се активира принудително, ако програмата увисне.

Стражеви таймери

```
uint8_t watchdog_check_reset(void){
    uint32_t reset_cause = 0;
    uint8_t val;
    reset_cause = (RCC->CSR & WWDGRSTF) | (RCC->CSR & IWDGRSTF);

    RCC->CSR &= ~(WWDGRSTF | IWDGRSTF);

    if(reset_cause){
        //Reset cause: WDT
        val = 1;
    }
    else{
        //Reset cause: non-WDT
        val = 0;
    }

    return val;
}
```

Пример – API функция за STML011 за проверка източника на рестарт. Този контролер има два стражеви таймера, затова се проверяват два флага на CSR регистъра.

Стражеви таймери

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR RSTF	WWDG RSTF	IWDG RSTF	SFT RSTF	POR RSTF	PIN RSTF	OBL RS TF	FW RSTF	RMVF	Res.	Res.	Res.	RTC RST	RTC EN	RTCSEL[1:0]	
r	r	r	r	r	r	r	r	rt_w				rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CSSLS ED	CSSLS EON	LSEDRV[1:0]		LSE BYP	LSERDY	LSEON	Res.	Res.	Res.	Res.	Res.	Res.	LSI RDY	LSION
	r	rw	rw		rw	r	rw							r	rw

Bit 30 **WWDGRSTF:** Window watchdog reset flag

This bit is set by hardware when a window watchdog reset occurs.
It is cleared by writing to the RMVF bit, or by a POR.
0: No window watchdog reset occurred
1: Window watchdog reset occurred

Bit 29 **IWDGRSTF**: Independent watchdog reset flag

This bit is set by hardware when an independent watchdog reset from V_{DD} domain occurs.
It is cleared by writing to the RMVF bit, or by a POR.
0: No watchdog reset occurred
1: Watchdog reset occurred

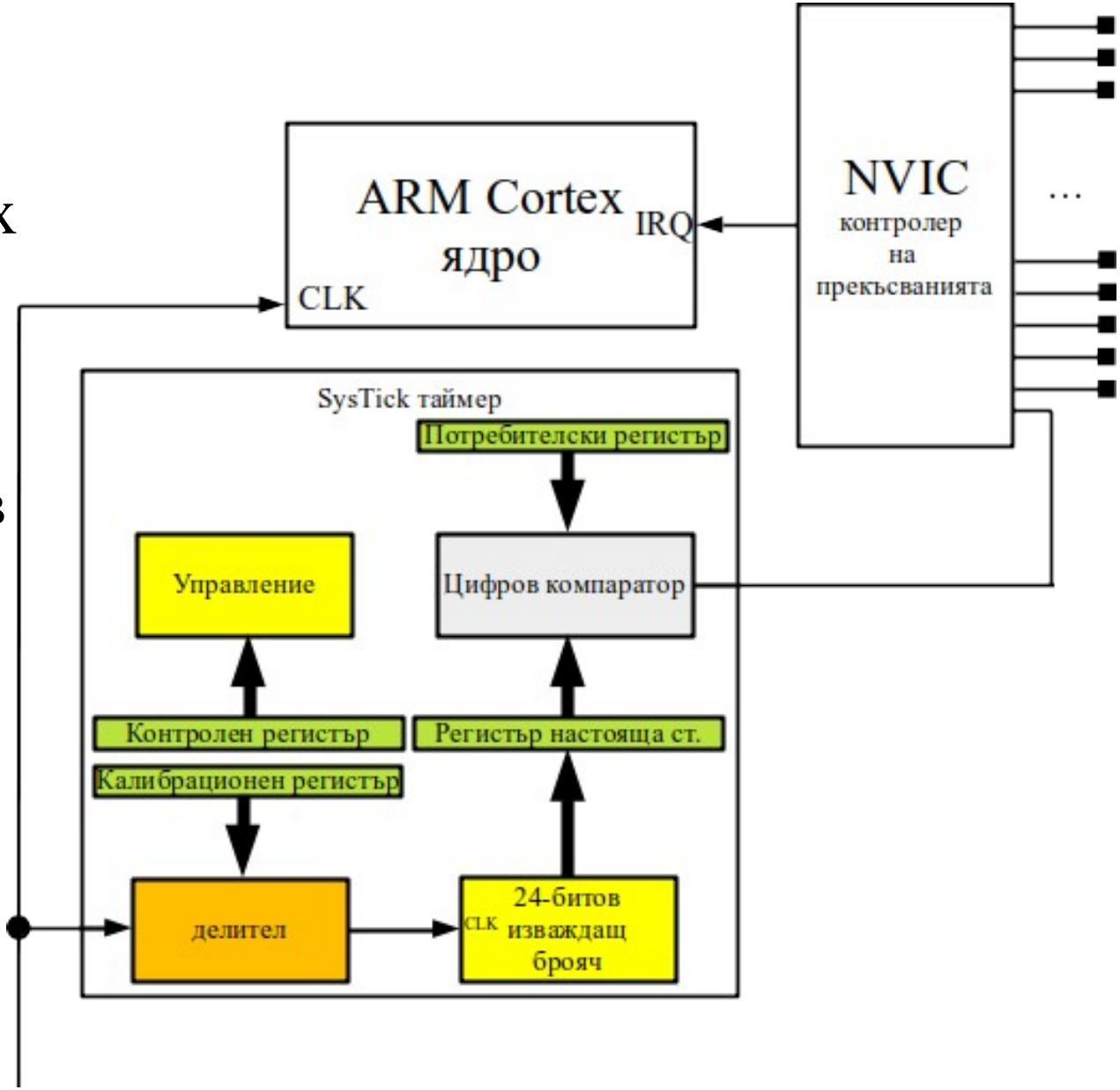
Системни таймери

Системни таймери (SysTick – System Tick) – таймери, използвани за генериране на прекъсвания на равни интервали от време за нуждите на библиотеки или системен софтуер, работещ в реално време.

Някои от приложенията на тези таймери са:
*в операционна система за реално време (за диспечера на ядрото ѝ)
*в системни библиотеки (за организиране на закъснения и периодична проверка за събития)
*във файлови системи (за вътрешната логика, house keeping, на системата).
Ако μCU не използва софтуер, изискващ SysTick, този таймер може да се използва с общо преназначение във всички приложения.

Системни таймери

Пример – микропроцесорите от фамилията ARM Cortex имат вграден системен таймер, чийто конфигурационни регистри са достъпни в адресното му поле.



Системни таймери

```
static HAL_StatusTypeDef ADC_Enable(ADC_HandleTypeDef* hadc){
  uint32_t tickstart = 0U;

  if (ADC_IS_ENABLE(hadc) == RESET){
    if (ADC_ENABLING_CONDITIONS(hadc) == RESET){
      SET_BIT(hadc->State, HAL_ADC_STATE_ERROR_INTERNAL);
      SET_BIT(hadc->ErrorCode, HAL_ADC_ERROR_INTERNAL);
      return HAL_ERROR;
    }

    __HAL_ADC_ENABLE(hadc);

    ADC_DelayMicroSecond(ADC_STAB_DELAY_US);

    /* Get tick count */
    tickstart = HAL_GetTick();

    /* Wait for ADC effectively enabled */
    while(__HAL_ADC_GET_FLAG(hadc, ADC_FLAG_RDY) == RESET){
      if((HAL_GetTick() - tickstart) > ADC_ENABLE_TIMEOUT){
        SET_BIT(hadc->State, HAL_ADC_STATE_ERROR_INTERNAL);
        SET_BIT(hadc->ErrorCode, HAL_ADC_ERROR_INTERNAL);
        return HAL_ERROR;
      }
    }
  }
  return HAL_OK;
}
```

Пример – HAL библиотеката на STM32L011 използва SysTick на μPU за таймаути.

Часовници за реално време (RTC)

Часовниците за реално време (RTC, **R**eal **T**ime **C**lock) - периферни схеми за отмерване на астрономическо време. Те представляват един брояч, към който са свързани няколко регистъра. Стойността на брояча се увеличава с едно на всяка една секунда. В регистрите се записва текущата стойност, като при препълването на един регистър започва попълването на следващия. Препълването на всеки регистър е пропорционално на:

*секунди
*минути
*часове
*дни
*седмици
*месеци
*години

Часовници за реално време (RTC)

Почти винаги RTC притежават **резервно захранване**, което позволява отмерване на времето, **дори когато системата не е включена**. Типичен пример са персоналните компютри – дори да са изключени, часът и датата остават верни.

За тактова честота на брояча се използват **генератори с изходна честота 1 Hz**. Тази честота трябва да е кварцово стабилизирана.

За да се генерира точно честота от 1 Hz, кварцовият резонатор трябва да има резонансна честота **кратна на степените на 2**. Така може да се използва препълването на цифров брояч, който ще има **най-проста хардуерна структура**, ако брои до числа, кратни на степ. на 2.

Часовници за реално време (RTC)

Например 4.096 MHz е валидна стойност. Ако се използва кристал 4 MHz, с времето часовника ще изостава.

В схемотехниката са се наложили кристали с честота 32768 Hz, който се намират лесно и се произвеждат специално за това приложение. Предимството на 32.768 кHz пред 4.096 MHz е, че **консумацията на модула ще бъде по-малка**.

Използва се 15-битов брояч ($2^{15} = 32768$) брояч, който ще се препълва 1 на всяка секунда, ако бъде тактуван от генератор с 32.768 кHz честота.

Часовници за реално време (RTC)

При избирането на кварц за такива цели е важна **точността** на честотата и **температурния обхват** на работа. Ако се налага голяма точност, някои от предлаганите на пазара ИС на часовници за реално време имат отделни регистри за допълнителна калибровка.

В зависимост от това дали са интегрирани в микроконтролера или не, RTC могат да се разделят на:
*вътрешни RTC
*външни RTC

Часовници за реално време (RTC)

При **външните RTC** се използват най-често **интерфейсите** паралелен, I^2C и SPI за връзка с микроконтролера. На фигурата на по-следващия слайд е дадена примерна блокова схема на часовник за реално време.
Понякога в RTC се вгражда и малка по обем **RAM памет,** която може да се използва от μCU за съхраняване на различни конфигурационни данни. Те ще се запазят и след изключването на устройството, защото към RTC има **допълнителна батерия** (най-често 3-волтова, 2032).

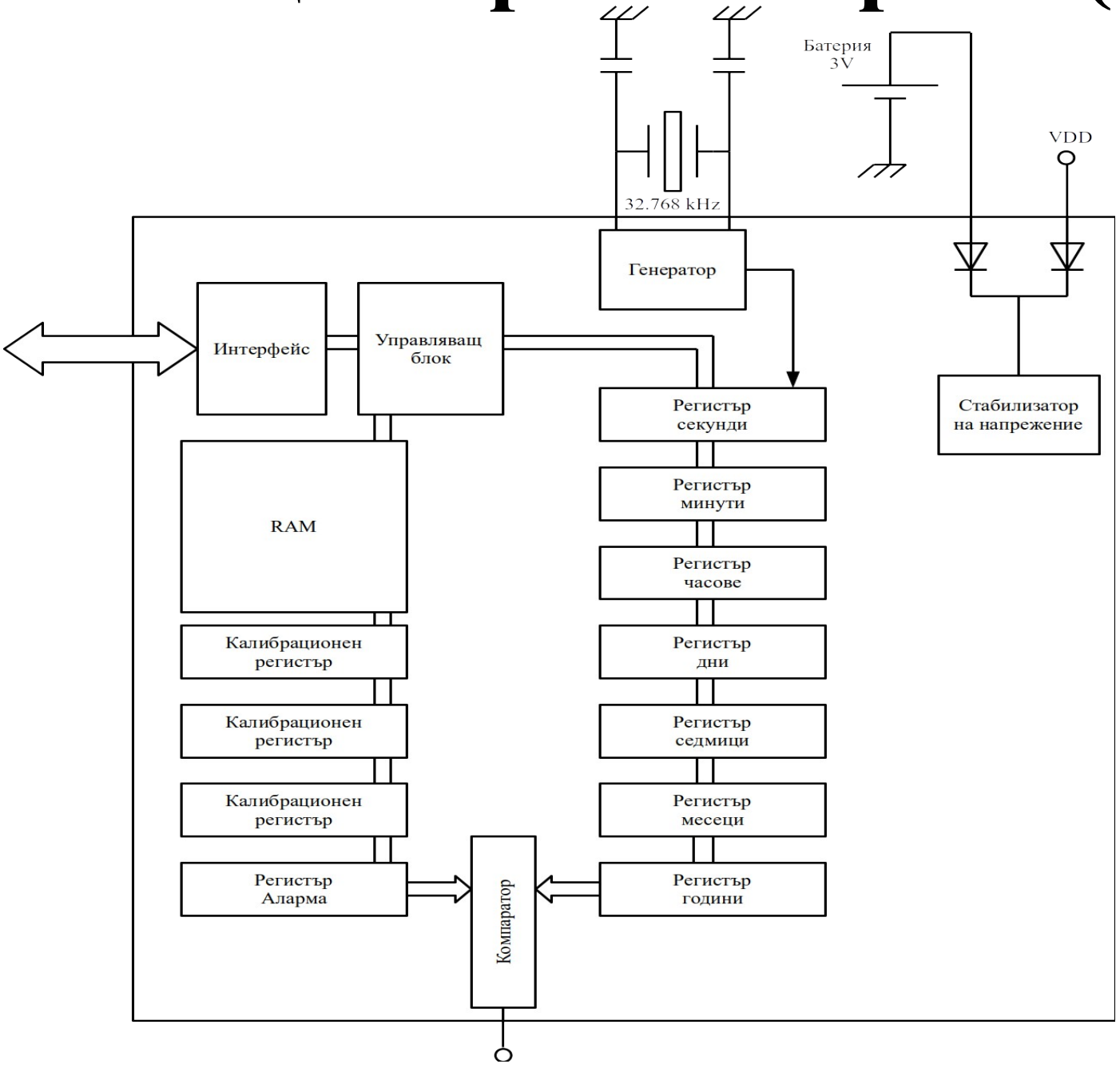
Предимството на тази RAM пред използването на EEPROM или Flash е, че тя няма ограничение на броя на циклите четене/запис. Недостатък – при премахване или разряд на батерията данните се губят.

Часовници за реално време (RTC)

Често в RTC се вгражда допълнителна **алармена система,** състояща се от един регистър и компаратор.

В регистъра се записва час и дата, когато искаме алармата да се задейства и компаратора постоянно сравнява тази стойност с регистрите на часовника. При съвпадение, флагът се прочита софтуерно, а понякога на ИС има и **отделен извод за сигнал**.

Часовници за реално време (RTC)



Часовници за реално време (RTC)

При **вътрешните RTC** регистрите са част от адресното поле на μPU.

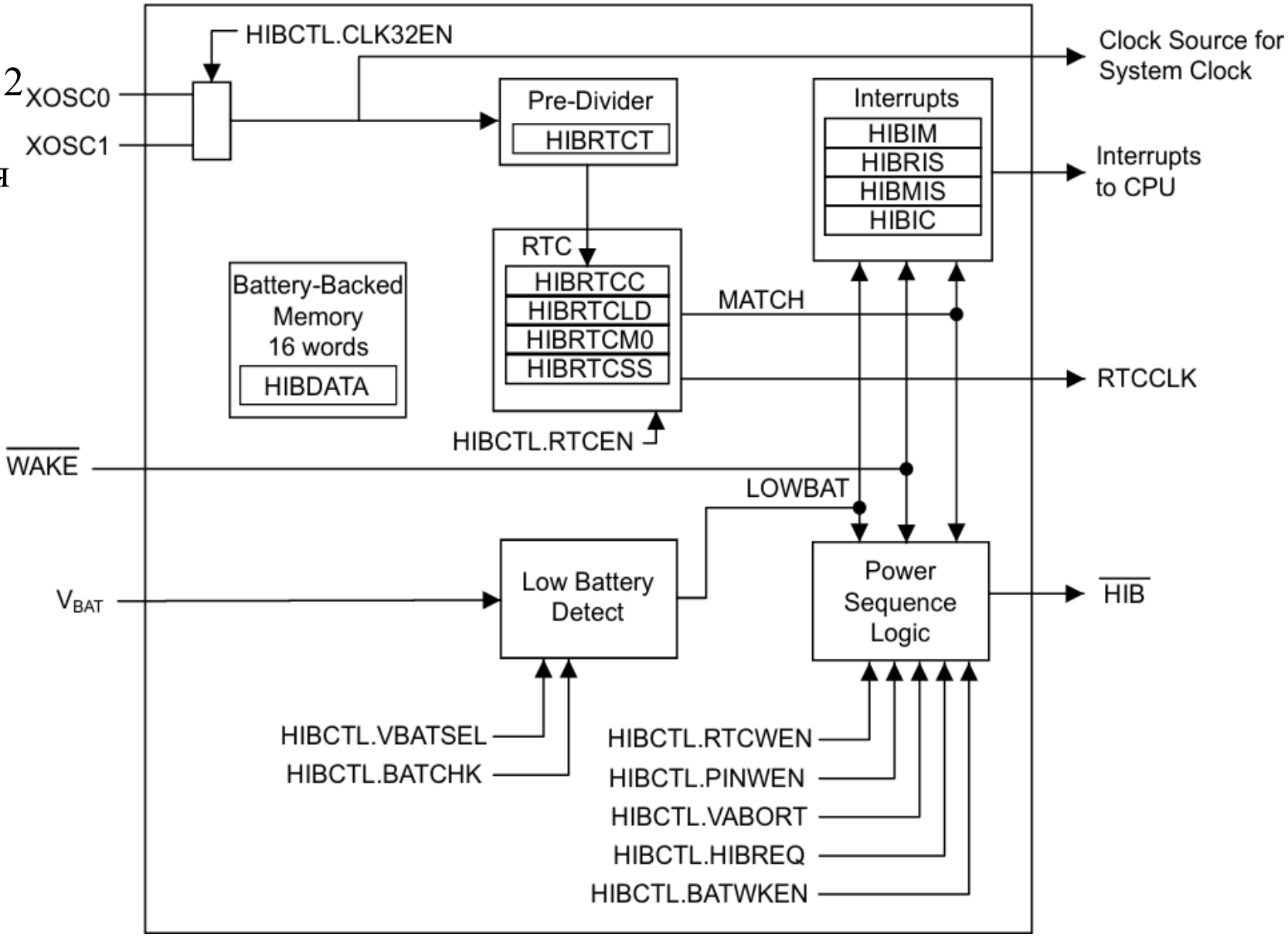
Захранването на RTC модулът е изведено на **отделен пин** на корпуса на μCU.

Кварцовият резонатор за RTC генератора също се извежда на **отделни изводи**.

Часовници за реално време (RTC)

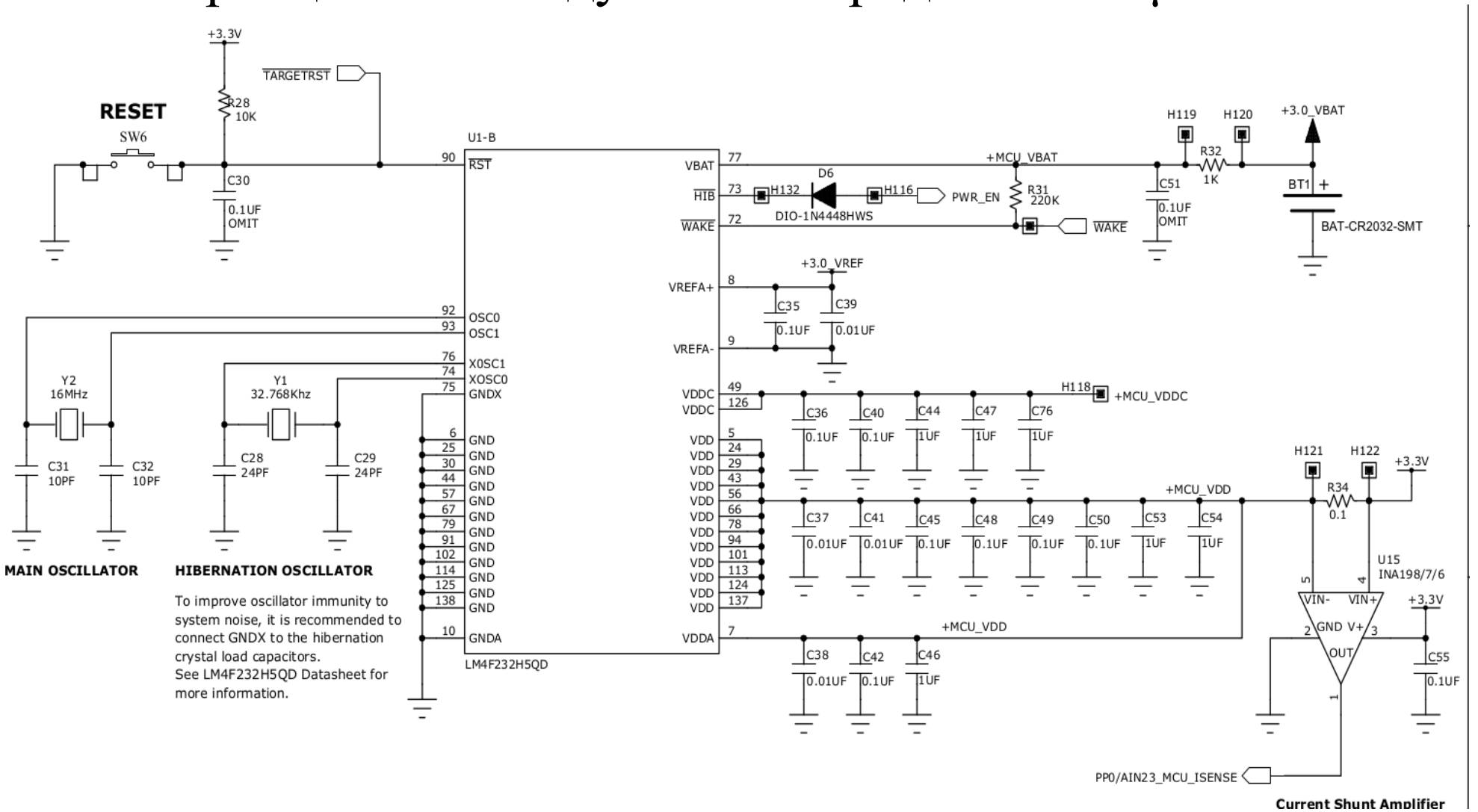
Пример - RTC на LM4F232 е част от хибернационния модул на MCU.

Figure 7-1. Hibernation Module Block Diagram



Часовници за реално време (RTC)

Пример - забележете стойността на шунтовия резистор R32, свързан към VBAT извода. Това е възможно, понеже токовете към хибернационния модул са от порядъка на x1μA.



Схеми за начално установяване

Схемите за начално установяване (НУ) се използват за правилното стартиране на микроконтролера и следене на захранващото напрежение по време на работа. Те биват няколко вида:

***POR** (**P**ower-**o**n **R**eset) – схема, която държи микроконтролера в рестарт, докато захранващото напрежение не премине даден праг при първоначалното пускане на системата. Така се осигурява нормалното захранване на логиката на процесора.

Схеми за начално установяване

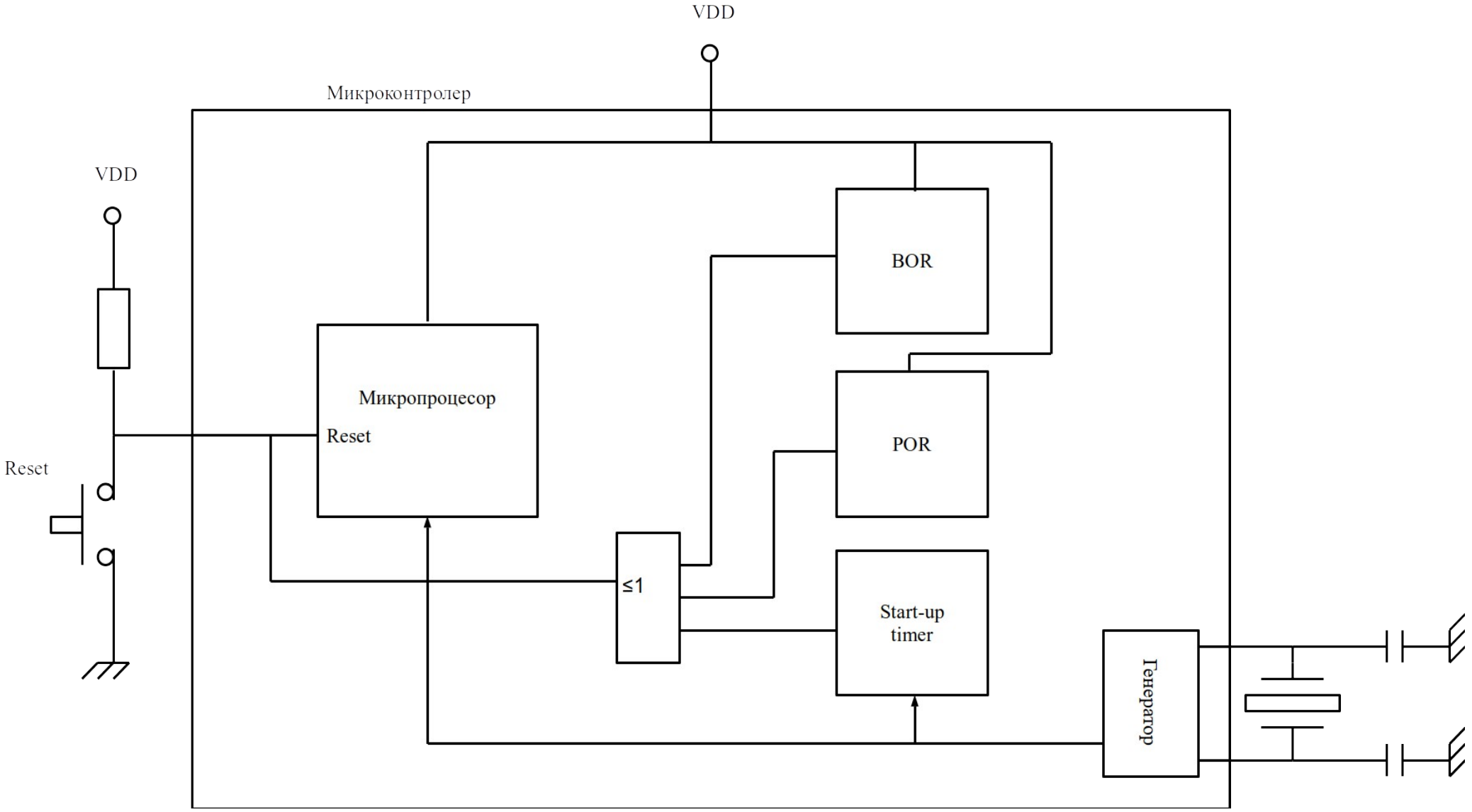
BOR (Brown-out reset) – схема, следяща за наличието на постоянно напрежение, докато микроконтролера работи. При временни пропадания в захранването (наречени още brown-out състояния) тази схема задържа ресет сигнала за известен период от време, след което го пуска. Понякога се реализира с хистерезис – след пропадането под една стойност, трябва да се върне над друга, за да продължи нормално работа.

Схеми за начално установяване

Start-up таймер – това е брояч, който отмерва фиксиран брой системни тактови сигнали при начално установяване и държи микроконтролера в ресет през това време. По този начин се гарантира, че тактовата честота се е стабилизирала (всеки генератор на сигнал при първоначалното си пускане формира нестабилни по амплитуда сигнали, което би довело до грешна работа на системата).

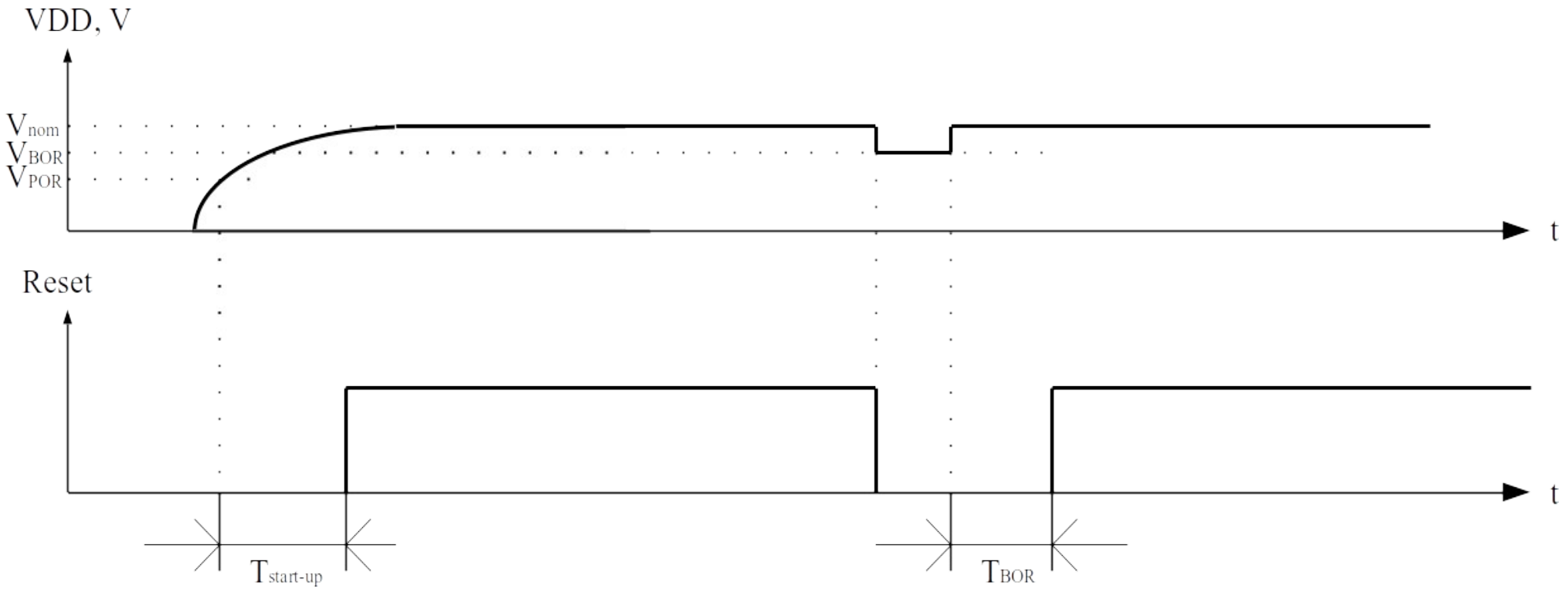
Low-bat детектор – схема следяща за разреждането на батерия, свързана към микроконтролера. Най-често става дума за батерията на RTC. Ако нейното напрежение спадне под един определен праг, генерира се прекъсване и управляващата програма трябва да извести потребителя за настъпилото събитие.

Схеми за начално установяване



Схеми за начално установяване

Работа на НУ модулите е демонстрирана на графиката по-долу. Използва се BOR схема без хистерезис.



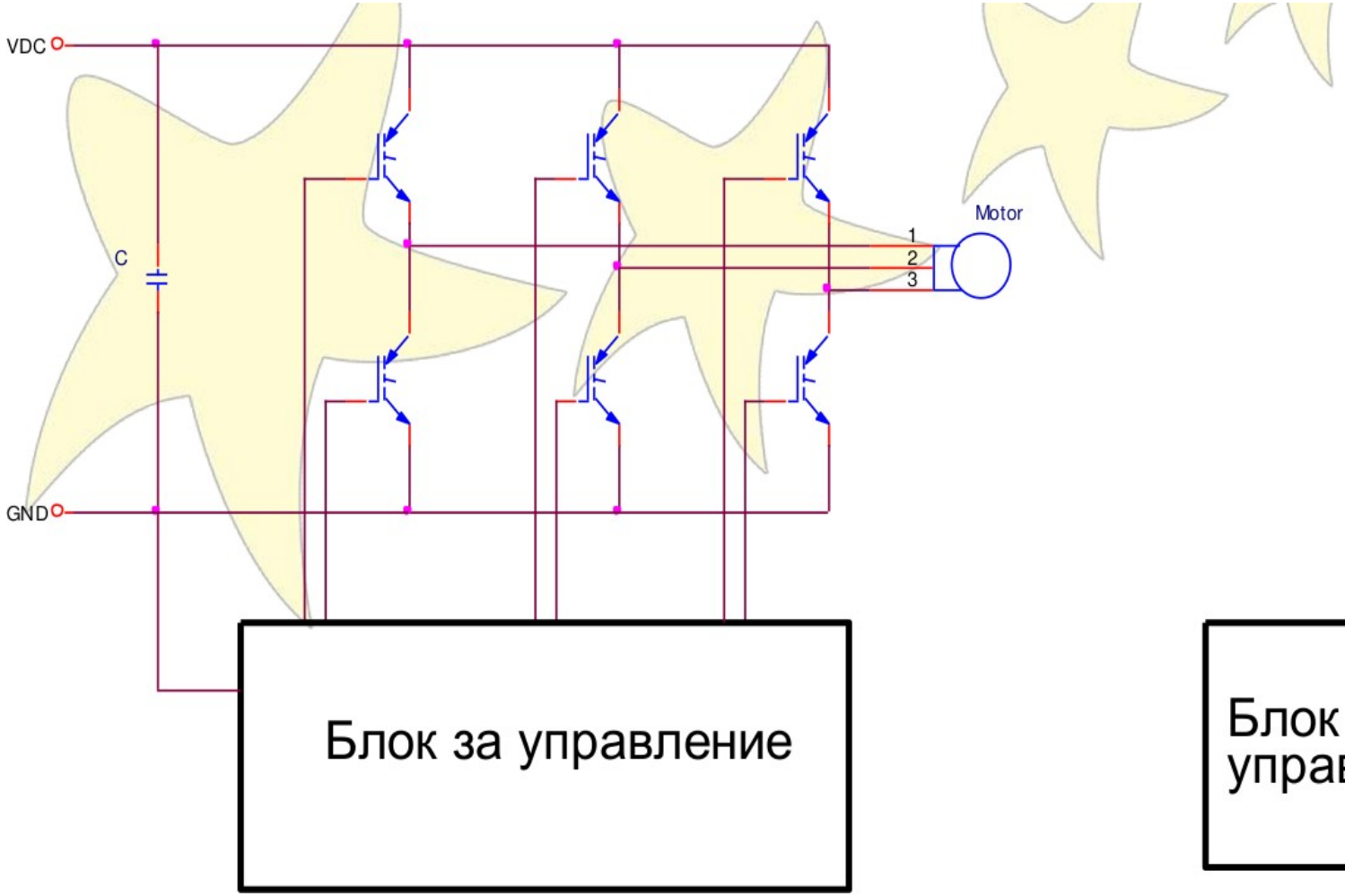
Блокове за управление на ел. двигатели

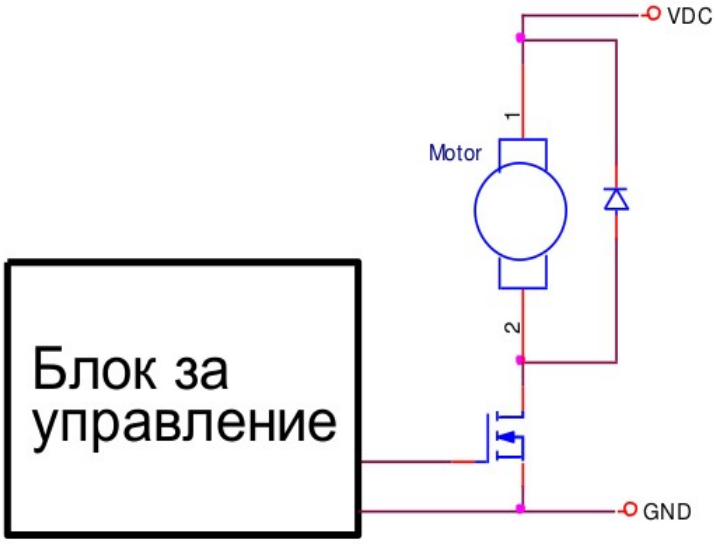
За управлението на електродвигатели се таймерни модули с възможност за генериране на ШИМ.

В зависимост от вида на електродвигателя най-често се използват от един до шест изхода с ШИМ.

На схемата на следващия слайд са показани начините на свързване на четков (дясно) и безчетков (ляво) постояннотоков електродвигател. Безчетковите мотори се наричат още стъпкови мотори.

Блокове за управление на ел. двигатели





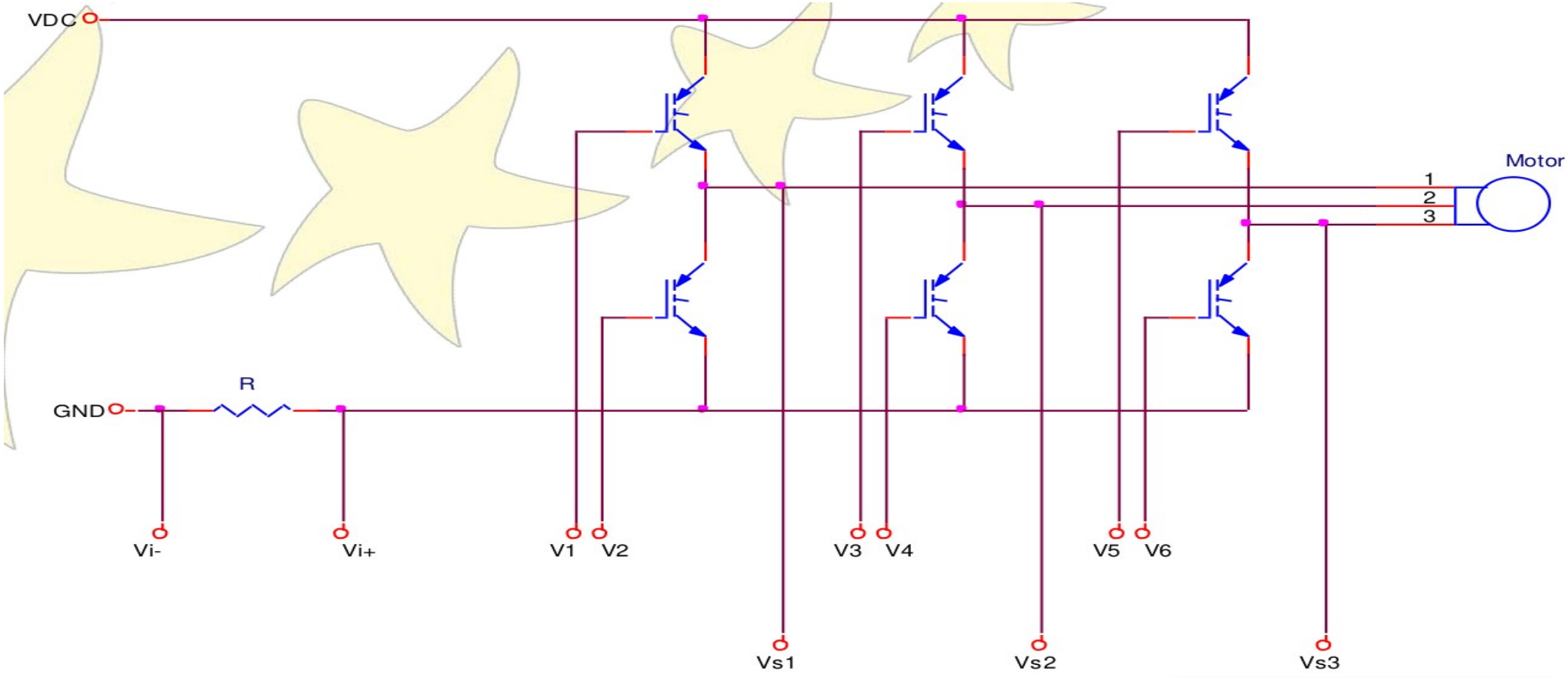
Блокове за управление на ел. двигатели

Често при управлението на мотори се използва **безсензорен метод** за отчитане позицията на ротора.

За тази цел се измерват напреженията, индуцирани от обратното ЕМП.

На схемата на следващия слайд това са Vs1, Vs2 и Vs3. Допълнително трябва да се следи и тока през мотора (Vi- и Vi+) с АЦП на μCU.

Блокове за управление на ел. двигатели



Блокове за управление на ел. двигатели

Стартирането и спирането на двигателя трябва да стане **по точно определен закон**, дефиниран от механиката.

Пример – на стъпковият мотор на глава от термопринтер за касови апарати M-T173H трябва да се подаде ШИМ с променяща се честота. **Честота се изменя в 25 стъпки** с точно-определена продължителност (дава се от производителя).

Забележете изискванията за **точността: ± 50µs**.

Блокове за управление на ел. двигатели

Стартиране:
25 стъпки
≈30 ms общо

2.3.8 Slow-up sequence

Table 2.3.2 shows the example slow-up sequence.

Table 2.3.2 Slow-up Sequence

Step	Motor drive frequency (pps)	Motor drive cycle (μs)
Start rush-driving	-	30000
1	332	3013
2	430	2327
3	512	1953
4	584	1712
5	649	1541
6	708	1412
7	763	1310
8	815	1227
9	864	1158
10	910	1099
11	954	1048
12	996	1004
13	1037	964
14	1076	929
15	1114	898
16	1150	869
17	1186	843
18	1220	819
19	1254	798
20	1287	777
21	1318	758
22	1350	741
23	1380	725
24	1410	709
25	1440	694

(Units: ms, Precision: ± 0.05 ms in the deceleration and acceleration range, however, +0.05/–0 ms in the constant speed range)

Блокове за управление на ел. двигатели

QEI (**Q**uadrature **E**ncoder **I**nterface) – модули за измерване на линейно или ротационно изместване чрез импулсни сигнали. Среща се и името ESI (**E**xtended **S**can **I**nterface).

Наричат се още двуканални инкрементиращи енкодери.

В практиката се използват сензори за позиция, чиито изход изработва единични импулси при преместването на обекта (напр. ротор на мотора).

Чрез измерване броя на импулсите и фазата между два канала може да се измери позиция, посока на въртене и скорост на въртене.
Най-често тези модули се използват при сензорно управление на електродвигатели.

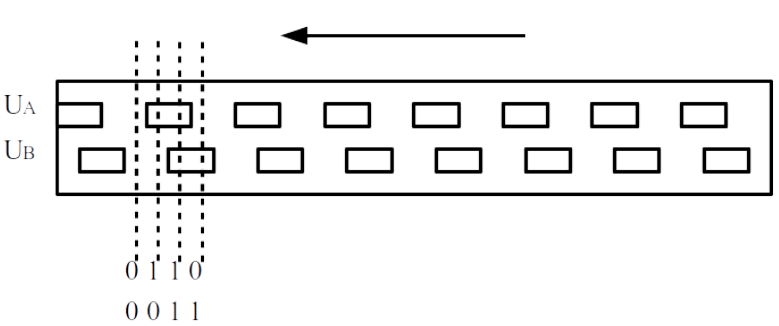
Блокове за управление на ел. двигатели

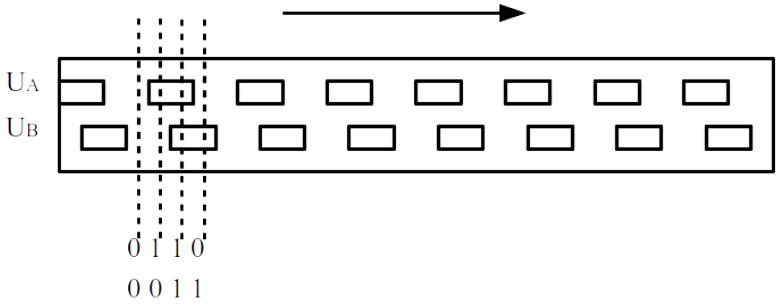
На следващия слайд е представена опитна постановка за измерване на **линейно преместване**.

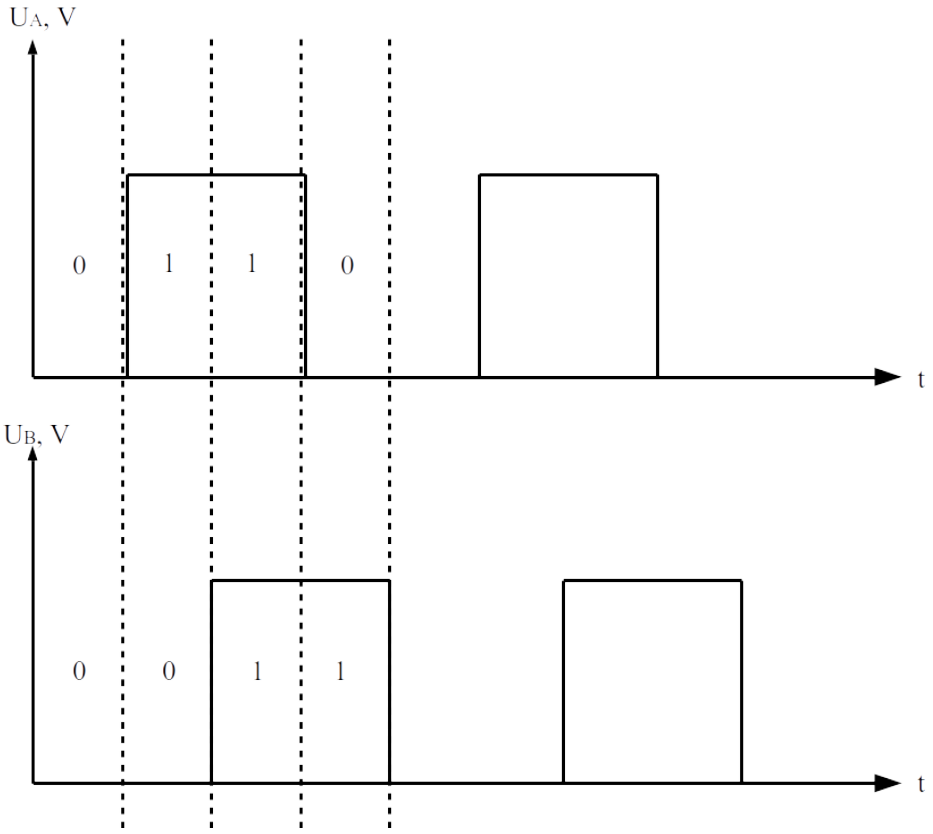
За целта се използва лента, която е с отвори, изместни един спрямо друг на **90 градуса**.

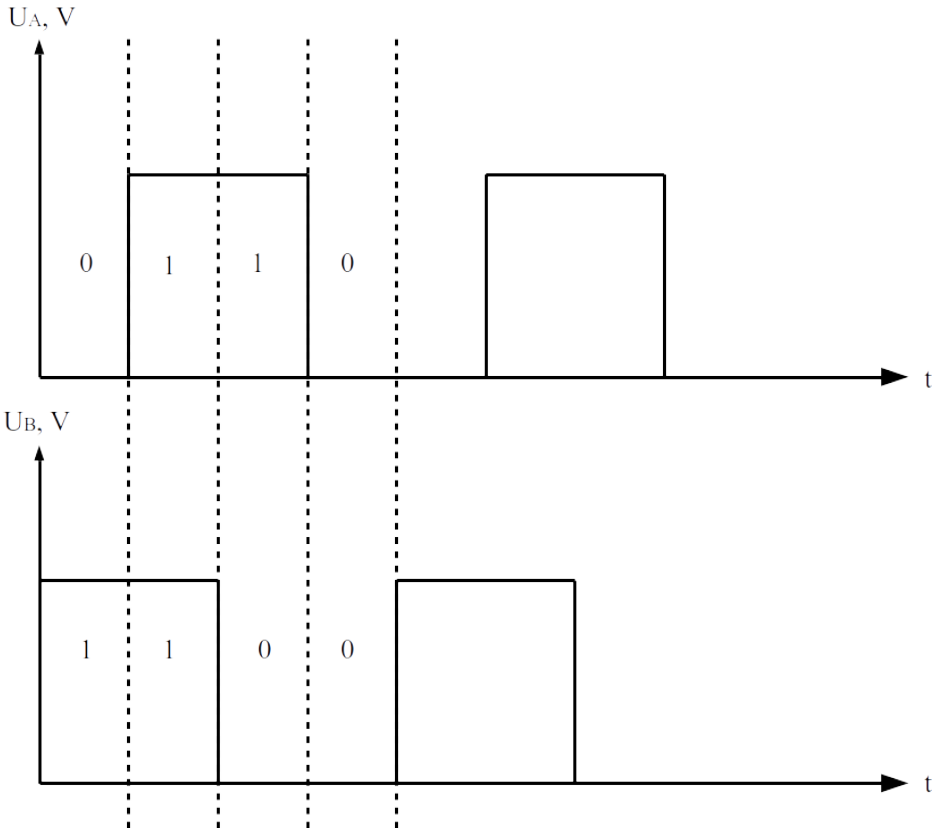
От едната страна на лентата има **IR излъчватели**, а от другата – **IR приемници**.

Блокове за управление на ел. двигатели









Блокове за управление на ел. двигатели

Когато лентата се движи наляво, поредицата от числа, която ще се засече е 00, 10, 11, 01 (или 0, 2, 3, 1).

Когато се движи надясно – 01, 11, 10, 00 (или 1, 3, 2, 0).

По поредицата може да се съди за посоката на движение (ляво или дясно).

Ако се измери честотата на единия от сигналите може да се определи скоростта на преместване, като се знаят броя на отворите.

Текущата позиция може да се измери спрямо началната чрез подходящ софтуерен алгоритъм и предварително известна стартова позиция.

Литература

[1] Г. Михов, “Цифрова схемотехника”, ТУ-София, 1999.