

# Дисплеи и индикация във вградените 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!***



Европейски социален фонд

# Съдържание

1. Управление на LED индикатори
2. Управление на LCD дисплеи
3. Управление на OLED дисплеи
4. Бутони и клавиатури
5. Тъч сензори
6. Ротационни енкодери

# Управление на LED индикатори

Светодиодната индикация (Light Emitting Diode, LED) може да се раздели на:

- \*индикация с един светодиода
- \*индикация със 7-сегментни индикатори
- \*индикация с буквено-цифрови индикатори
- \*индикация със светодиодни матрица

В зависимост от това дали в даден момент се управляват всички сегменти/пиксели, се казва, че индикацията е [1]:

- \*статична
- \*динамична

# Управление на LED индикатори

От курса ППЕ е известно, че светодиодите имат пад на **напрежение в права посока  $V_F$** , зависещ от цвета на светодиода.

Типични стойности за дифузни LED са:

	$V_{Fmin}, V$	$V_{Fmax}, V$
<b>Червен</b>	1.8	2.2
<b>Зелен</b>	2	2.3
<b>Жълт</b>	2.2	2.8
<b>Бял</b>	3.2	3.4
<b>Син</b>	3.2	3.4

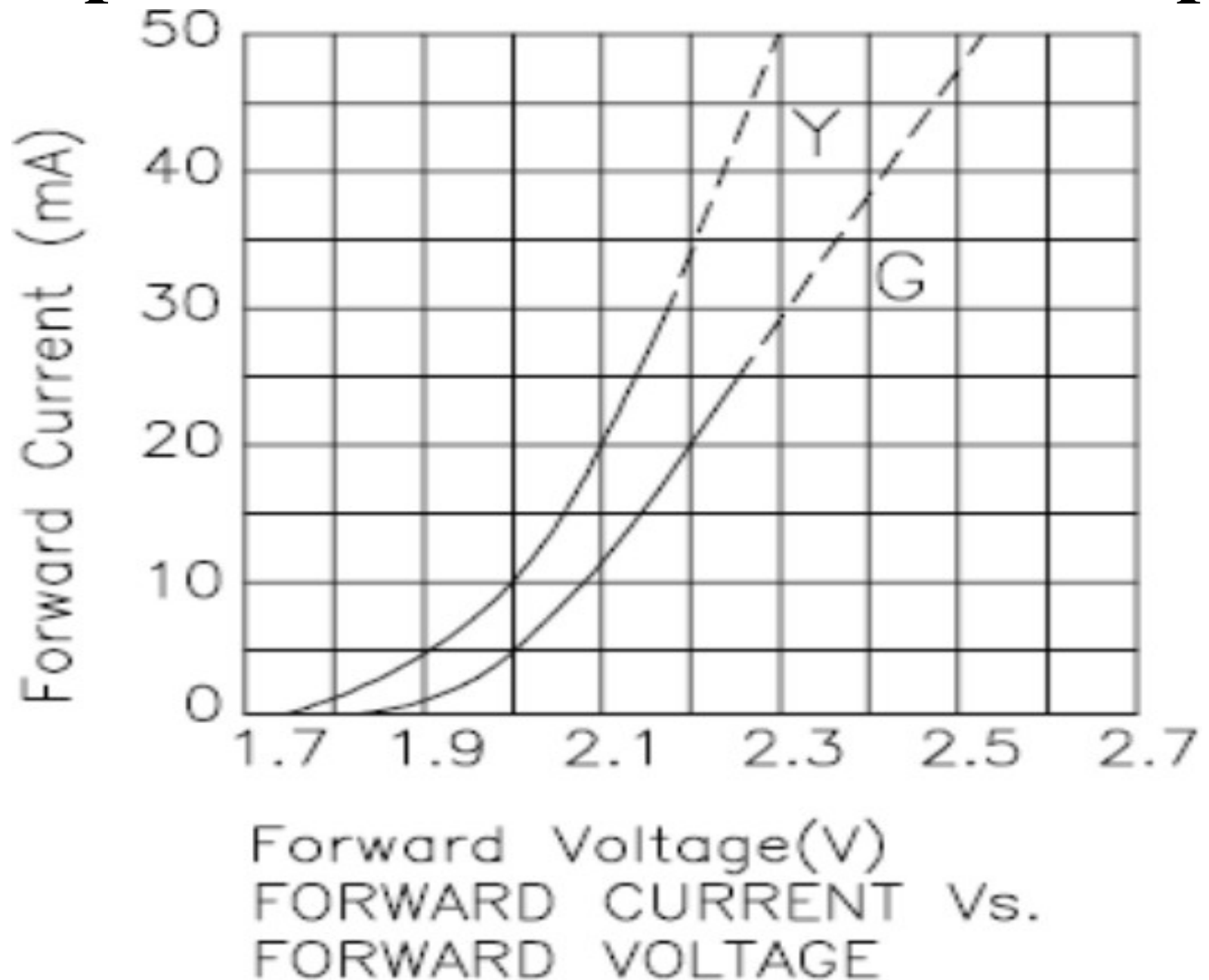
# Управление на LED индикатори

Ярките светодиоди (bright LED) имат по-високи падове и светят по-ярко от дифузните при едни и същи токове (напр. ярък зелен LED може да има  $V_F = 3\text{ V}$ ).

**Токът в права посока  $I_F$**  варира в по-големи граници. Дифузните светодиоди имат  $I_F = 10 \div 30\text{ mA}$ , ярките  $I_F = 1 \div 20\text{ mA}$ , а мощните –  $x1 \div x10\text{ A}$ .

Типична ВАХ на маломощен LED е показана на следващия слайд[2].

# Управление на LED индикатори



# Управление на LED индикатори

Яркостта на светодиодиите (luminous intensity) се мери в кандели и за индикаторни светодиоди варира в обхвата  $0.6 \div 1800 \text{ mcd}$ .

Интензитетът зависи от тока в права посока  $I_F$ [3]:

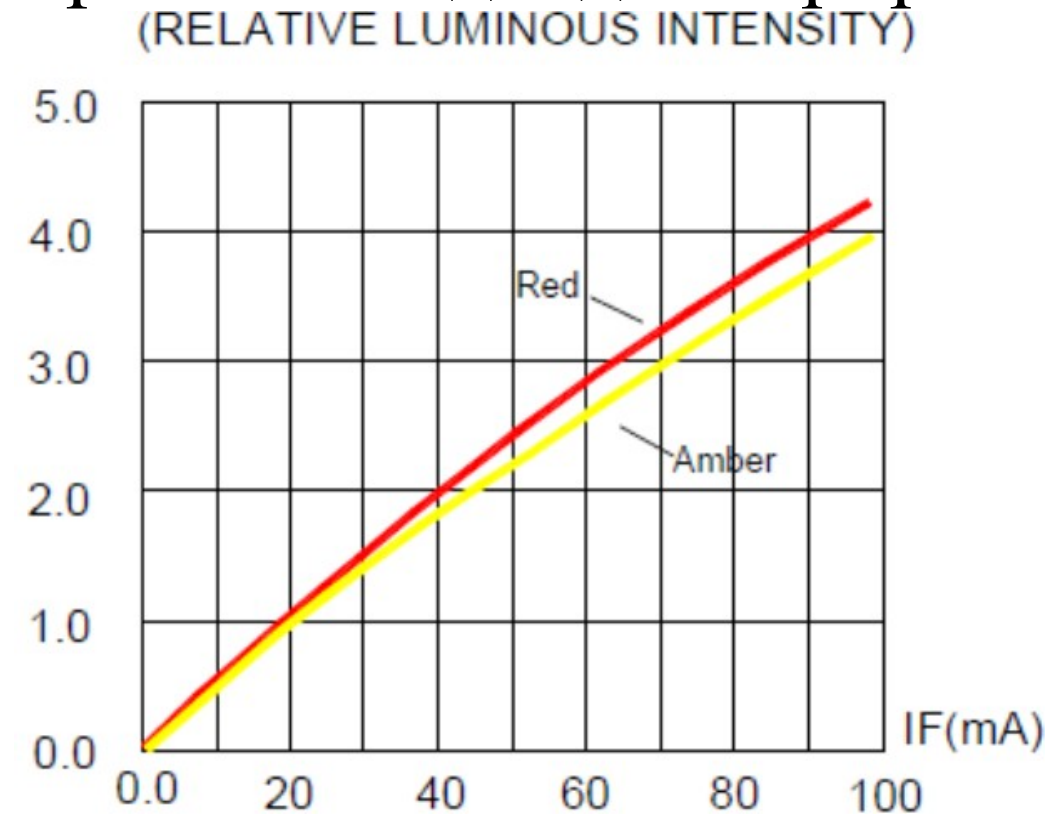


FIG.2 RELATIVE LUMINOUS INTENSITY VS. FORWARD CURRENT



# Управление на LED индикатори

Яркостта на светодиодиите зависи от ъгъла, от който наблюдаващия вижда светодиода. Това се нарича **ЪГЪЛ на виждане** (viewing angle) [3]. Използва се също понятието “far field pattern”.

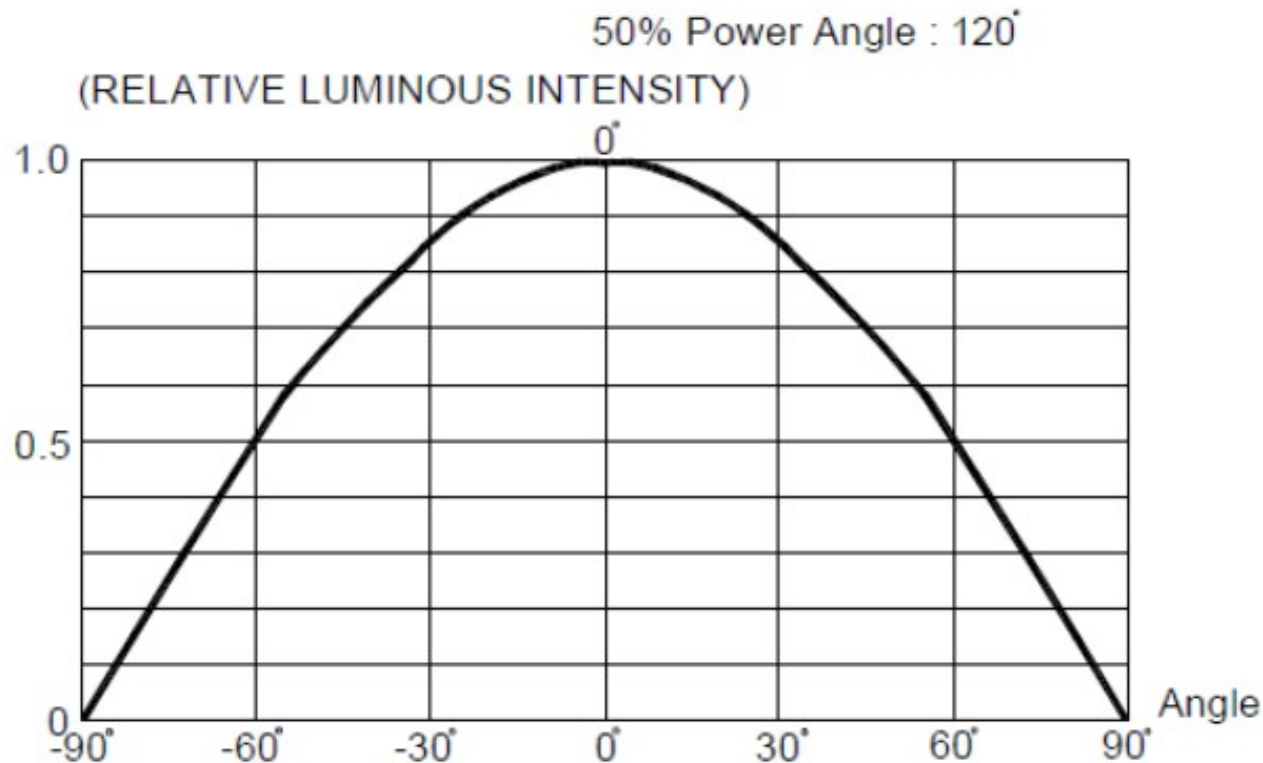


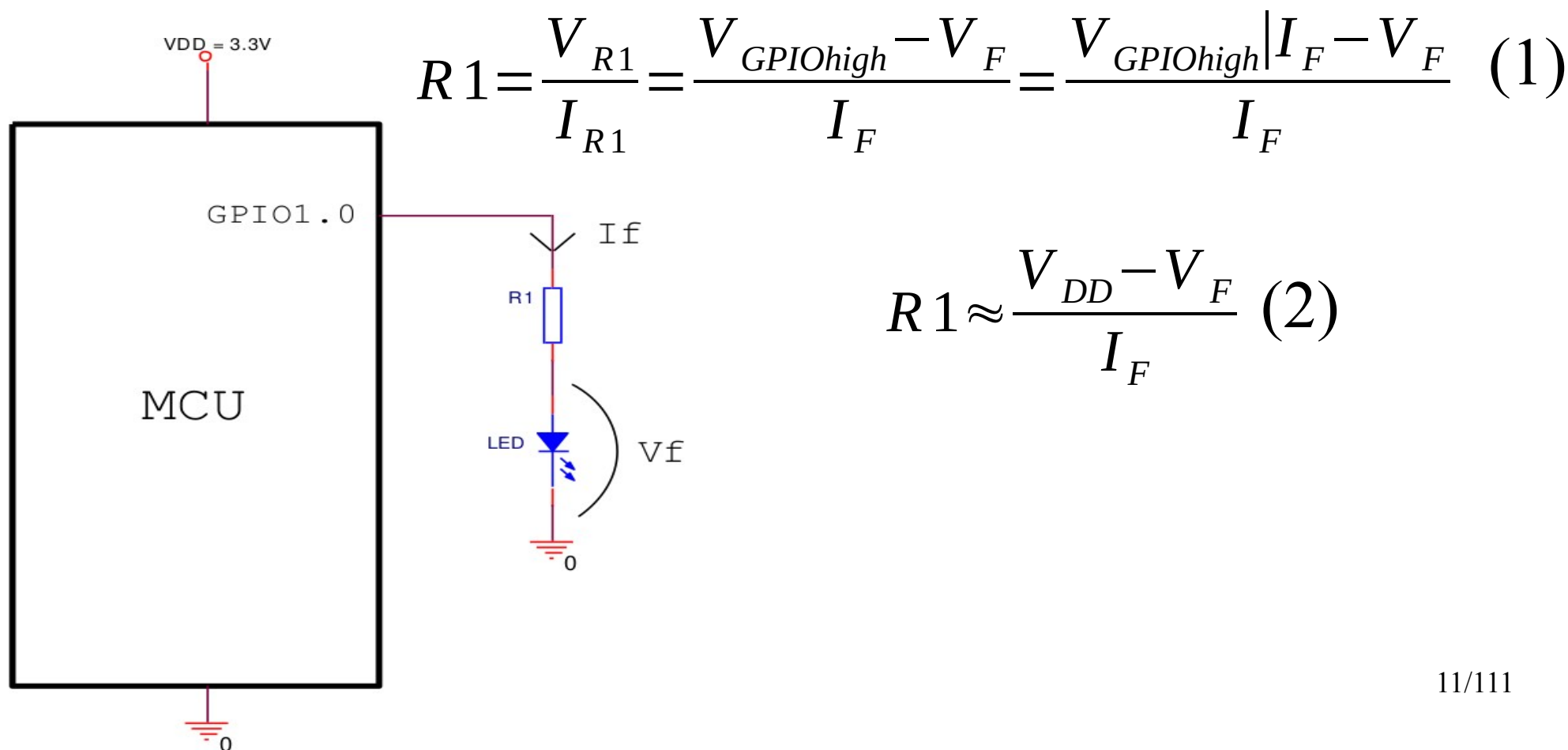
FIG.6 FAR FIELD PATTERN

# Управление на LED индикатори

**ВНИМАНИЕ!** Всеки един от изброените параметри трябва да се провери от техническата спецификация (datasheet) за конкретния модел светодиода, за конкретния производител.

# Управление на LED индикатори

За повечето  $\mu$ CU номиналното захранване е 3.3 V или 5 V. Това означава, че в изхода на GPIO ще има приблизително захранващото напрежение.



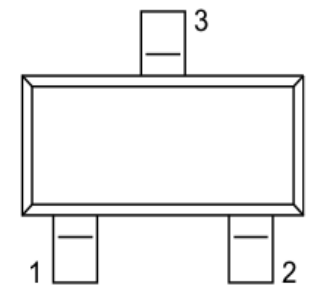
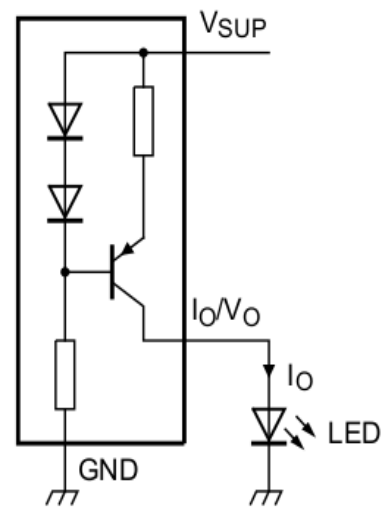
# Управление на LED индикатори

Светодиодите се захранват с генератори на ток. Такива има в интегрално изпълнение.

*Пример* – NCR402T на Nexperia е параметричен, линеен генератор на ток в три-изводен SOT23 корпус.

$$I_F = 17 \div 23 \text{ mA.}$$

**Table 2. Pinning information**

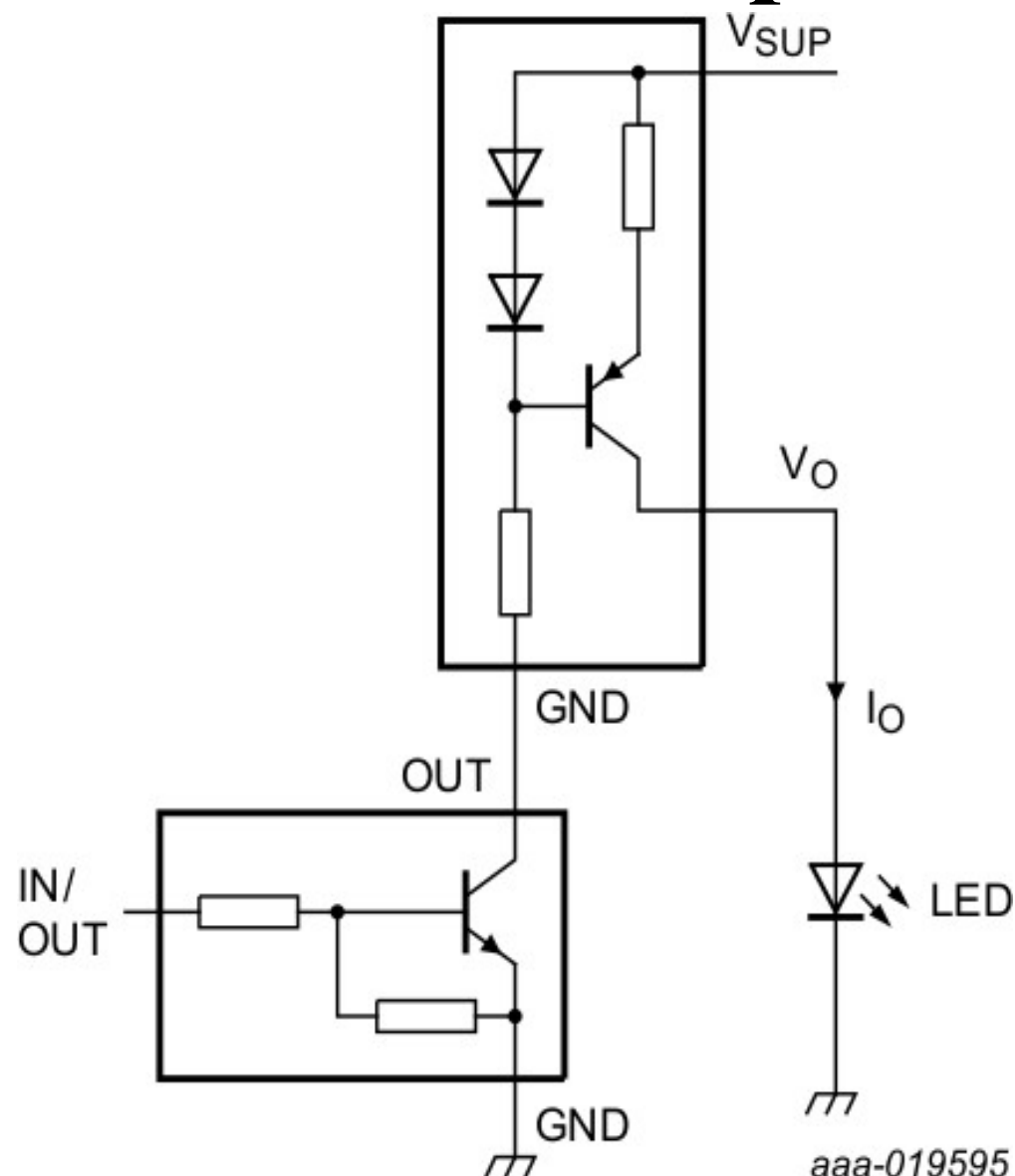
Pin	Symbol	Description	Simplified outline	Graphic symbol
1	GND	ground	 <p>TO-236AB (SOT23)</p>	 <p>aaa-019596</p>
2	V <sub>SUP</sub>	supply voltage		
3	I <sub>O</sub> /V <sub>O</sub>	output current/output voltage		

# Управление на LED индикатори

Захранващото напрежение  $V_{SUP}$  може да варира в широки граници  $5 \div 20\text{ V}$ .

За да стане управляем, генераторът се нуждае от електронен ключ. Така се получава схемата по-долу.

GPIO извода на микроконтролера се свързва към базата “IN/OUT” на цифровия транзистор.



# Управление на LED индикатори

Такова схемно решение може да е подходящо за някои приложения (габаритните LED светлини на автомобил се проектират  $50 \div 70 \text{ mA} / 12 \text{ V}$ ), но да се окаже **твърде скъпо** за други.

Затова схемата, показана преди два слайда, с **токоограничаващия резистор** се използва най-често за индикация на ел. панели. Тази схема разчита на две условия:

- \*напрежението  $V_{DD}$  да е стабилизирано;

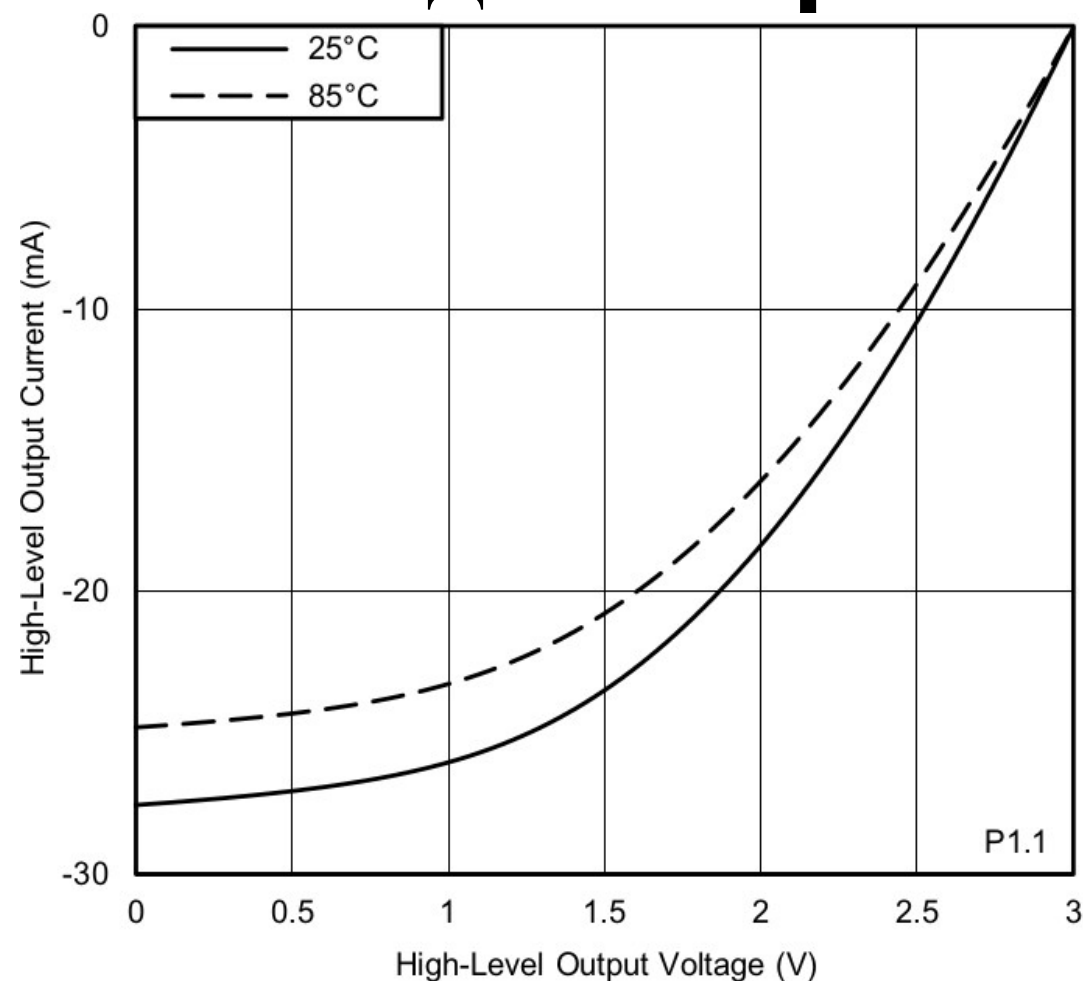
- \*напрежението  $V_F$  да е неизменящо се.

Ако едно от двете не може да бъде гарантирано ( $V_{DD}$  се взима директно от батерия,  $V_F$  се променя от температурата/старееене), трябва да се използва управляем генератор на ток.

# Управление на LED индикатори

Във формула (1) се прави едно допускане, за да се стигне до съкратената формула (2), и това е – приема се, че **високото логическо ниво на GPIO е равно на захранващото напрежение**.

Това, обаче, е **много грубо допускане**. Реално изходната характеристика на CMOS стъпало изглежда така (MSP430FR6989):



$V_{CC} = 3.0 \text{ V}$

Figure 5-12. Typical High-Level Output Current vs High-Level Output Voltage

# Управление на LED индикатори

Тоест може да се окаже, че

$$*V_{\text{GPIOhigh}} = V_{\text{DD}} - 0.5 \text{ V при ток през светодиода } 10 \text{ mA}$$

$$*V_{\text{GPIOhigh}} = V_{\text{DD}} - 1 \text{ V при ток през светодиода } 20 \text{ mA}$$

което прави формула (2) невалидна.

Затоа изходите на  $\mu\text{CU}$  трябва да се **буферират с електронни ключове**. Тогава формула (2) винаги ще важи.

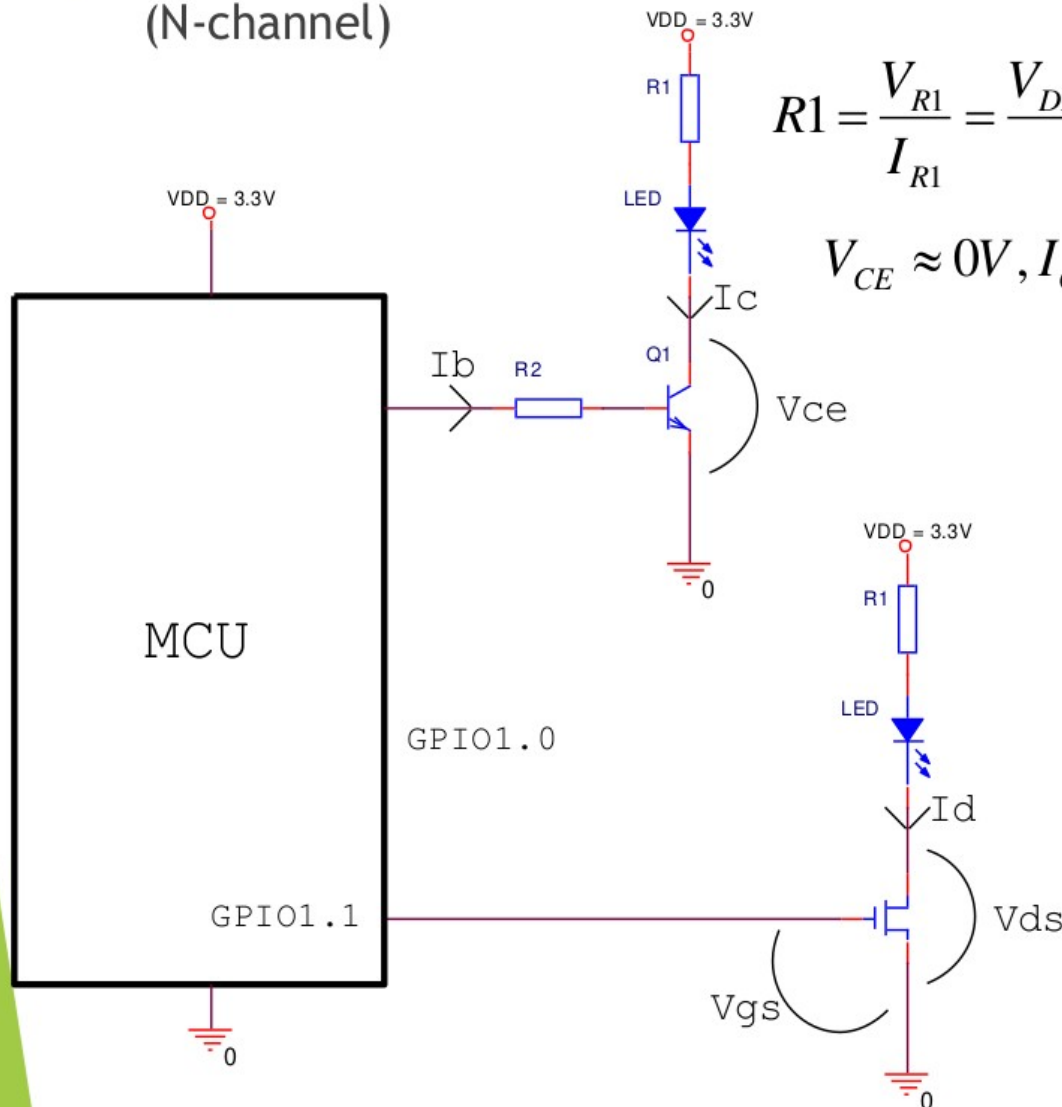
На следващия слайд са показани схеми за буфериране на изходите на  $\mu\text{CU}$  с NPN и NMOS транзистори.

**ВНИМАНИЕ!**  $V_{\text{GStres}} = 1 \div 2 \text{ V}$  за контролери със захранване 3.3 V, иначе може да не отпусти NMOS-а.



# Управление на LED индикатори

- Connecting an LED to a MCU with an electronic switch using bipolar (NPN) and MOS (N-channel)



$$R1 = \frac{V_{R1}}{I_{R1}} = \frac{V_{DD} - V_F - V_{CE}}{I_C} \approx \frac{V_{DD} - V_F}{I_F} \quad R2 = \frac{V_{R2}}{I_{R2}} = \frac{V_{DD} - V_{BE}}{I_B}$$

$$V_{CE} \approx 0V, I_C = I_F$$

$$I_B = \frac{I_C}{h_{FE}} = \frac{I_F}{h_{FE}}$$

$$R1 = \frac{V_{R1}}{I_{R1}} = \frac{V_{DD} - V_F - V_{DS}}{I_D} \approx \frac{V_{DD} - V_F}{I_F}$$

$$V_{DS} \approx 0V, I_D = I_F, V_{GS\_threshold} \leq V_{DD}$$

# Управление на LED индикатори

Интересно схемно решение може да се види в дебъгера на ST Microelectronics ST-Link: с един GPIO извод се управляват два светодиода.

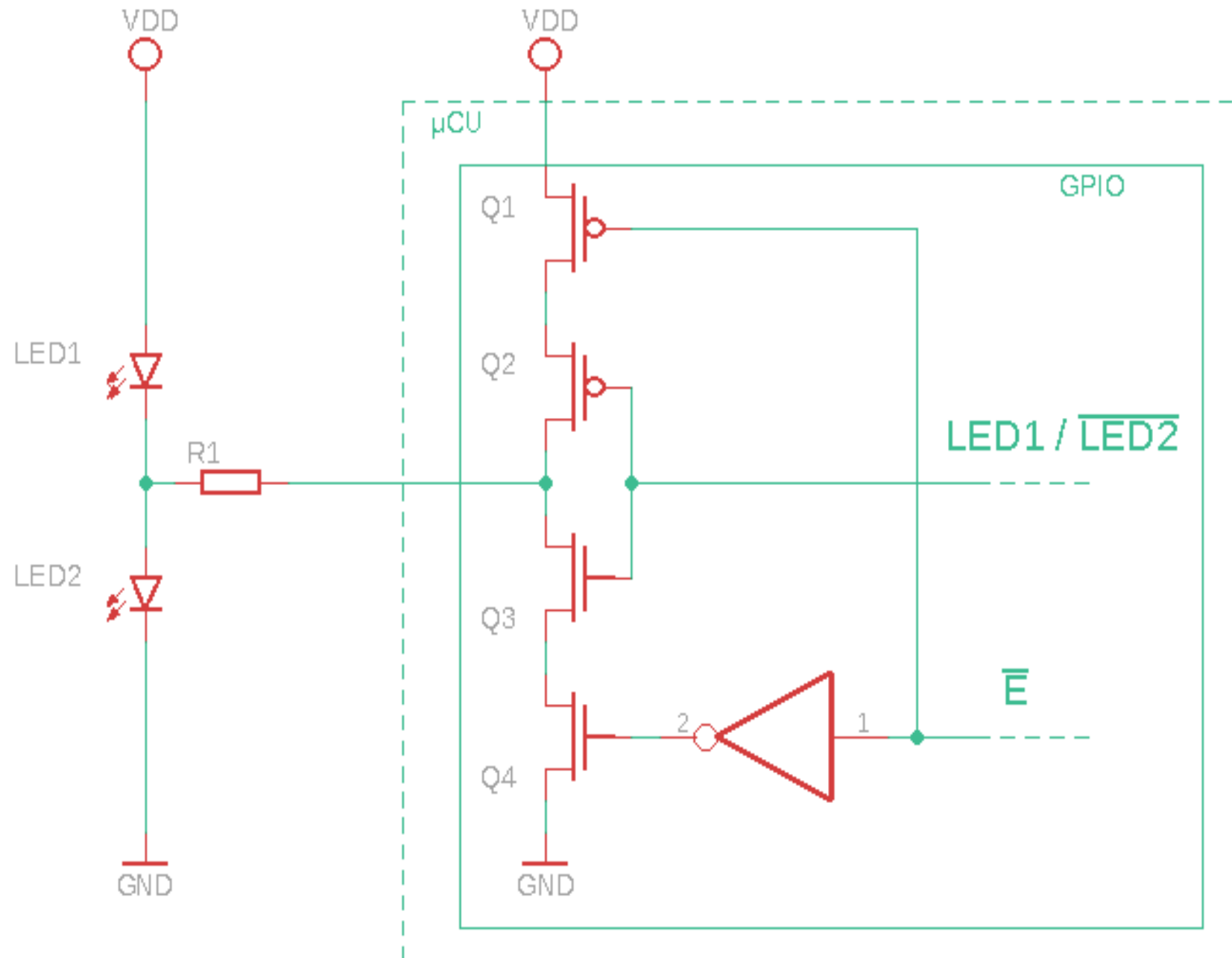
Когато !E = 0 се пуска светодиод, който е избран чрез сигнала LED1/!LED2. Когато Q3 и Q4 са отпушени, LED1 свети, LED2 е шунтиран. Когато Q1 и Q2 са отпушени, LED2 свети, LED1 е шунтиран.

Когато !E = 1, GPIO изводът е конфигуриран като вход и

$$V_{F1} + V_{F2} > VDD,$$

следователно и двата LED са изгасени.

# Управление на LED индикатори



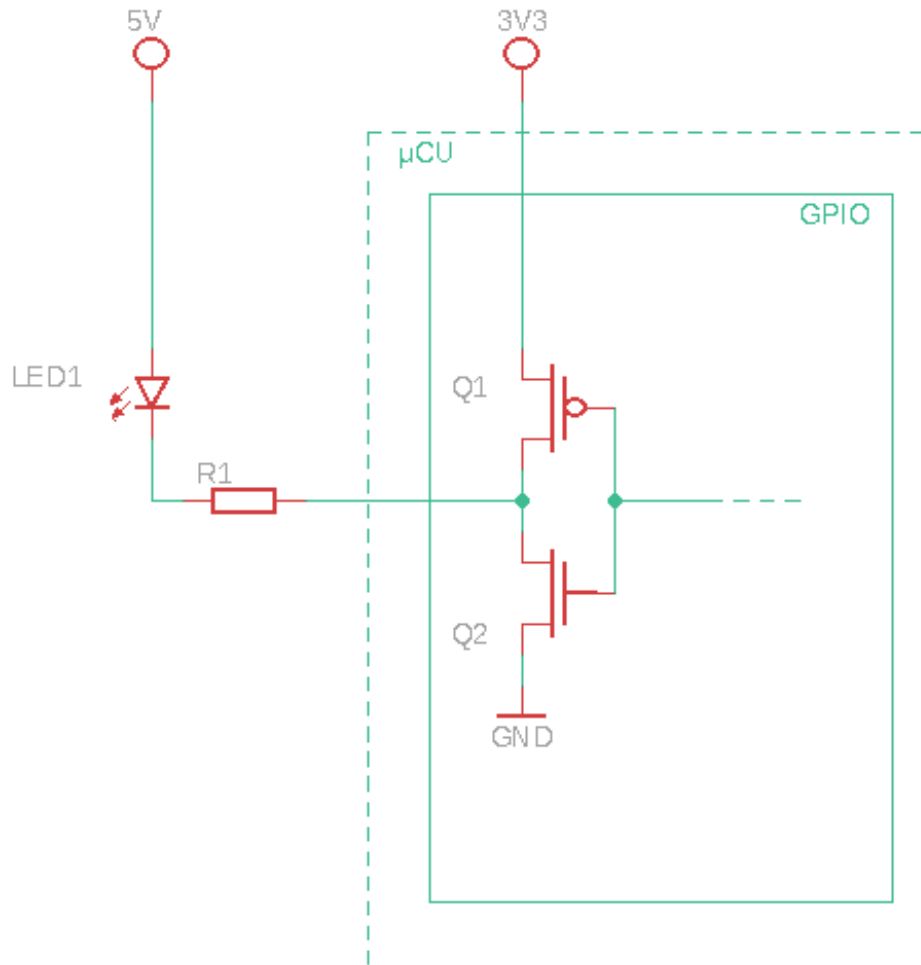
# Управление на LED индикатори

Фундаментална грешка може да се допусне с ярък светодиод с голям пад, например с цвят зелено. Схемата вляво използва противотактно изходно стъпало (push-pull) и разчита, че при  $\text{GPIO}_{\text{high}} = 3.3 \text{ V}$ , а падът  $5 - 3.3 = 1.7 \text{ V}$  няма да е достатъчен, за да отпусти диода. Всъщност, при ярките светодиоди светлина може да се види и при  $10 \div 100 \mu\text{A}$ , т.е.  $1.7 \text{ V}$  е в началото на ВАХ, но ток все пак ще протече и е **възможно диода да свети слабо**, когато уж трябва да е изключен.

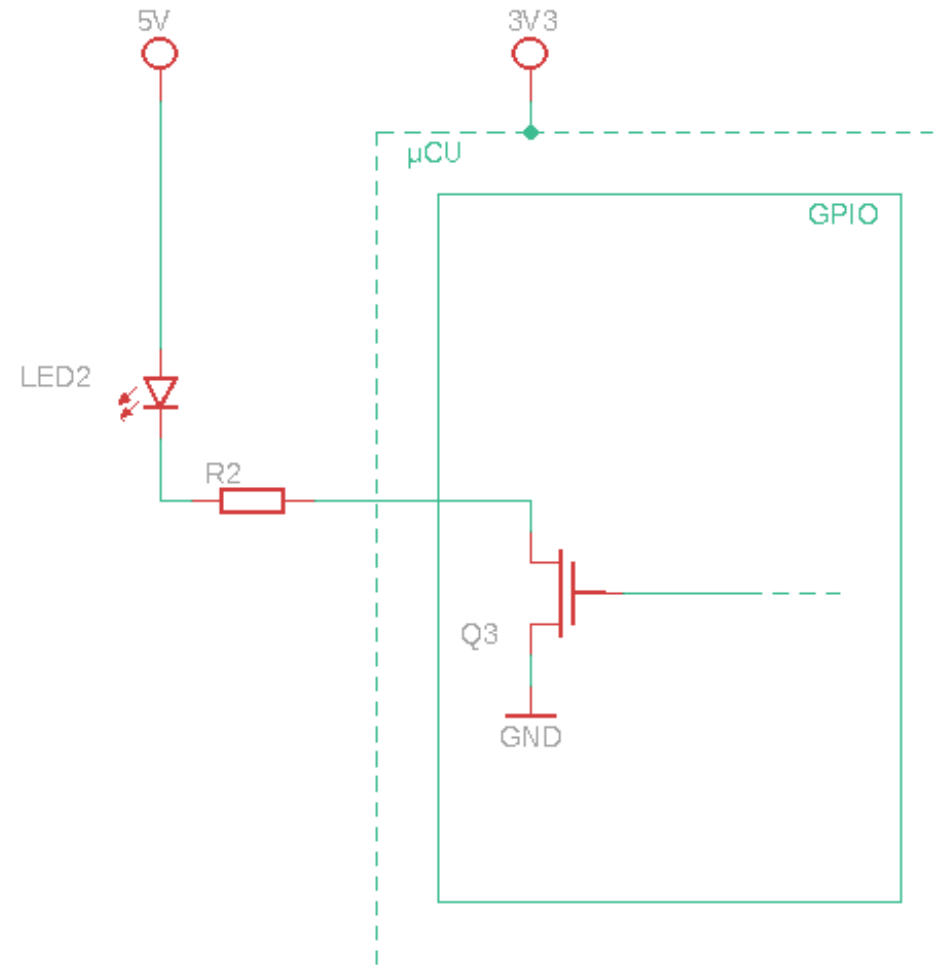
За да не се случва това, **трябва да се използва стъпало по схема отворен-дрейн.**

# Управление на LED индикатори

НЕ!!!



ДА



# Управление на LED индикатори

На пазара съществуват интегрирани в един корпус светодиоди с различни цветове. Най-често това са **дву- и трицветни светодиоди**.

Ако бъдат свързани към изходите на един таймер, **чрез ШИМ може да се изменя цвета на светодиода**, увеличавайки и намалявайки коефициента на запълване на всеки един цвят поотделно.

За да се получат всички видими цветове, трябва да се използва светодиод с интегрирани:

- \*червено (R)

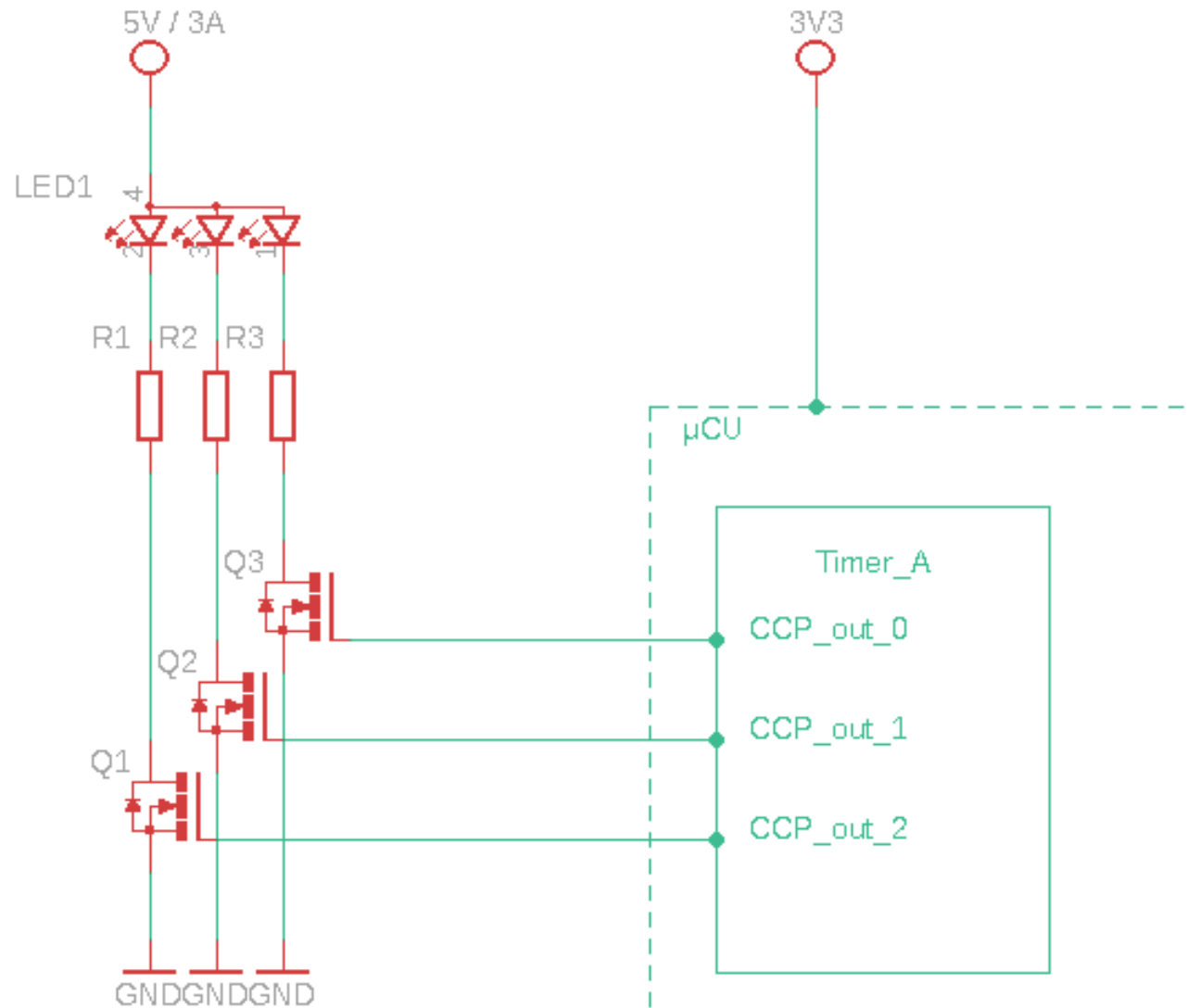
- \*зелено (G)

- \*синьо (B)

Тогава, за да се получи бял цвят, трябва да се зададе коеф. на запълване 100 % и на трите цвята.

# Управление на LED индикатори

Управление на мощен RGB светодиод. Чрез използване на таймер в ШИМ режим, може да се направи лампа, чийто цвят се задава програмно.



# Управление на LED индикатори

7-сегментните светодиодни индикатори могат да изобразяват цифри.

14-сегментните светодиодни индикатори могат да изобразяват цифри и букви[1].

В зависимост от това дали всеки сегмент се управлява с отделен сигнал, или съответните сегменти са свързани в паралел и в различни периоди от време се пускат само отделни сегменти, казва се че има два вида управление:

- \*статична индикация;

- \*динамична.



# Управление на LED индикатори

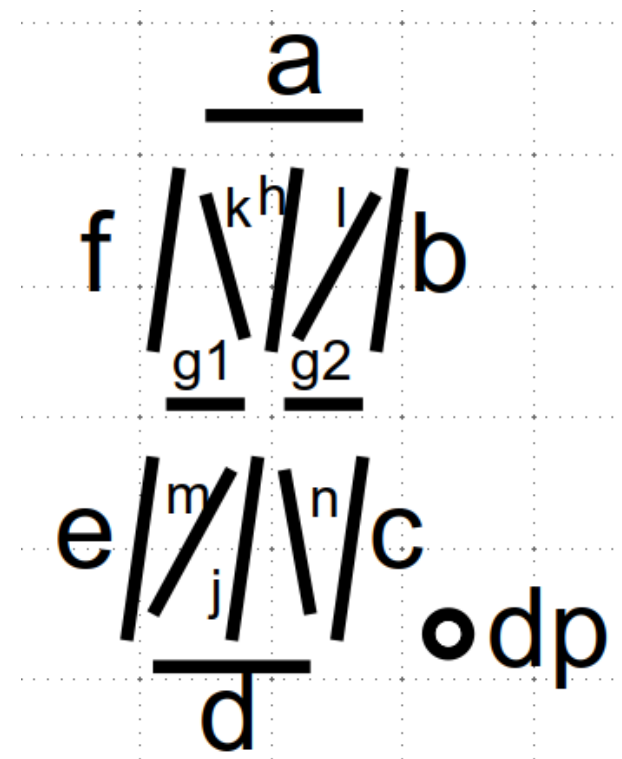
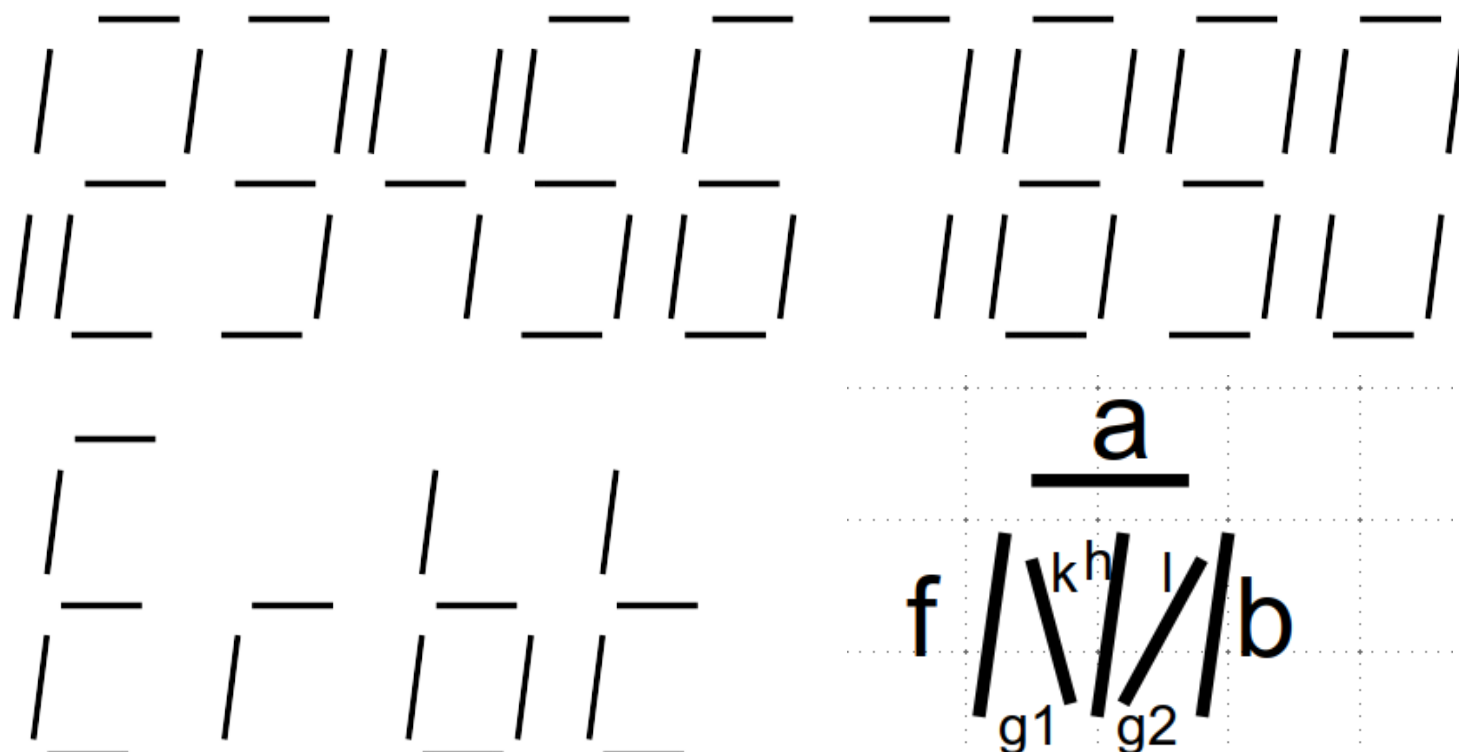
**Статична индикация** – за всеки сегмент от всеки индикатор има по един управляващ сигнал от GPIO порта на  $\mu$ CU.

*Предимство* - във всеки един момент от времето на индикаторите се изобразяват зададените цифри. Ако се прави снимка или се снима видео на табло, индикацията ще бъде винаги видима.

*Недостатък* – необходимите изводи на  $\mu$ CU растат пропорционално на броя на индикаторните елементи. Един индикатор ще заеме 8 извода, 2  $\rightarrow$  16 извода, 3  $\rightarrow$  24 извода, 4  $\rightarrow$  32 извода, 8  $\rightarrow$  64 извода и т.н.

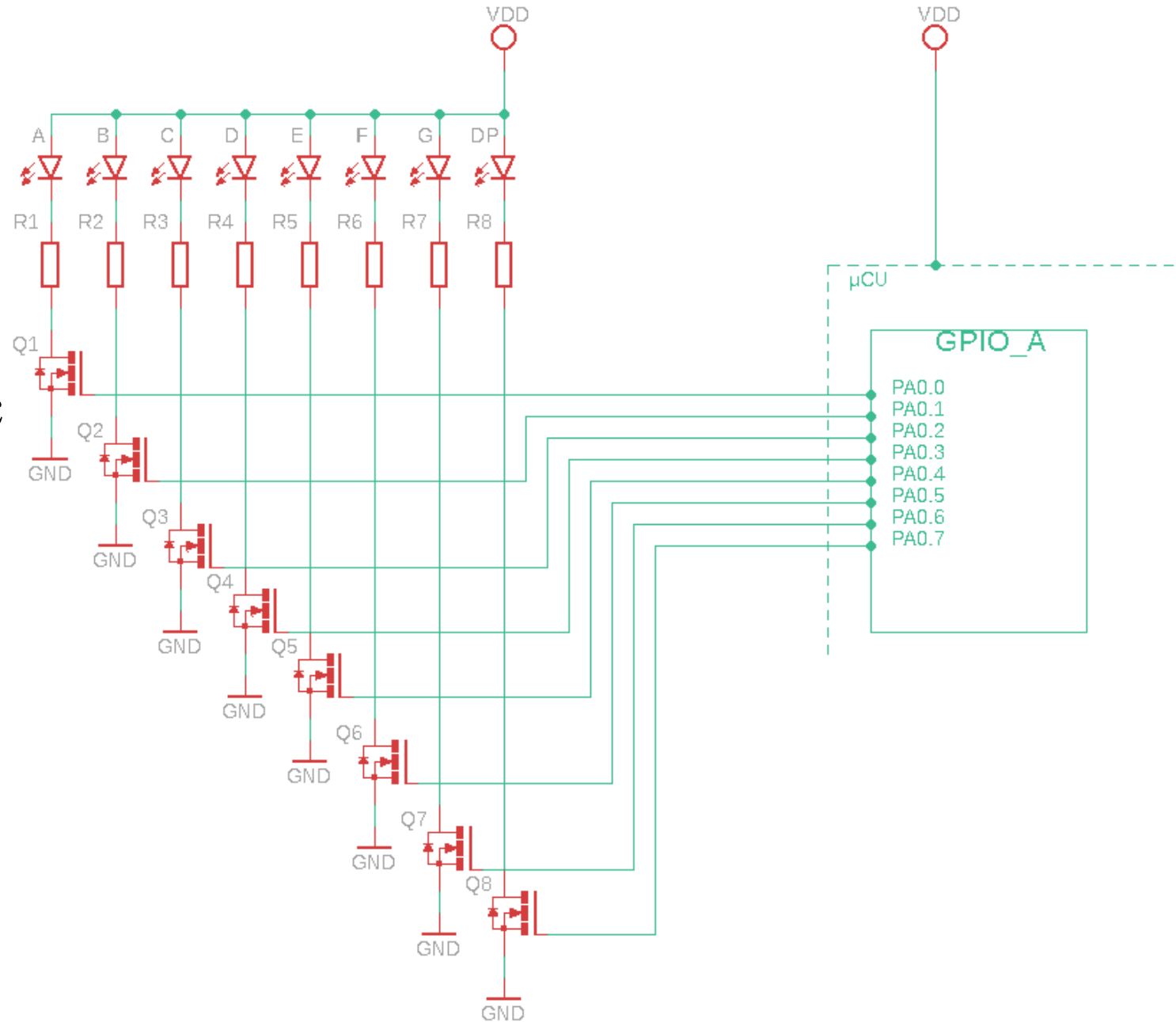
# Управление на LED индикатори

$\begin{array}{c} \text{a} \\ \text{f} / \text{g} / \text{b} \\ \text{e} / \text{d} / \text{c} \bullet \text{dp} \end{array}$



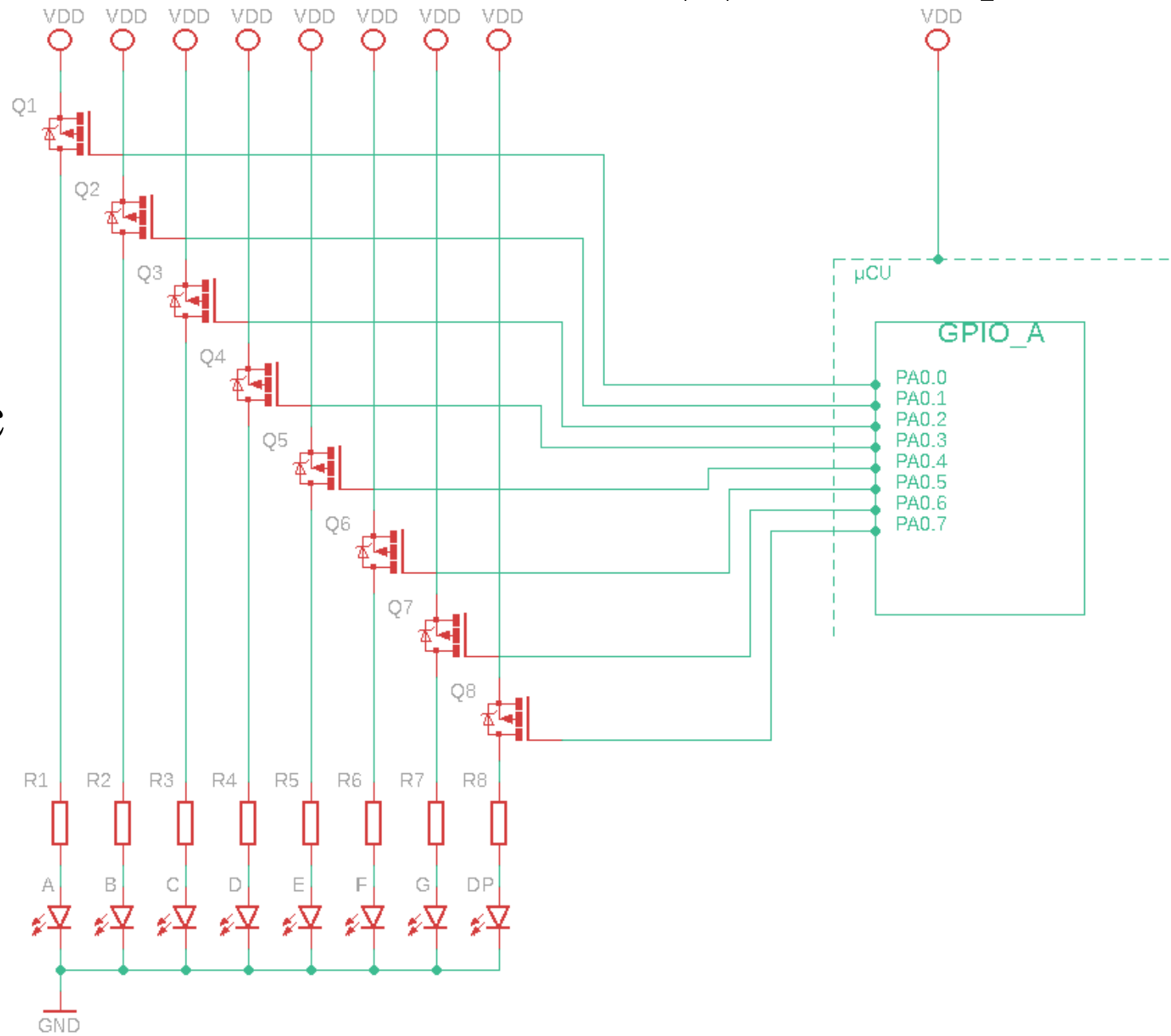
# Управление на LED индикатори

Статична  
индикация,  
1x7-сегментен  
индикатор,  
общ анод,  
ел. ключове с  
MOSFET.



# Управление на LED индикатори

Статична  
индикация,  
1x7-сегментен  
индикатор,  
общ катод,  
ел. ключове с  
MOSFET.



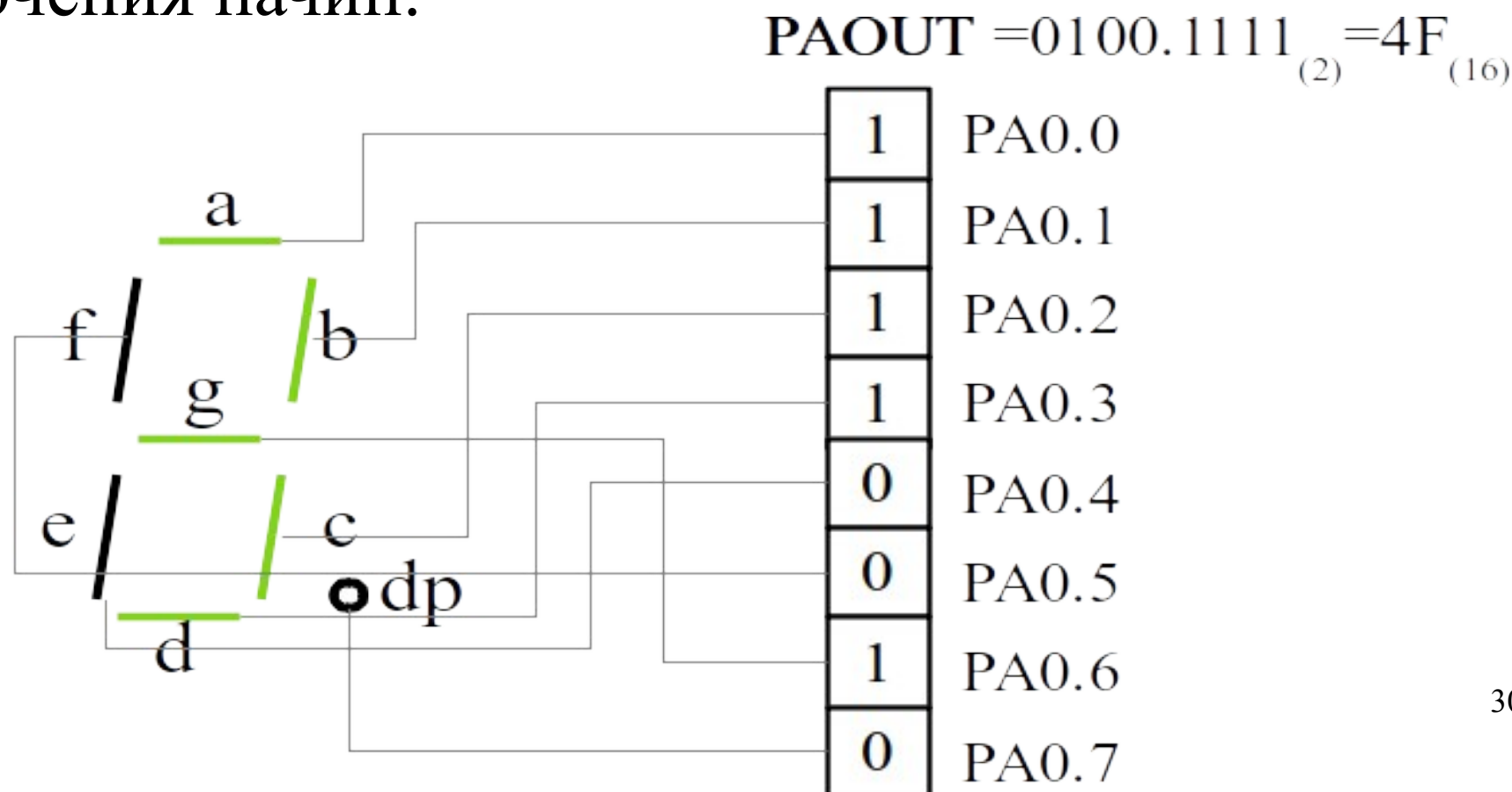
# Управление на LED индикатори

**ASCII таблица** — таблица, която показва връзката на всяка буква (char) в езика C и ѝ съответстващото шестнадесетично число, което се записва в паметта на  $\mu$ CU.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	\$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

# Управление на LED индикатори

*Пример* – за да се изобрази цифрата 3 на индикатора, в изходния регистър на GPIO порта, който се казва PAOUT, трябва да се запише числото 0x4F, ако управляващите сигнали са свързани към сегментите по посочения начин.



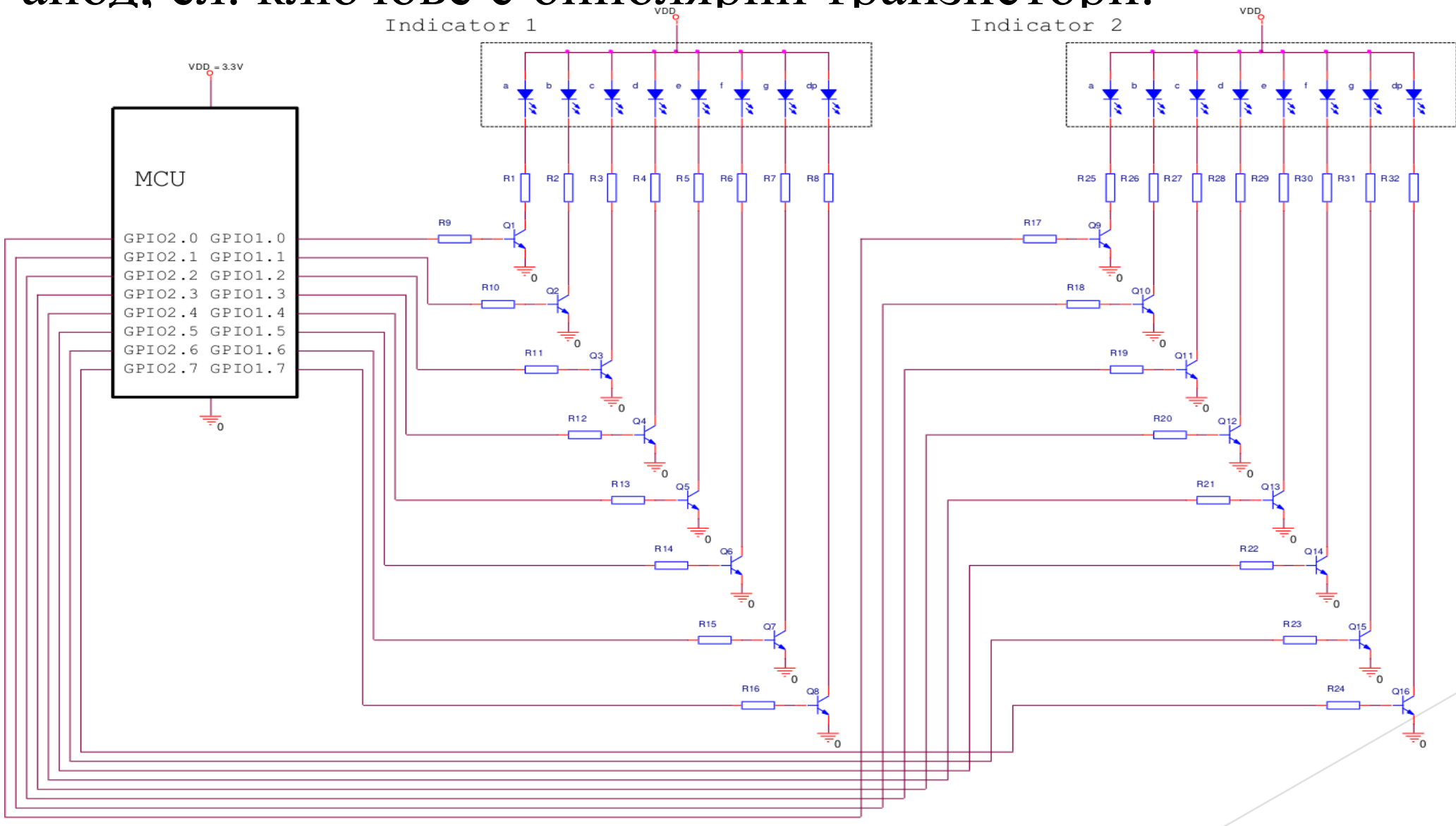
# Управление на LED индикатори

*Пример* – използване на статична индикация и таблица на съответствието в LPC845.

```
const uint8_t digit_map_char[12] = {  
    0x3F, //0 (0)  
    0x06, //1 (1)  
    0x5B, //2 (2)  
    0x4F, //3 (3)  
    0x66, //4 (4)  
    0x6D, //5 (5)  
    0x7D, //6 (6)  
    0x07, //7 (7)  
    0x7F, //8 (8)  
    0x6F, //9 (9)  
    0x40, //- (10)  
    0x00, //*space* (11)  
};  
  
uint8_t char_to_encoded_number(char ch){  
    uint8_t digit;  
    uint8_t encoded_num = 0;  
  
    if(ch >= '0' && ch <= '9'){  
        digit = ch - 0x30;  
  
        if(digit < 12){  
            encoded_num = digit_map_char[digit];  
        }  
    }  
  
    if(ch == ' '){  
        encoded_num = digit_map_char[11];  
    }  
  
    return encoded_num;  
}
```

# Управление на LED индикатори

Статична индикация, 2x7-сегментни индикатори, общ анод, ел. ключове с биполярни транзистори.





# Управление на LED индикатори

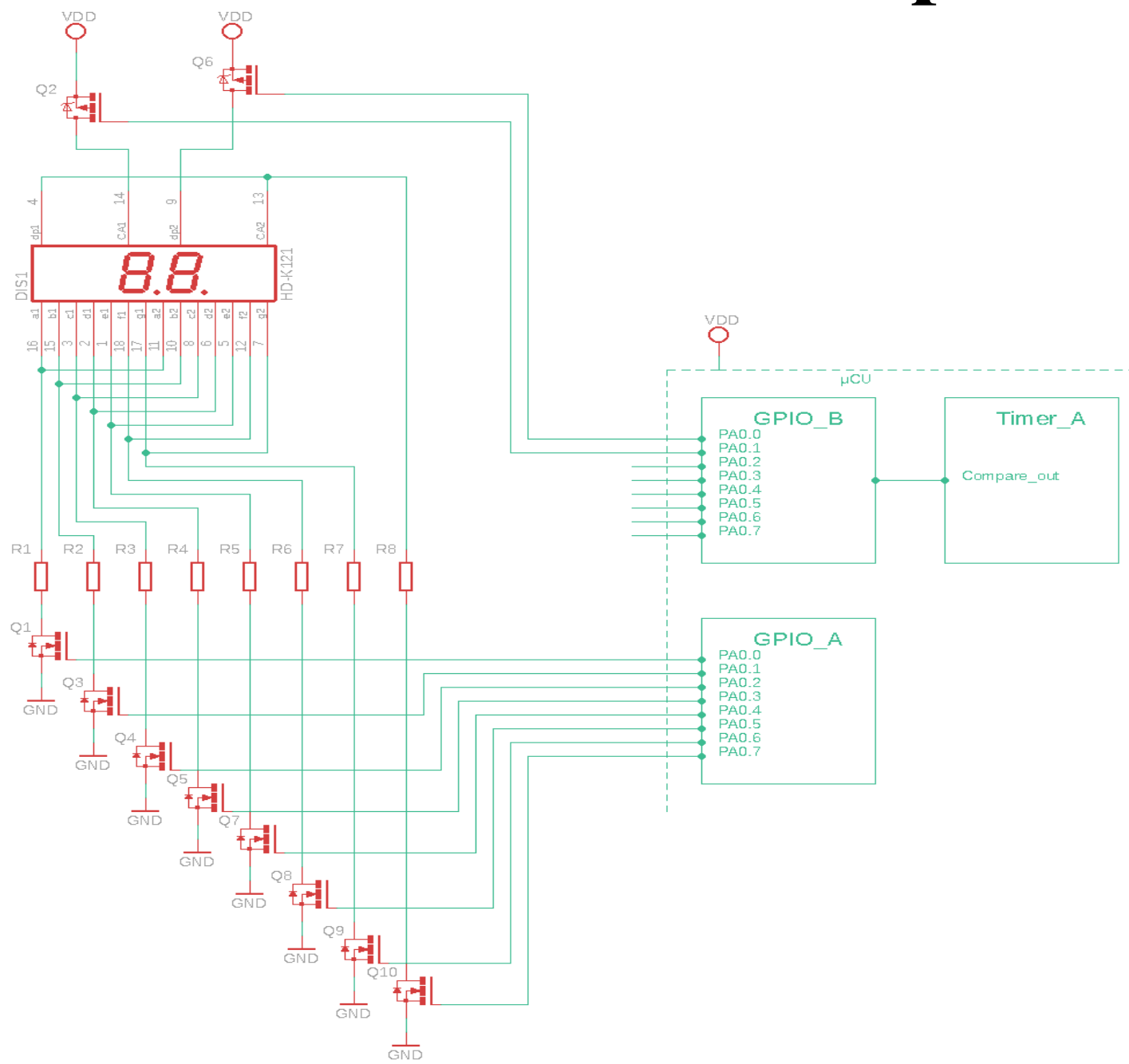
**Динамична индикация** – съответстващите сегменти на различните индикатори се свързват в паралел и има допълнителни сигнали за избор на индикатор. Тези сигнали се превключват във времето много бързо, така че човешкото око да не забележи и да вижда псевдо-едновременно всички цифри, на всички индикатори.

*Предимство* – броят на управляващите сигнали е по-малък от този на статичната индикация. Един индикатор ще заеме 8 извода,  $2 \rightarrow 10$  извода,  $3 \rightarrow 11$  извода,  $4 \rightarrow 12$  извода,  $8 \rightarrow 16$  извода и т.н.

*Недостатък* – в даден момент от времето се изобразява само една цифра и ако панела се заснеме с фотоапарат или видеокамера, ще се виждат само някои от цифрите.

# Управление на LED индикатори

Динамична индикация,  
2x7-сегментен индикатор,  
общ анод,  
ел. ключове с MOSFET.  
Токът през Q1 и Q5 е 8 пъти по-голям от останалите транзистори.



# Управление на LED индикатори

*Пример* - използване на динамична индикация и таблица на съответствието в LPC845 за обхождане на 8 индикатора.

Функцията `segm_led_callback` се извиква периодично на всяка 1 ms.

Променливата `digit` съдържа номера на индикаторния елемент, който в настоящия квант от време е активен.

За сегментите и за избор на индикатор се използват 2 преместващи регистъра, които намаляват броя на използваните изводи от 16 на 3.

# Управление на LED индикатори

```
void segm_led_callback(void){
    uint8_t encoded_num;
    uint8_t dot = 0;
    static uint8_t digit = 1;

    if(digit < 5){
        encoded_num = char_to_encoded_number(line_1_buff[digit-1]);

        if(line_1_dot_position == digit){
            dot = 1;
        }
        segm_led_show_digit(1, digit, encoded_num, dot);
    }
    else{
        encoded_num = char_to_encoded_number(line_2_buff[digit-5]);

        if(line_2_dot_position == (digit-4)){
            dot = 1;
        }
        segm_led_show_digit(2, digit-4, encoded_num, dot);
    }
    digit++;

    if(digit > 8){
        digit = 1;
    }
}
```

//12.34  
//5.678  
**char** line\_1\_buff[5] = {'1', '2', '3', '4'};  
**uint8\_t** line\_1\_dot\_position = 2;  
**char** line\_2\_buff[5] = {'5', '6', '7', '8'};  
**uint8\_t** line\_2\_dot\_position = 1; 36/111

# Управление на LED индикатори

```
void segm_led_show_digit(
uint8_t line_number,
uint8_t digit_position,
uint8_t segments,
uint8_t dot
){
    uint8_t tx_buff[2] = { 0x00, 0xFF };

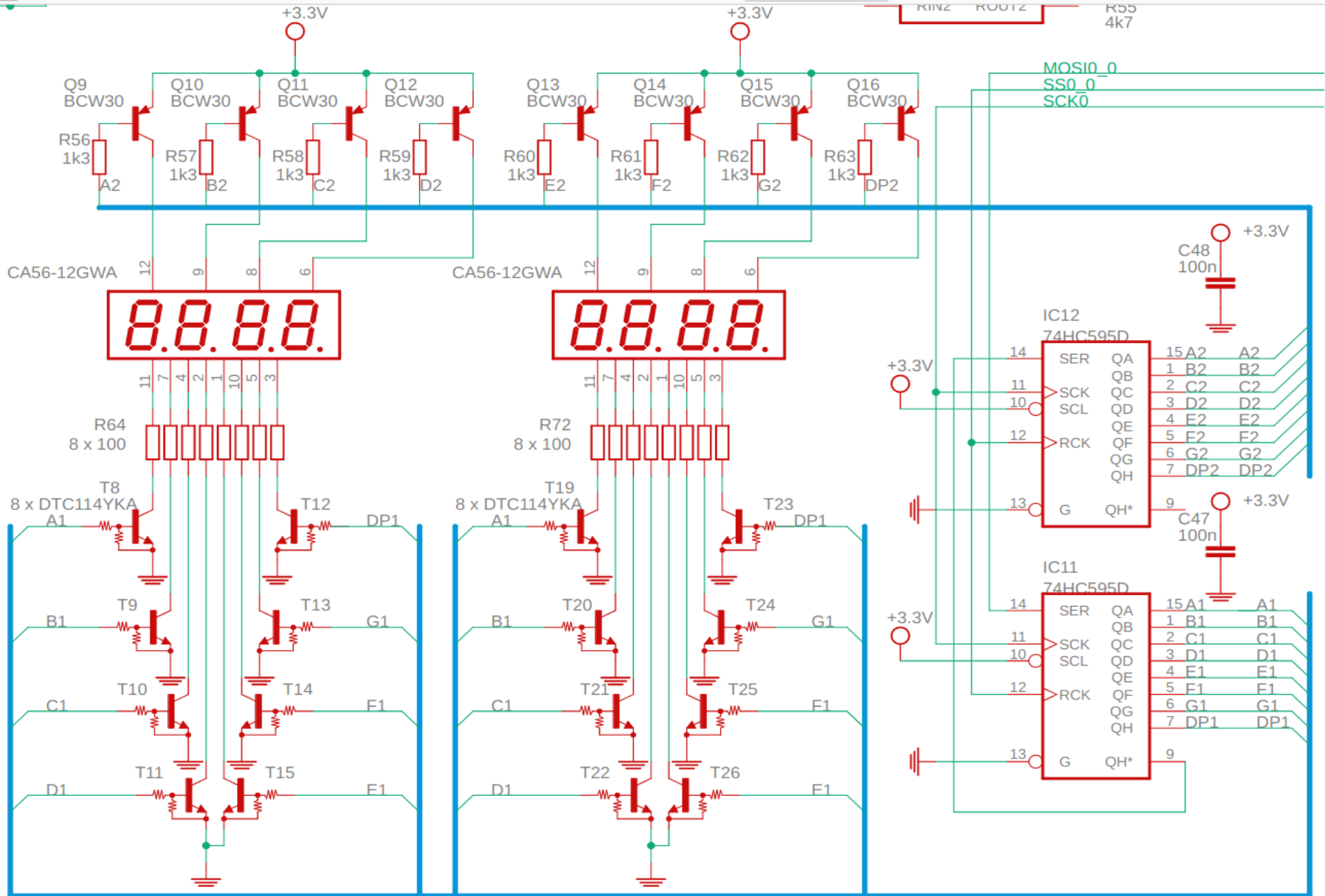
    switch(line_number){
    case 1:
        switch(digit_position){
        case 1:
            tx_buff[1] &= ~BIT0;
            break;
        case 2:
            tx_buff[1] &= ~BIT1;
            break;
        case 3:
            tx_buff[1] &= ~BIT2;
            break;
        case 4:
            tx_buff[1] &= ~BIT3;
            break;
        }
        break;
    case 2:
        switch(digit_position){
        case 1:
            tx_buff[1] &= ~BIT4;
            break;
        case 2:
            tx_buff[1] &= ~BIT5;
            break;
        case 3:
            tx_buff[1] &= ~BIT6;
            break;
        case 4:
            tx_buff[1] &= ~BIT7; }
            break;
        }
        break;
    }

    if(dot){
        segments |= 0x80;
    }

    tx_buff[0] = segments;

    while(pps_flags.spi_busy){ }
    spi_busy = 1;
    GPIO_PinWrite(GPIO, 1, 7, 0);
    spi_write_half_word(tx_buff);
    GPIO_PinWrite(GPIO, 1, 7, 1);
    spi_busy = 0;
}
```

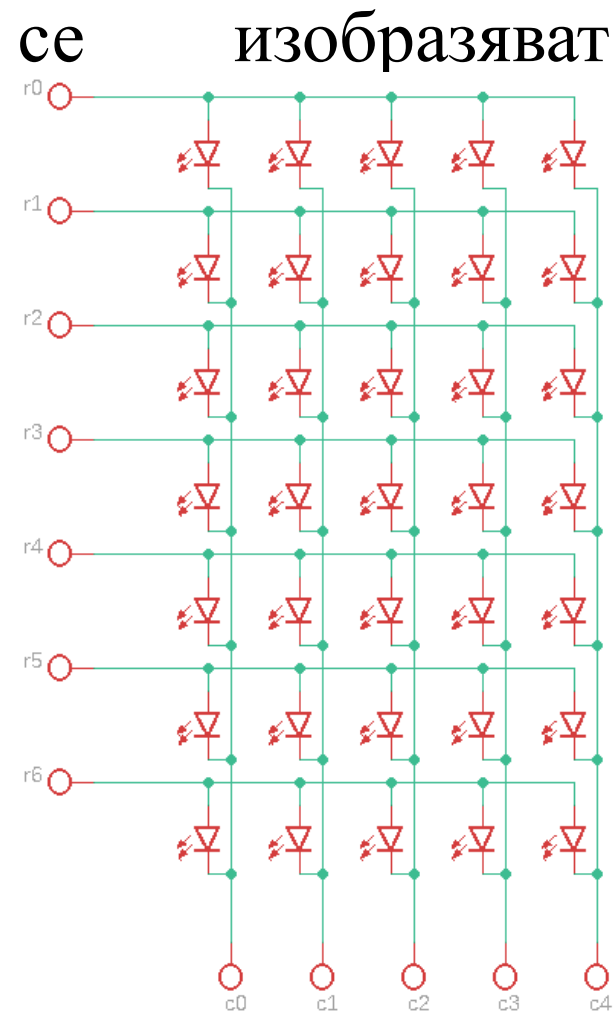
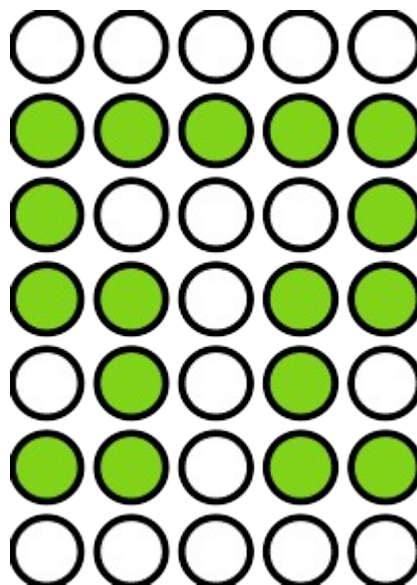
# Управление на LED индикатори



# Управление на LED индикатори

Управление на **LED матрица** (LED dot matrix)– светодиоди /едноцветни или многоцветни/ се разполагат един до друг в пластмасов корпус. Подреждат се в квадрат или правоъгълник. С тяхна помощ може да се изобразяват букви/цифри/специални символи.

Анодите на светодиодите от всеки ред са свързани накъсо, също и катодите от всяка колона. Задължително се използва динамична индикация.

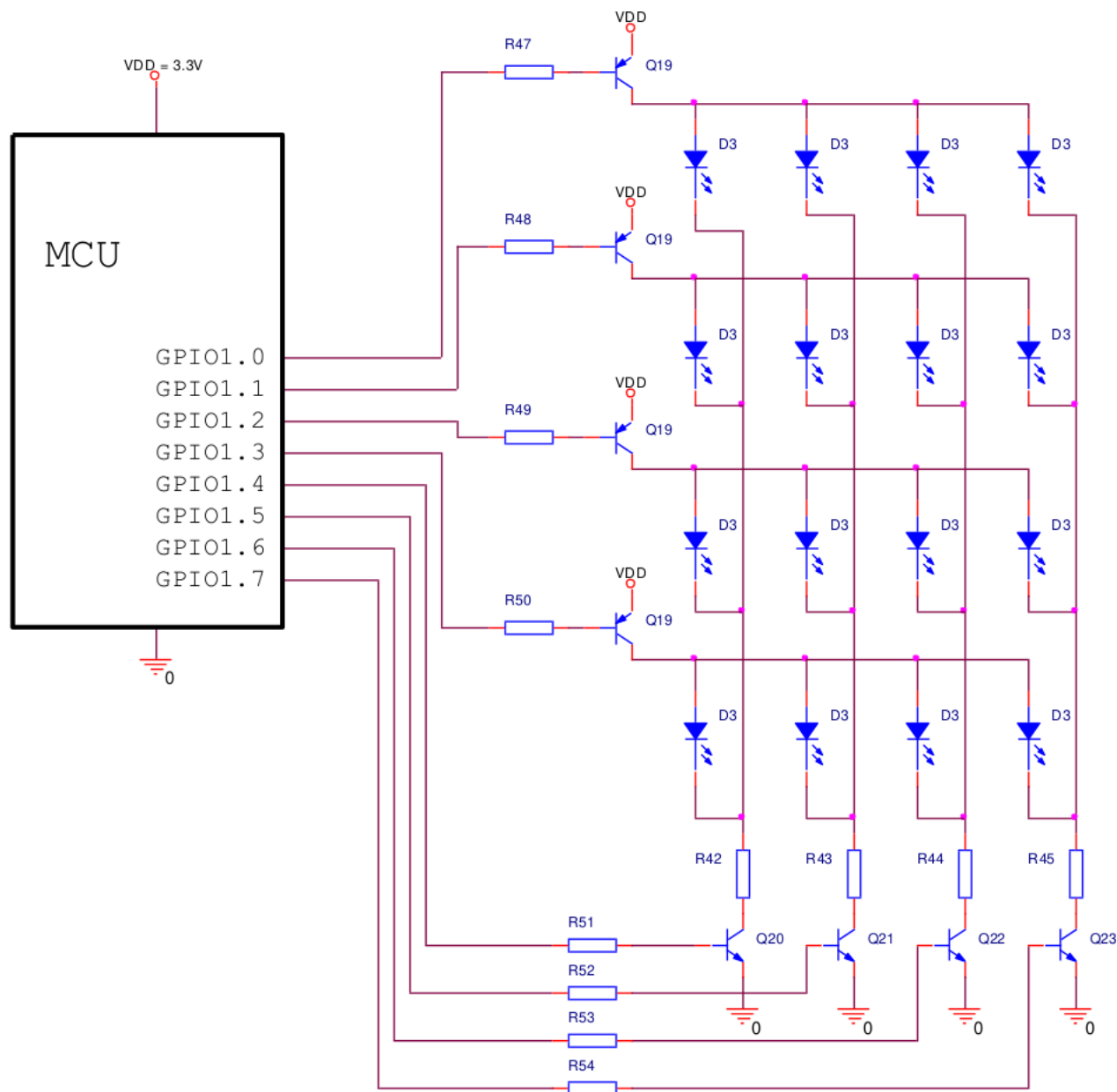


# Управление на LED индикатори

Динамична индикация, матричен светодиоден индикатор, ел. ключове с биполярни транзистори.

Транзисторите на редовете трябва да издържат тока на всички пиксели, докато тр. на колоните – само на 1 светодиод.

Обхождането (опресняването) става по редове.





# Управление на LED индикатори

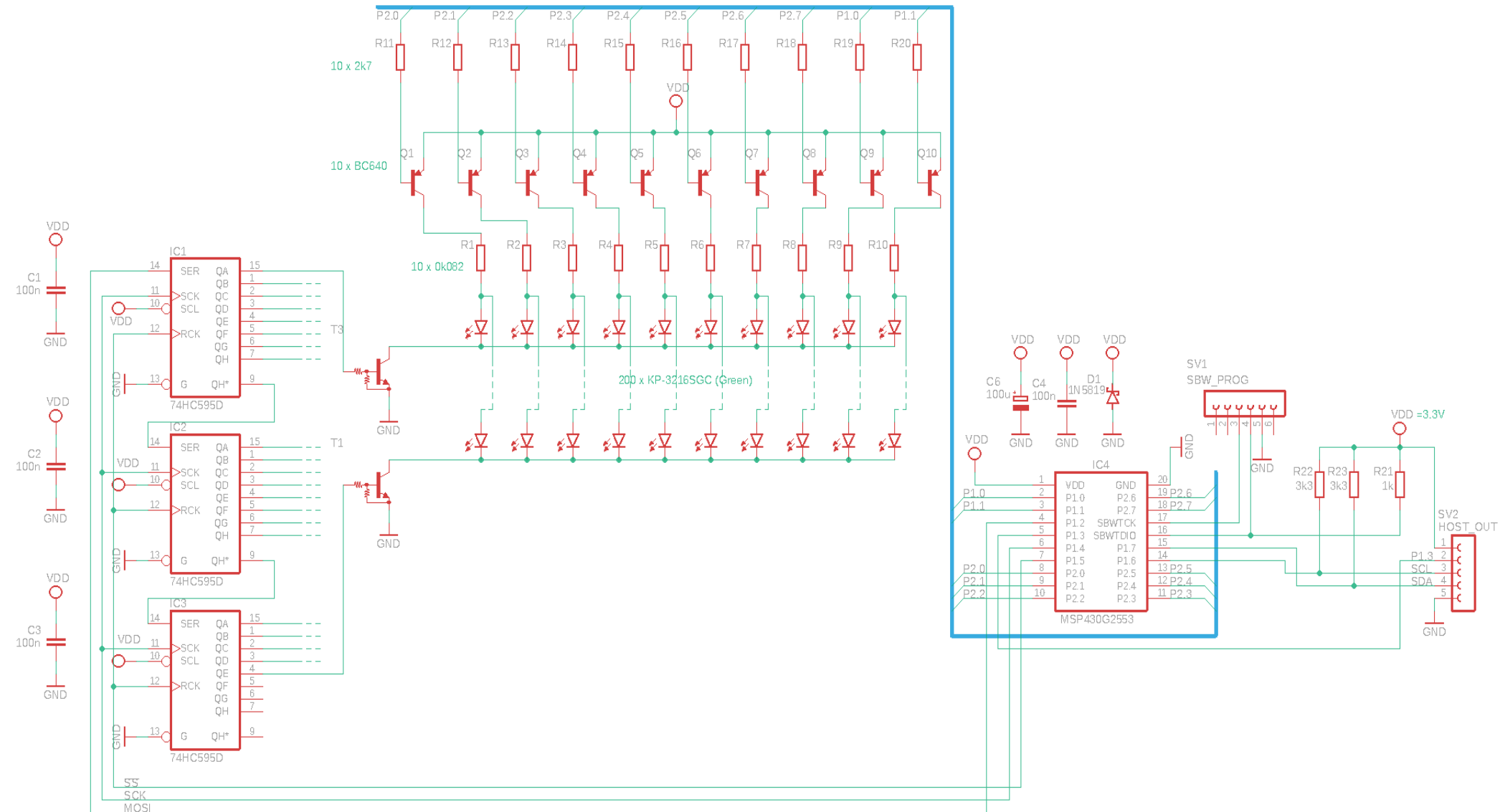
*Пример* – управление на зелена светодиодна матрица с резолюция 10x20. Изображението се зарежда по I2C интерфейс. Управлението е реализирано с MSP430G2553.

По I2C се приемат 2 байта, от които само 10 бита се използват. Всеки бит отговаря на един светодиод от съответния ред.

Таймер реализира обхождането на редовете. Когато контролера на дисплея е готов да приеме данни за нов ред, изработва синхро-сигнал на P1.3. Тогава главното устройство (HOST) трябва да изпрати данните за следващия ред и т.н.

Показано е свързване общ катод на всеки ред / общ анод на всяка колона. Обхождането е ред по ред.

# Управление на LED индикатори



# Управление на LED индикатори

```
int main(void){
    uint8_t i;
    uint16_t row_data;

    WDTCTL = WDTPW + WDT HOLD;

    init();

    while(1){
        set_sync(1);
        for(i = 0; i < 20; i++){
            row_data = i2c_recv[i][0];
            row_data |= (i2c_recv[i][1]<<8);
            draw_row(i, row_data);
            __bis_SR_register(CPUOFF + GIE); //Awaken by timer & i2c interrupt
        }
        set_sync(0);
        __delay_cycles(1000000);
    }
}
```

# Управление на LCD дисплеи

**Течен кристал** — вещество в течно и твърдо състояние, което има свойствата и на течност, и на твърдо вещество [4].

Електрически ток може да окаже влияние на течния кристал. В зависимост от напрежението, предизвикало този ток, молекулите на кристала се завъртат под определен ъгъл.

В следствие на ъгъла, на който са ориентирани молекулите, светлината от околността може или да минава през тях, или да не минава.

# Управление на LCD дисплеи

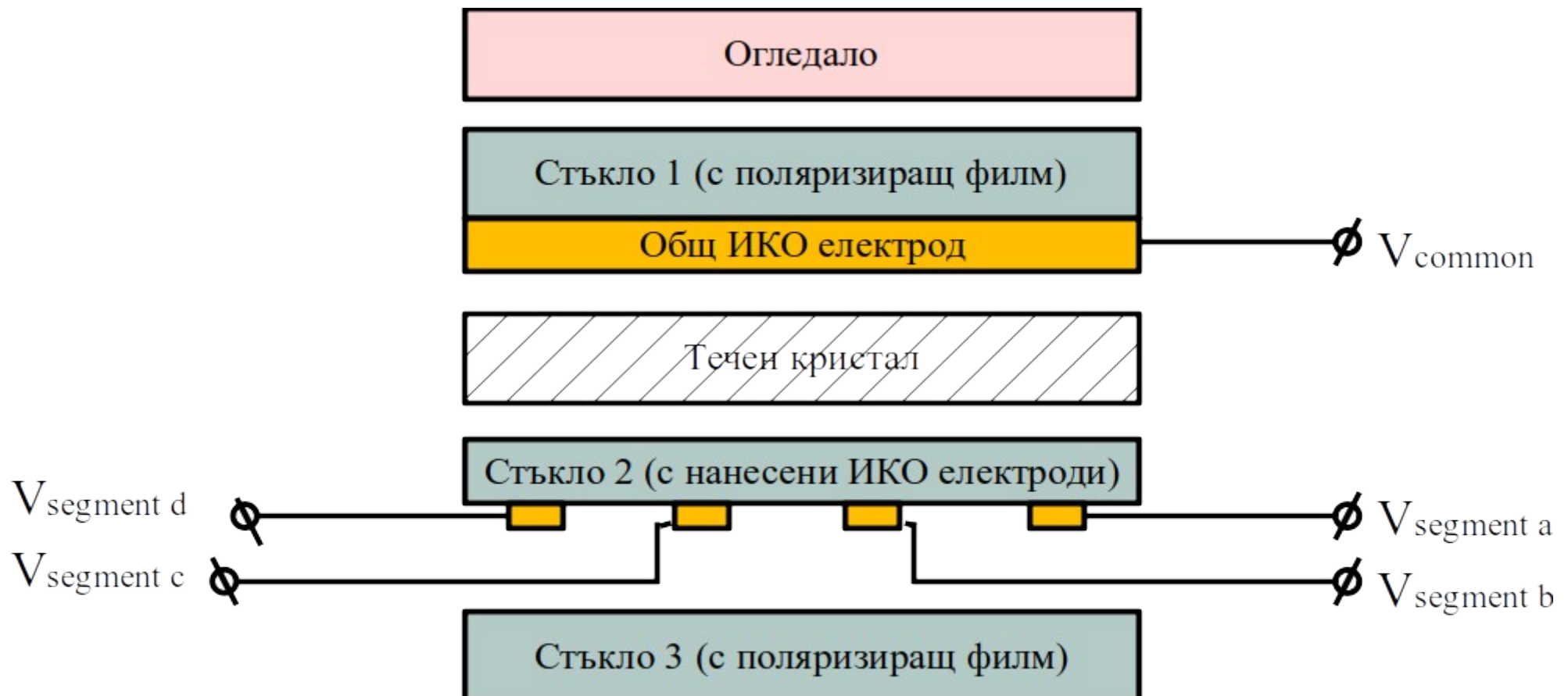
Електроди от индиево-калаен оксид, ИКО, (Indium Tin Oxide, ITO) се оформят в сегмент от цифра/буква или пиксел. Те се отлагат на стъклена подложка, зад която има огледало.

**ИКО е прозрачен метал.**

**\*Ако не е приложено напрежение**, околната светлина ще мине през стъклото и ще се отрази от огледалото. Течният кристал под ИКО няма да се вижда.

**\*Ако се приложи напрежение** между ИКО електрод и общ електрод, нанесен върху огледалото, молекулите на течния кристал под ИКО ще се преориентират и няма да пропускат вече светлина. **Течният кристал ще се вижда като черно петно** под прозрачния метал.

# Управление на LCD дисплеи



# Управление на LCD дисплеи

**LCD** дисплеите се управляват с променливо напрежение ( $30 \div 100$  Hz). Подаването на постоянно напрежение е възможно, но ще съкрати живота на течния кристал и ще понижи качеството му[1].

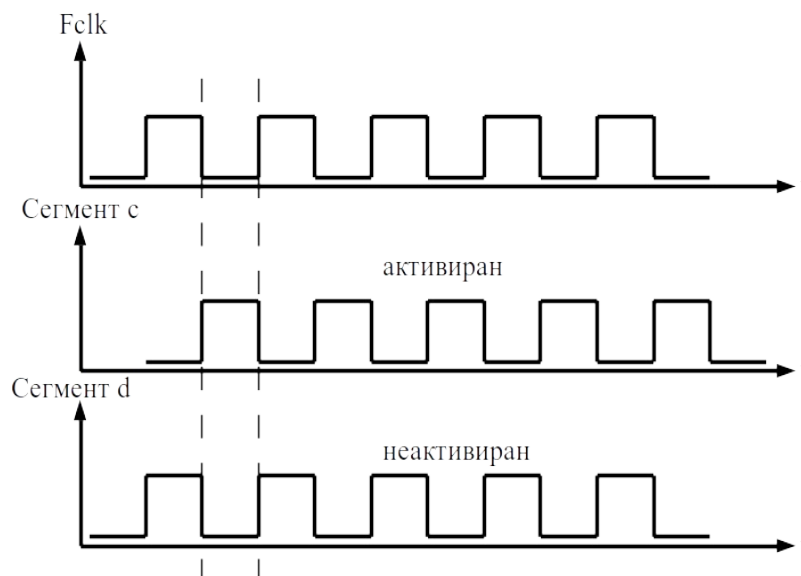
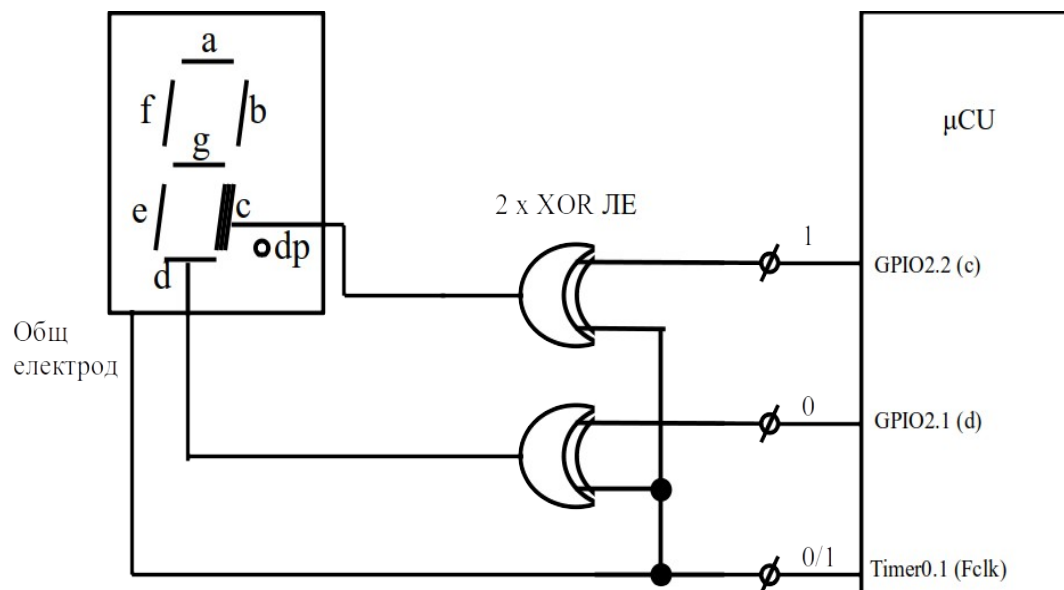
Както LED индикаторите, така и **LCD** могат да се разделят на:

- \*7-сегментни
- \*буквено-цифрови (alphanumeric)
- \*матрични

На следващия слайд е показано управлението на 7-сегментен индикатор. Забележете как сигналите са син- и противофазни. При противофазни сигнали съответния сегмент ще бъде активиран (видим). При синфазни сигнали сегментът ще бъде деактивиран (невидим).

# Управление на LCD дисплеи

Чрез използването на XOR ЛЕ,  $\mu$ CU ще може да управлява дисплея с постоянни напрежения (0 и 1). Те се преобразуват на променливи чрез XOR схемите и генерирания тактов сигнал от таймер.



XOR таблица на истинност

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0



# Управление на LCD дисплеи

**Буквено-цифрови дисплеи** – символите се изобразяват с малки матрици (напр. 5x8 пиксела), сигналите за които се получават от контролер на дисплея (драйвер).  $\mu$ CU се свързват към този драйвер посредством паралелен или сериен интерфейс.  $\mu$ CU диктува на кой ред, на кой индекс, какъв символ да бъде изобразен, а драйверът “рисува” символа на съответната матрица.

На пазара има много буквено-цифрови LCD дисплеи, но повечето са базирани или емулират драйвера на фирмата Hitachi – HD44780 [5].

# Управление на LCD дисплеи

LCD екранът в различните модели може да съдържа различен брой редове с различен брой символи:

- \*1 ред с 8 символа (съкращава се с 1x8), 1x16 LCD

- \*2x16 LCD, 2x20 LCD

- \*3x16 LCD, 3x20 LCD

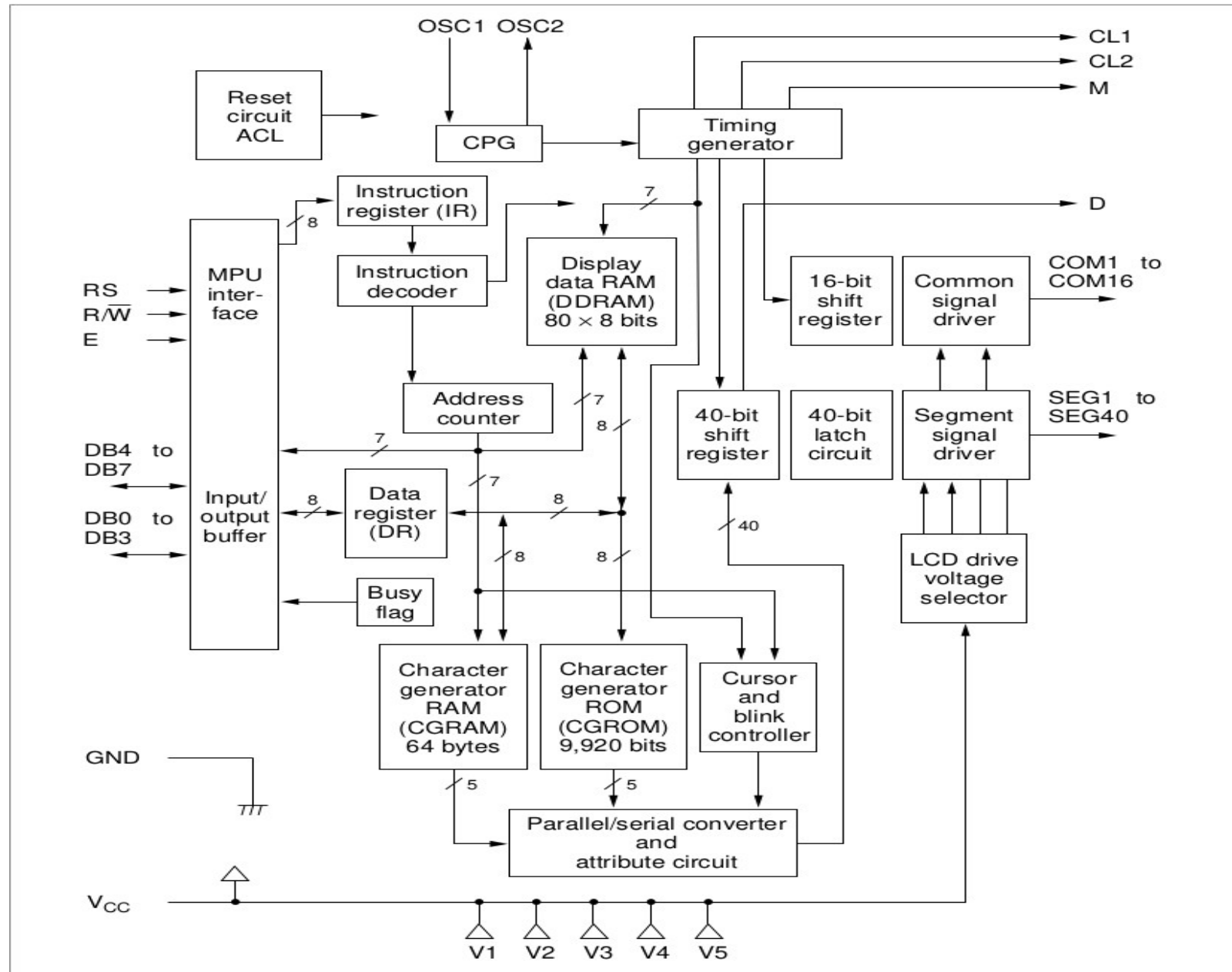
- \*4x16 LCD, 4x20 LCD

- \*и т.н.

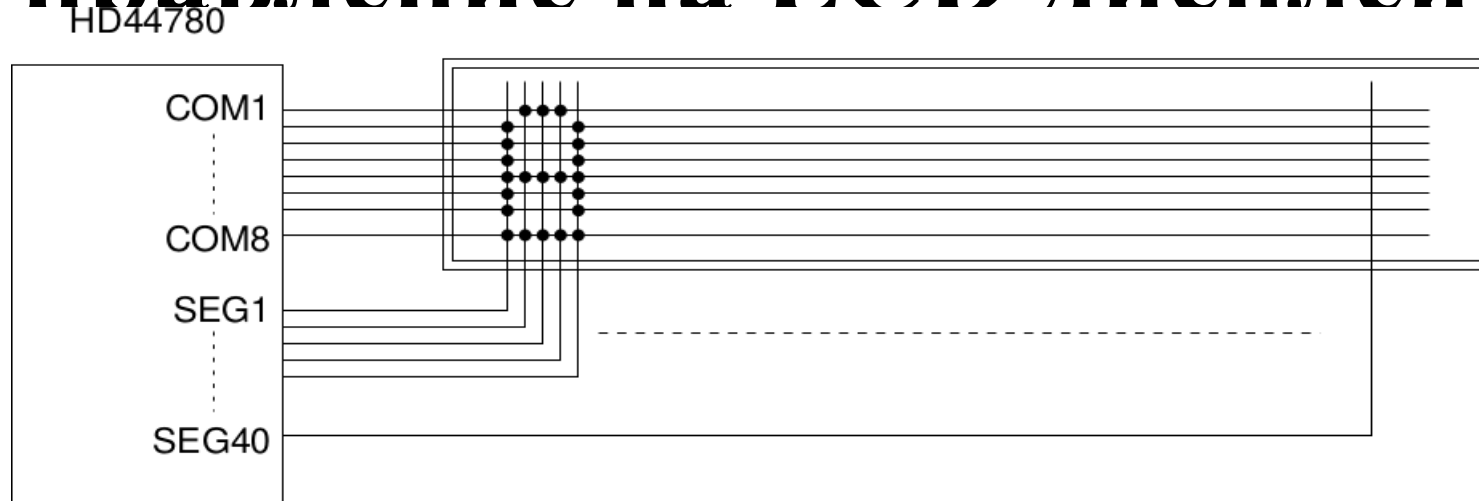
Блокова схема на HD44780 е показана на следващия слайд.

# Управление на LCD дисплеи

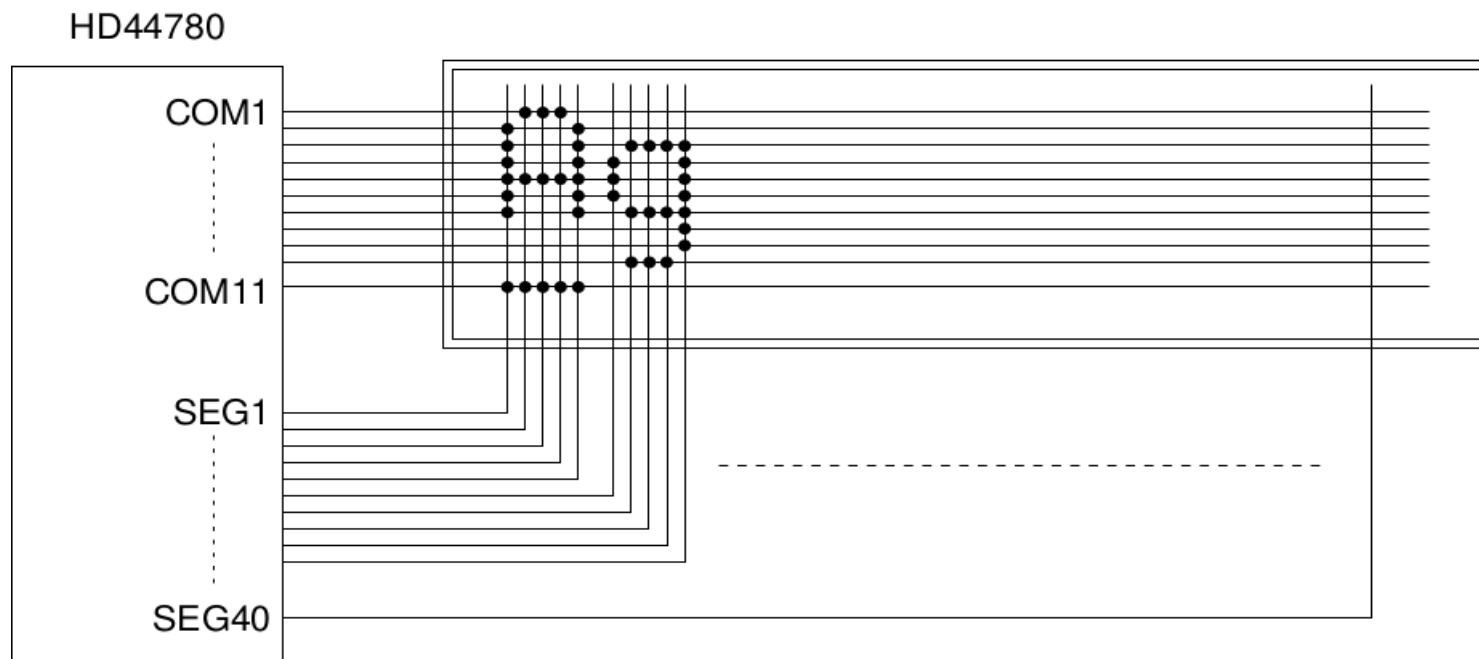
HD44780U Block Diagram



# Управление на LCD дисплеи



Example of a  $5 \times 8$  dot, 8-character  $\times$  1-line display (1/4 bias, 1/8 duty cycle)



Example of a  $5 \times 10$  dot, 8-character  $\times$  1-line display (1/4 bias, 1/11 duty cycle)

# Управление на LCD дисплеи

**MPU интерфейс** (MPU interface) – интерфейсен модул за свързване на драйвера с  $\mu$ CU. Съдържа следните сигнали:

**\*DB0 ÷ DB7** – паралелен 8-битов интерфейс, използван за трансфер на данни (символи, които трябва да се изобразяват) и инструкции (команди, поддържани от драйвера). Има режим, в който се използват само 4-бита от този интерфейс. За да се използва този режим, софтуерът на  $\mu$ CU трябва да следва специална инициализационна процедура.

**\*R/!W** – сигнал, контролиращ посоката на изводите DB0 ÷ DB7. Те може да са входове или изходи (съответно да се пише в или чете от LCD драйвера). Напр. може да се чете какво има на екрана в момента, както и да се следи BUSY флаг, показващ готовност да се приемат нови данни.

# Управление на LCD дисплеи

**E** – сигнал, под чието управление данните или инструкциите, подадени на DBxx изходите се прехвърлят във вътрешните регистри на LCD. Ако DBxx са изходи, може да се използва за обратния процес – да се прехвърлят данни от DBxx във входния GPIO регистър на  $\mu$ CU.

**RS** – сигнал, който трябва да се управлява от  $\mu$ CU, за да укаже дали данните по DBxx магистралата са данни (1) или инструкции (0).

# Управление на LCD дисплеи

**Регистър за инструкции** (instruction register) – съдържа инструкциите (командите), изпращани от  $\mu$ CU. Това са всъщност числа, които конфигурират LCD дисплея. Някои от опциите, които може да бъдат избрани са: изчистване на дисплея, вкл./изкл. на дисплея, вкл./изкл. на курсор, избор на броя активни редове, и др.

**Даннов регистър** (data register) – съдържа потребителските данни (символите), които трябва да бъдат изобразени. Символите се представят с ASCII кодовете им, т.е. ако трябва да се запише цифрата '1',  $\mu$ CU ще изпрати 0x31, 'U' – 0x55, и т.н. Понеже данновия регистър е само един, то записването в него символи се прехвърлят в буферна RAM памет, наречена **Data Display RAM**, DDRAM (да не се бърка с DDR RAM!).

# Управление на LCD дисплеи

**DDRAM** (**D**ata **D**isplay **R**andom **A**ccess **M**emory) – енергозависима памет, съдържаща символите, които се показват в момента (или предстои да бъдат показани) на дисплея.

**CGROM** (**C**haracter **G**enerator **R**ead **O**nly **M**emory) – енергонезависима памет, интегрирана в драйвера, която съдържа двоичния еквивалент на шрифта на символите от ASCII таблицата. Всеки байт от тази памет съответства на един ред от матрицата на символа. Поддържат се два размера:

\*5x8 пиксела (8 символа, описани с 8 реда x8 бита |3  
неизползвани|);

\*5x10 пиксела (4 символа, описани с 11 реда x8 бита |3  
неизползвани|).



# Управление на LCD дисплеи

## Instruction table at "Normal mode"

(When "EXT" option pin connect to VDD, the instruction set follow below table)

Instruction	Instruction Code										Description	Instruction Execution Time		
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		OSC=380KHz	OSC=540kHz	OSC=700KHz
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRAM. and set DDRAM address to "00H" from AC	1.08 ms	0.76 ms	0.59 ms
Return Home	0	0	0	0	0	0	0	0	1	x	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	1.08 ms	0.76 ms	0.59 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	26.3 us	18.5 us	14.3 us
Display ON/OFF	0	0	0	0	0	0	1	D	C	B	D=1:entire display on C=1:cursor on B=1:cursor position on	26.3 us	18.5 us	14.3 us
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	x	x	S/C and R/L: Set cursor moving and display shift control bit, and the direction, without changing DDRAM data.	26.3 us	18.5 us	14.3 us
Function Set	0	0	0	0	1	DL	N	x	x	x	DL: interface data is 8/4 bits N: number of line is 2/1	26.3 us	18.5 us	14.3 us
Set CGRAM	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter	26.3 us	18.5 us	14.3 us
Set DDRAM address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter	26.3 us	18.5 us	14.3 us
Read Busy flag and address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0	0	0
Write data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM)	26.3 us	18.5 us	14.3 us
Read data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM)	26.3 us	18.5 us	14.3 us

### Note:

Be sure the ST7032 is not in the busy state (BF = 0) before sending an instruction from the MPU to the ST7032. If an instruction is sent without checking the busy flag, the time between the first instruction and next instruction will take much longer than the instruction time itself. Refer to Instruction Table for the list of each instruction execution time.

Списък с инструкции на дисплея Sitronix ST7032, драйверът на който е аналогичен и съвместим с HD44780.

# Управление на LCD дисплеи

Връзка между адресите на DDRAM клетките и позицията на символа върху дисплея. Забележете адреса на първия символ от втория ред. Започва от 0x40.

Понеже командата за DDRAM адрес се разбира от контролера само, ако най-старшият бит DB7 = 1, то адреса на първия ред е  $0x00 \mid 0x80 = 0x80$ , а адреса на втория ред е  $0x40 \mid 0x80 = 0xC0$ .

0x00	0x01	0x02	0x03		0x27
a	b	c	d	...	*
\$	#	!	+	...	@
0x40	0x41	0x42	0x43		0x67

# Управление на LCD дисплеи

Задаването на нов ред става чрез задаване началния адрес на най-левия символ:

\*ред 1 = 0x80

\*ред 2 = 0xC0

\*ред 3 = 0x94

\*ред 4 = 0xD4

и т.н.

Това означава, че следната конструкция е невалидна:

```
char *str = (char *)"First line.\nSecond line.";
```

# Управление на LCD дисплеи

Трябва да се напише:

```
char *str1 = (char *)"First line."  
char *str2 = (char *)"Second line.";
```

След това се изпълнява кода:

```
lcd_string(1, str1);  
lcd_string(2, str2);
```

```
void lcd_string(uint8_t line_num, uint8_t *str){  
    Chip_GPIO_SetPinState(LPC_GPIO_PORT, 0, 4, 0); //RS=0  
    vTaskDelay(DELAY_1ms);  
  
    switch(line_num){  
    case 1:  
        spi_tx_byte = 0x80;  
        Chip_SPI_RWFrames_Blocking(LPC_SPI0, spi_tx_byte);  
        vTaskDelay(DELAY_1ms); // >26us  
        break;  
    case 2:  
        spi_tx_byte = 0xC0;  
        Chip_SPI_RWFrames_Blocking(LPC_SPI0, spi_tx_byte);  
        vTaskDelay(DELAY_1ms); // >26us  
        break;  
    }  
  
    Chip_GPIO_SetPinState(LPC_GPIO_PORT, 0, 4, 1); //RS=1  
    vTaskDelay(DELAY_1ms);  
  
    while(*str != '\0'){  
        spi_tx_byte = *str;  
        Chip_SPI_RWFrames_Blocking(LPC_SPI0, spi_tx_byte);  
        vTaskDelay(DELAY_1ms); // >26us  
        str++;  
    }  
}
```

# Управление на LCD дисплеи

Ако в ASCII таблицата няма подходящ символ за дадено приложение, програмистът може да дефинира свои чрез CGRAM.

Character Code (DDRAM Data)								CGRAM Address						Character Patterns (CGRAM Data)									
b7	b6	b5	b4	b3	b2	b1	b0	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0		
0	0	0	0	-	0	0	0	0	0	0	0	0	0	-	-	-	1	1	1	1	1		
					0	0	0				0	0	1				0	0	0				
					0	0	0				0	1	0				0	0	0				
					0	0	0				0	1	1				0	0	0				
					0	0	0				1	0	0				0	0	0				
					0	0	0				1	0	1				0	0	0				
					0	0	0				1	1	0				0	0	0				
					0	0	0				1	1	1				0	0	0				
0	0	0	0	-	0	0	1	0	0	1	0	0	0	-	-	-	1	1	1	1	0		
					0	0	1				0	0	1				1	0	0	1			
					0	0	1				0	1	0				1	0	0	1			
					0	0	1				0	1	1				1	1	1	0			
					0	0	1				1	0	0				1	0	0	0			
					0	0	1				1	0	1				1	0	1	0			
					0	0	1				1	1	0				0	1	0	1			
					0	0	1				1	1	1				1	0	0	0			

Table 4. Relationship between CGRAM Addresses, Character Codes (DDRAM) and Character patterns (CGRAM Data)

# Управление на LCD дисплеи

```
uint8_t circle_symbol[8];
```

```
void init_special_chars(void){  
    circle_symbol[0] = 0x00;  
    circle_symbol[1] = 0x0E;  
    circle_symbol[2] = 0x1F;  
    circle_symbol[3] = 0x1F;  
    circle_symbol[4] = 0x1F;  
    circle_symbol[5] = 0x1F;  
    circle_symbol[6] = 0x0E;  
    circle_symbol[7] = 0x00;  
    ...  
}
```

```
void main(void){  
    ...  
    init_special_chars();  
    display_cgram_char(line_num, pos, SPEC_CHAR_CIRCLE, circle_symbol);  
    ...  
}
```

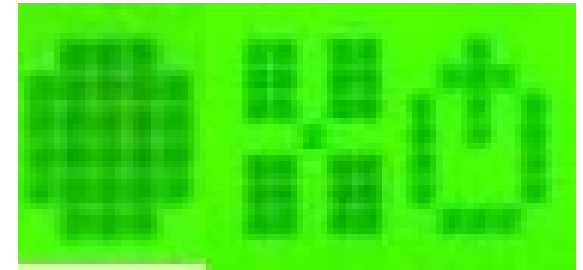
# Управление на LCD дисплеи

```
void display_cgram_char(uint8_t line, uint8_t pos, uint8_t cgram_char_num, uint8_t
*pattern){
    uint8_t i;
    uint8_t cgram_addr;

    switch(cgram_char_num){
    case SPEC_CHAR_CIRCLE:
        cgram_addr = 0;
        break;
    case SPEC_CHAR_FAN:
        cgram_addr = 8;
        break;
    case SPEC_CHAR_LINK:
        cgram_addr = 16;
        break;
    ...
    default:
        cgram_addr = 0;
        break;
    }

    for(i = 0; i < 8; i++){
        write_cmd(cgram_addr | LCD_CGRAM_ADDR | i); //LCD_CGRAM_ADDR = 0x40
        write_data(pattern[i]);
    }

    write_cmd(line+pos);
    write_data(cgram_char_num);
}
```



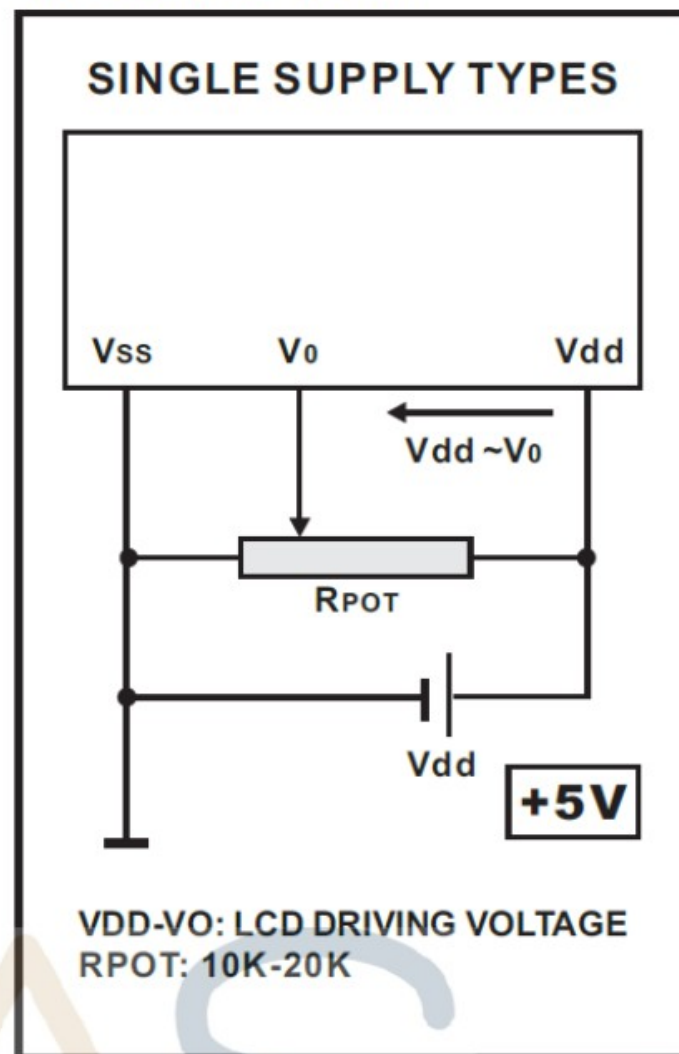
# Управление на LCD дисплеи

Изводите на LCD дисплеите са показани по-долу[6]:

## 8.PIN ASSIGNMENT

PIN	SYMBOL	FUNCTION
1	Vss	GND
2	Vdd	Power supply for LCM (+5.0V)
3	V0	Contrast Adjust
4	RS	Register Select Signal
5	R/W	Data Read / Write
6	E	Enable Signal
7-14	DB0 - DB7	Data bus line
15	LED+	Power supply for BKL (+5.0V)
16	LED-	Power supply for BKL (0V)

## 9.POWER SUPPLY





# Управление на LCD дисплеи

**DB0 ÷ DB7, RS, R /!W, E** са сигнали на LCD драйвера (HD44780).

**Vdd, Vss** – захранващи изводи за целия LCD модул. Обикновено е 3.3 V или 5 V.

**V0** – настройка на контраст. Трябва да се свърже към напрежение, което може да се донастройва. Например, използва се тример-потенциометър, както е показано на миналия слайд. Крайните му изводи се свързват към Vdd и Vss, а средната точка – към V0 на LCD. При производството на уреда, в който се вгражда LCD-то, трябва да се настрои контраста да е максимален, гледайки го от ъгъл, от който най-често потребителя ще го гледа. След това тримера може да се покрие с лак, за да не се развива от вибрации.

# Управление на LCD дисплеи

**LED+**, **LED-** - изводи на светодиодиите, които са поставени под стъклото на LCD. Съвкупността от тези светодиоди се нарича **подсветка** (backlight). Подсветката се използва, за да може символите да се виждат и в тъмно помещение. При някои LCD модули се изисква свързването на външен токоограничаващ резистор последователно на LED+ и LED- изводите. При други този резистор е вграден и въпросните изводи може да се свържат към захранване и маса. Консумацията на ток от светодиодиите варира от  $5 \div 200 \text{ mA}$ , и захранването трябва да е съобразено с тази стойност.

# Управление на LCD дисплеи

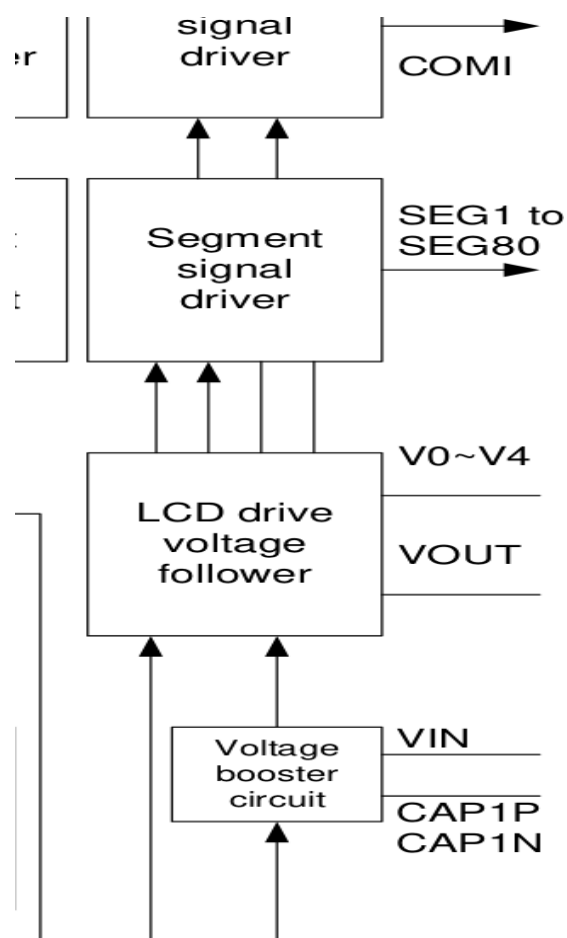
В електрониката се стремим да избягваме донастройващите тримери. Въртенето на тримера от оператор при производството е **бавен и неточен процес**. **Вибрации** по време на експлоатацията може да **променят позицията** на средната точка, което налага заливането на елемента с лак – също бавен процес.

Затова фирмата Sitronix е вградила в своя дисплеи програмируем усилвател, с чиято помощ може софтуерно да се задава амплитудата на поляризиращото напрежение.

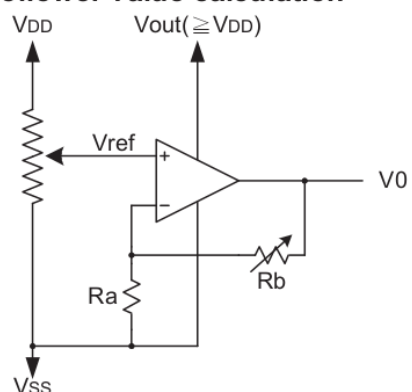
За да може дисплея да работи на 3.3 V, вграден е и капацитивен постояннотокков повишаващ преобразувател, свързан към хранящите изводи на програмируемия усилвател.

# Управление на LCD дисплеи

Пример – Sitronix ST7032 има повишаващ преобразувател и програмируем усилвател за задаване на поляризиращо напрежение и постигане на оптимален контраст. C0 ÷ C5 и Rab0 ÷ Rab2 са битове от два регистъра, контролиращи усилвателя.



V0 voltage follower value calculation



$$V0 = \left(1 + \frac{R_b}{R_a}\right) * V_{ref}$$

$$\text{While } V_{ref} = V_{DD} * \left(\frac{\alpha + 36}{100}\right)$$

C5	C4	C3	C2	C1	C0	$\alpha$
0	0	0	0	0	0	0
0	0	0	0	0	1	1
0	0	0	0	1	0	2
⋮						⋮
1	1	1	1	0	1	61
1	1	1	1	1	0	62
1	1	1	1	1	1	63

Rab2	Rab1	Rab0	1+Rb/Ra
0	0	0	1
0	0	1	1.25
0	1	0	1.5
0	1	1	1.8
1	0	0	2
1	0	1	2.5
1	1	0	3
1	1	1	3.75

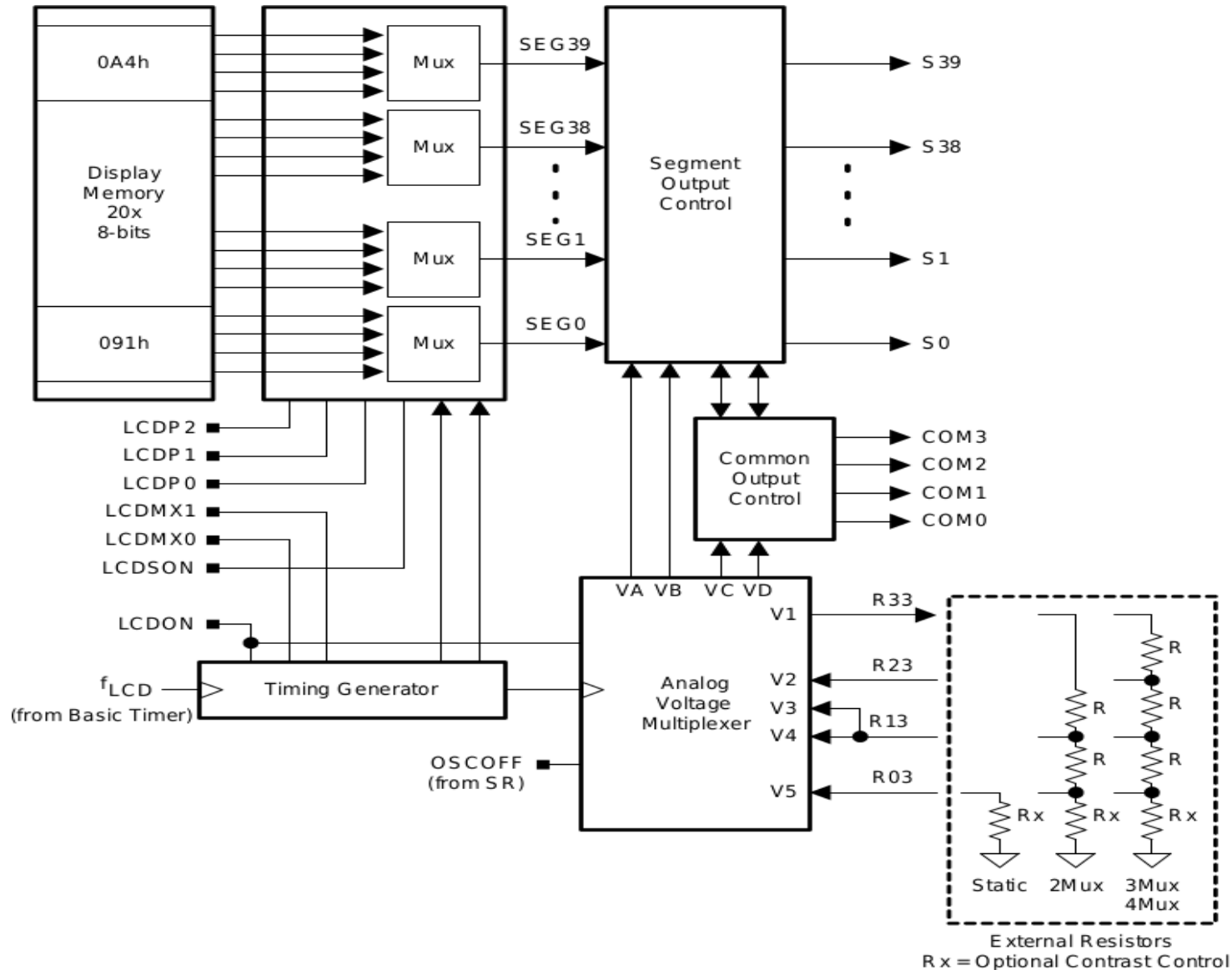
# Управление на LCD дисплеи

Някои  $\mu$ CU имат вграден драйвер за LCD.

Тогава не се налага използването на HD44780 и изводите на LCD стъклото може да се свържат директно.

# Управление на LCD дисплеи

Пример – MSP430FG4618 има вграден LCD драйвер.



# Управление на LCD дисплеи

Забележете, че и драйвера на MSP430, и драйвера HD44780 изискват няколко напрежения за управлението на LCD. Обикновено те се получават от общ еталонен източник, който захранва резисторен делител с много средни точки, в които се получават необходимите напрежения. Тези напрежения се използват, за да се **формират различни амплитуди при динамична индикация**. Както и при светодиодните дисплеи, така и тук динамичната индикация **позволява да се намали броя на управляващите сигнали**. При LCD се използва факта, че ниските поляризиращи напрежения не могат да направят сегмента видим [7].

# Управление на LCD дисплеи

С 4 извода се обхождат  
4 сегмента.

In 2-mux mode, each MSP430 segment pin drives two LCD segments, and two common lines (COM0 and COM1) are used. Figure 36-6 shows some example 2-mux 1/2-bias waveforms.

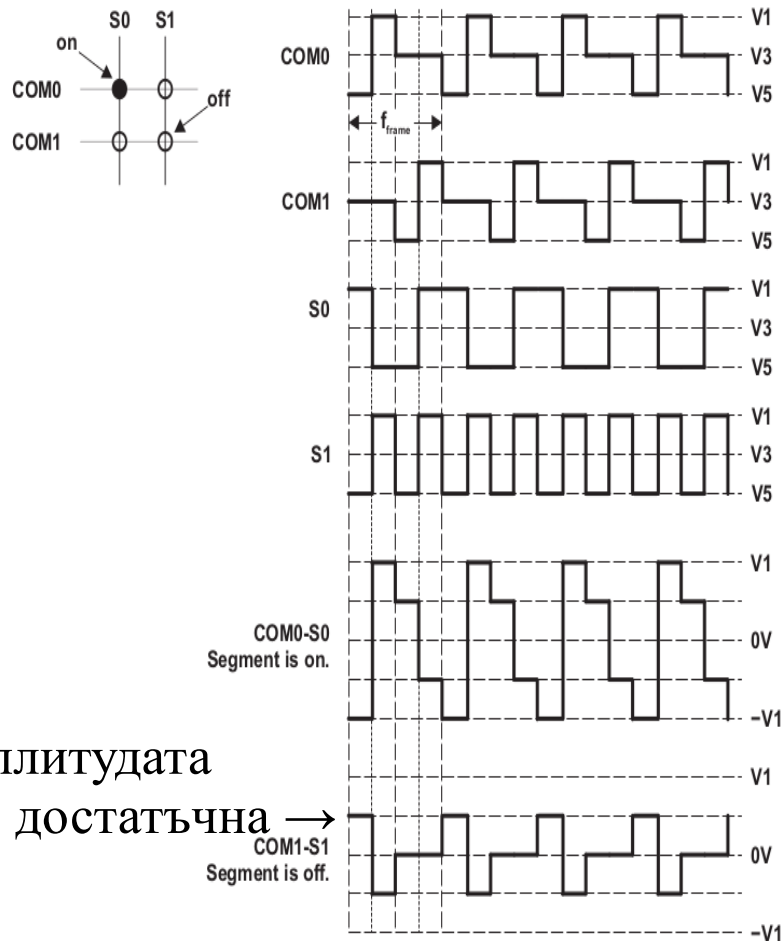


Figure 36-6. Example 2-Mux Waveforms

С 8 извода се обхождат  
16 сегмента.

In 8-mux mode, each MSP430 segment pin drives eight LCD segments, and eight common lines (COM0 through COM7) are used. Figure 36-10 shows some example 8-mux 1/3-bias waveforms.

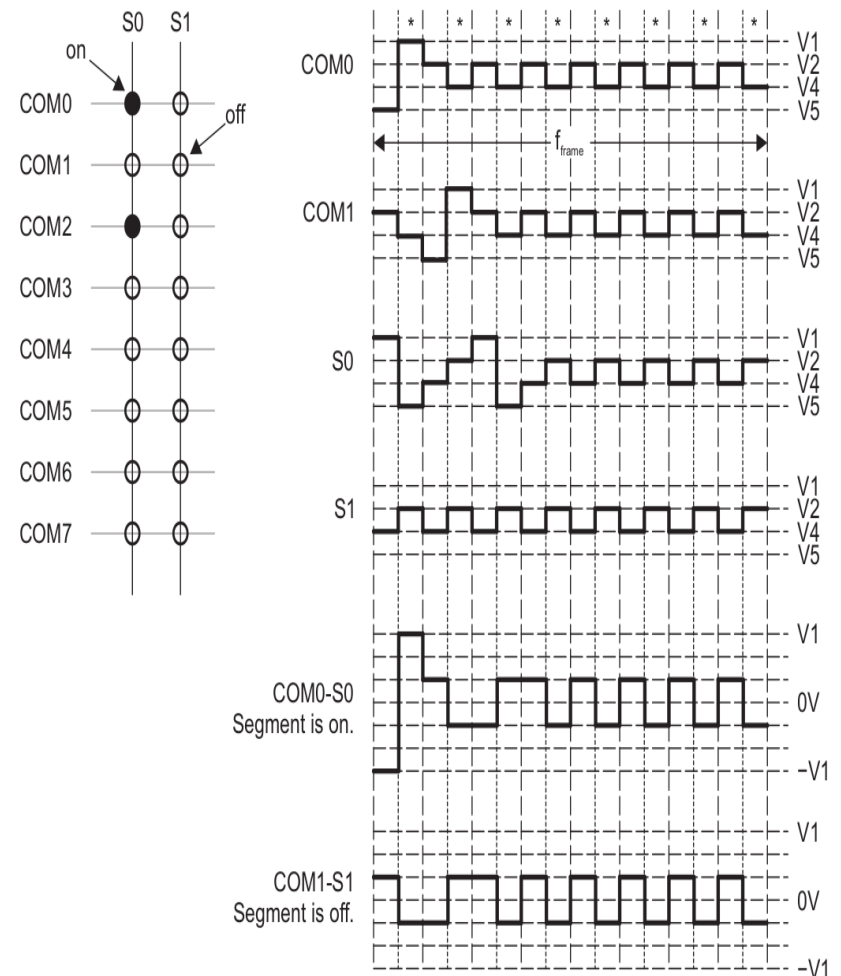


Figure 36-10. Example 8-Mux, 1/3 Bias Waveforms (LCDLP = 0)

Амплитудата  
не е достатъчна →



# Управление на LCD дисплеи

Описаните дотук дисплеи са буквено-цифрови, което означава, че те могат да изобразяват само ASCII или потребителски-дефинирани символи.

За да се покаже **изображение**, трябва да се използват **графични LCD дисплеи**.

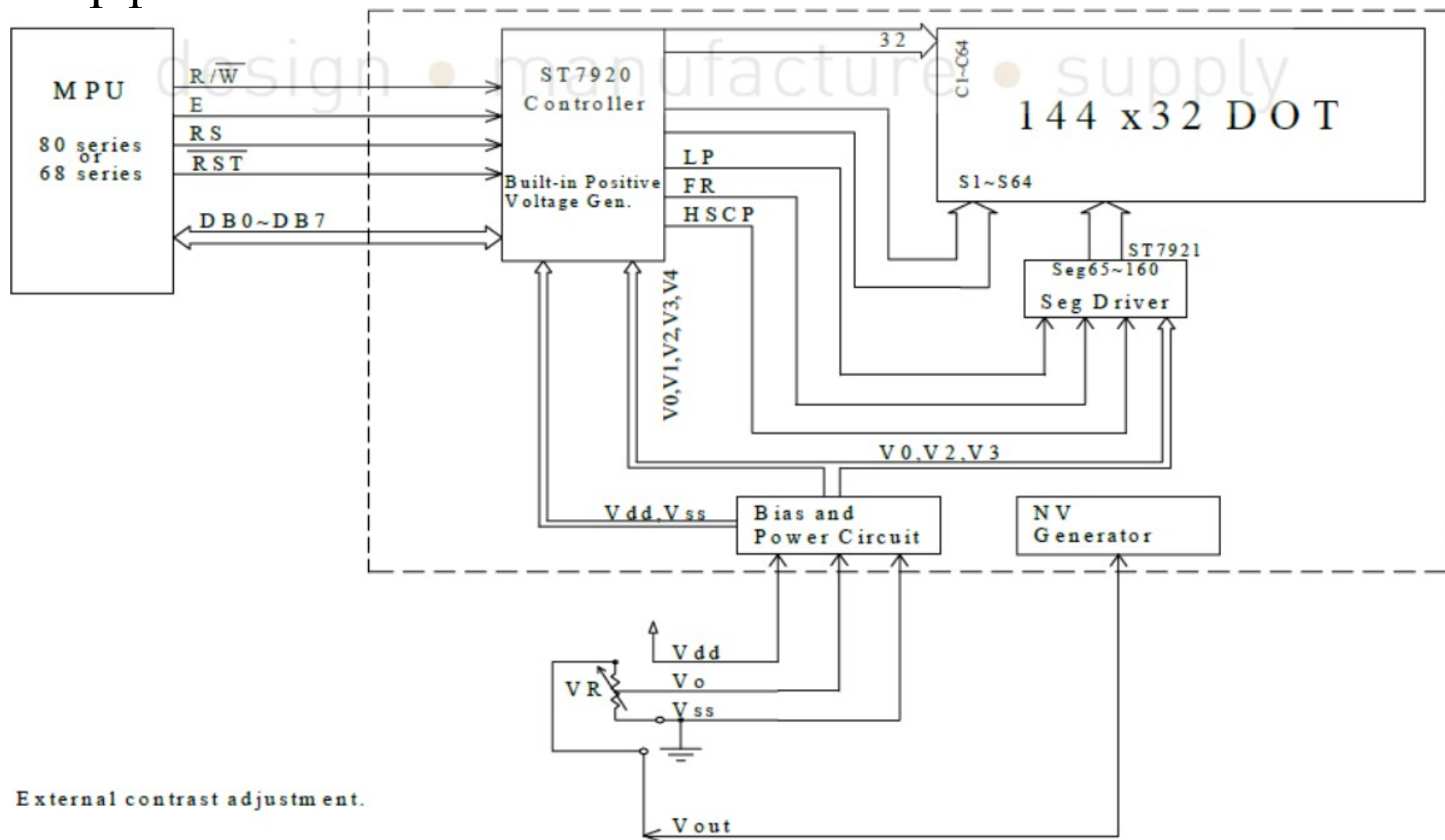
При LCD дисплеите няма сегменти, а множество от точки (пиксели). Броя на точките определя **резолюцията** на дисплея.

Графичните LCD се разделят на:

- \*монохромни (пикселите са само черни точки)
- \*цветни (пикселите могат да бъдат цветни точки)
- \*цветни тъч (екранът е чувствителен на допир и може да извести  $\mu$ CU за точката, която потребителя е натиснал).

# Управление на LCD дисплеи

Блокова схема на графичен LCD дисплеи с паралелен интерфейс.



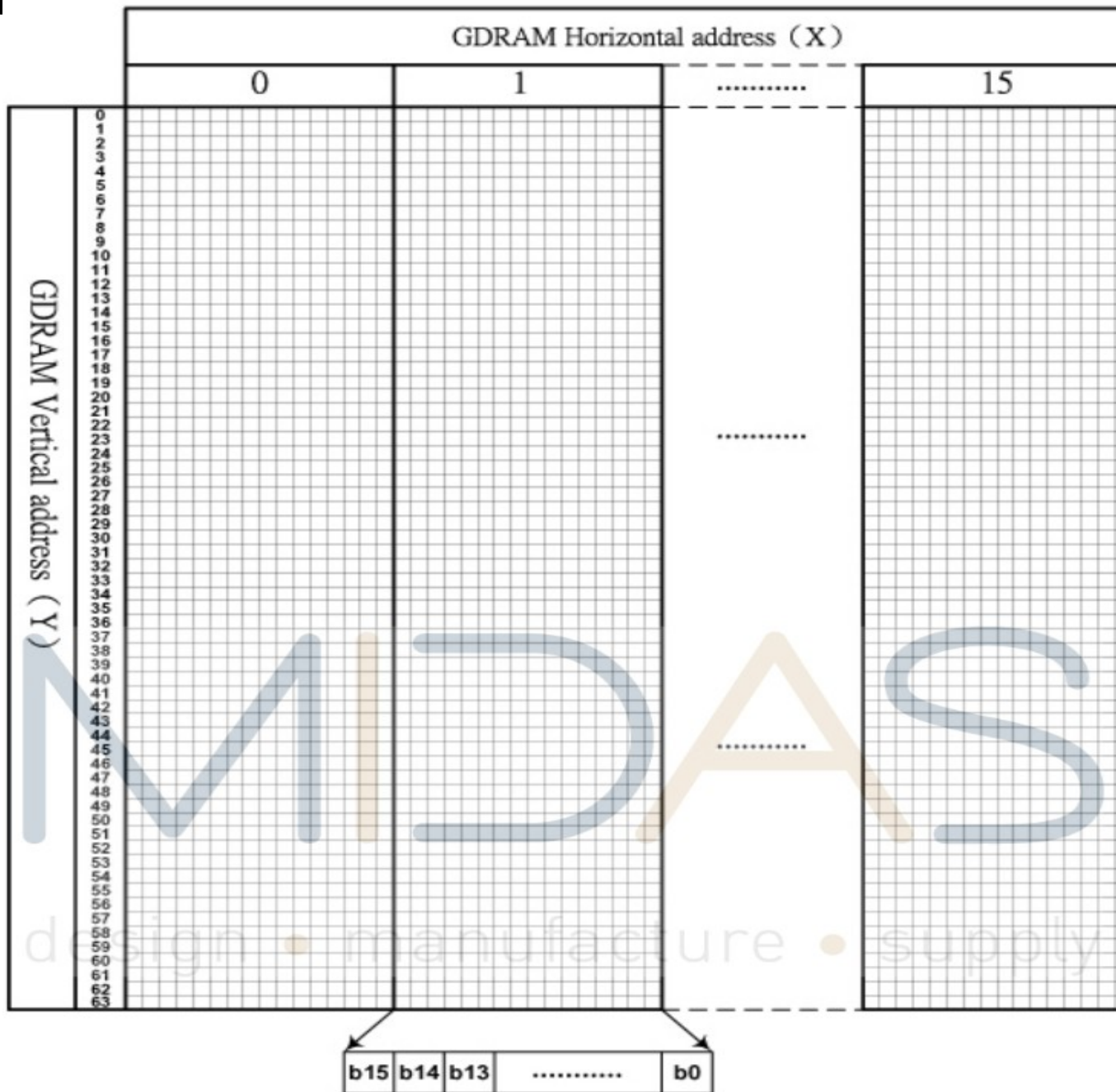
# Управление на LCD дисплеи

**GDRAM (Graphic Display RAM)** – RAM памет, в която на всеки бит от всяка клетка отговаря за управлението на един пиксел.

За да се рисува по екрана, трябва да се спазва следната процедура [6]:

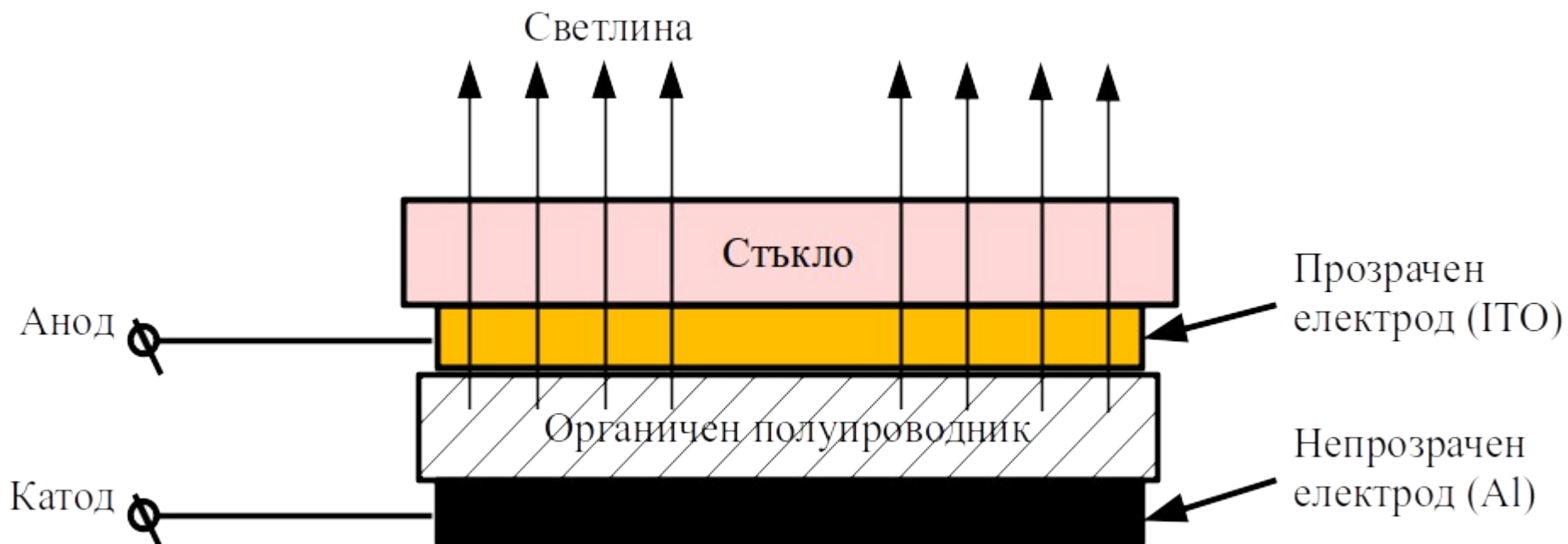
- \*задава се вертикален адрес (Y) на GDRAM колона
- \*задава се хоризонтален адрес (X) на GDRAM ред
- \*записва се MSB ( $D_{15} \div D_8$ ) октета на данните за пиксела в регистъра, чийто адрес е пресечната точка на X и Y
- \*записва се LSB ( $D_7 \div D_0$ ) октета на данните за пиксела

# Управление на LCD дисплеи



# Управление на OLED дисплеи

**OLED дисплеи (Organic Light Emitting Diode)** – дисплеи, при които всеки пиксел се реализира със светодиод. Понеже светодиодите са направени от органични материали, имащи свойствата на полупроводници, оттам идват и имената им – “органични”. Типична конструкция на органичен LED е показана по-долу:



# Управление на OLED дисплеи

OLED технологията позволява **по-големи ъгли на виждане**.

OLED не се нуждаят от подсветка.

OLED дисплеите могат да се разделят на:

- \*буквено-цифрови (използващи аналози на контролера HD44780, но всеки пиксел от 5x8 матрицата е органичен LED).

- \*графични

Управлението на такива дисплеи е аналогично на дотук описаните LCD.

OLED драйверите се захранват с ниски напрежения  $3.3 \div 5 \text{ V}$ , но се **изискват високи напрежения за OLED матрицата** ( $8 \div 18 \text{ V}$ ).

# Управление на OLED дисплеи

Използвани интерфейси са:

- \*паралелен
- \*сериен SPI
- \*RGB (18/16/6-битов)

Обикновено OLED драйвера е монтиран на гърба на OLED матрицата и се свързва към  $\mu$ CU посредством лентов кабел (stripline ribbon cable).

# Управление на OLED дисплеи

RGB интерфейсът съдържа следните сигнали:

**\*Enable** – сигнал за разрешаване на видео входа. Данните трансфери по DB0 ÷ DB17 се извършват, когато този сигнал е в ниско ниво.

**\*DOTCLK** – тактов вход за точката. Записите в DDRAM паметта се тактуват от този сигнал. Един фронт по DOTCLK линията прехвърля данните от DB0 ÷ DB17 (RGB цветове) във регистър от DDRAM за 1 пиксел от екрана.

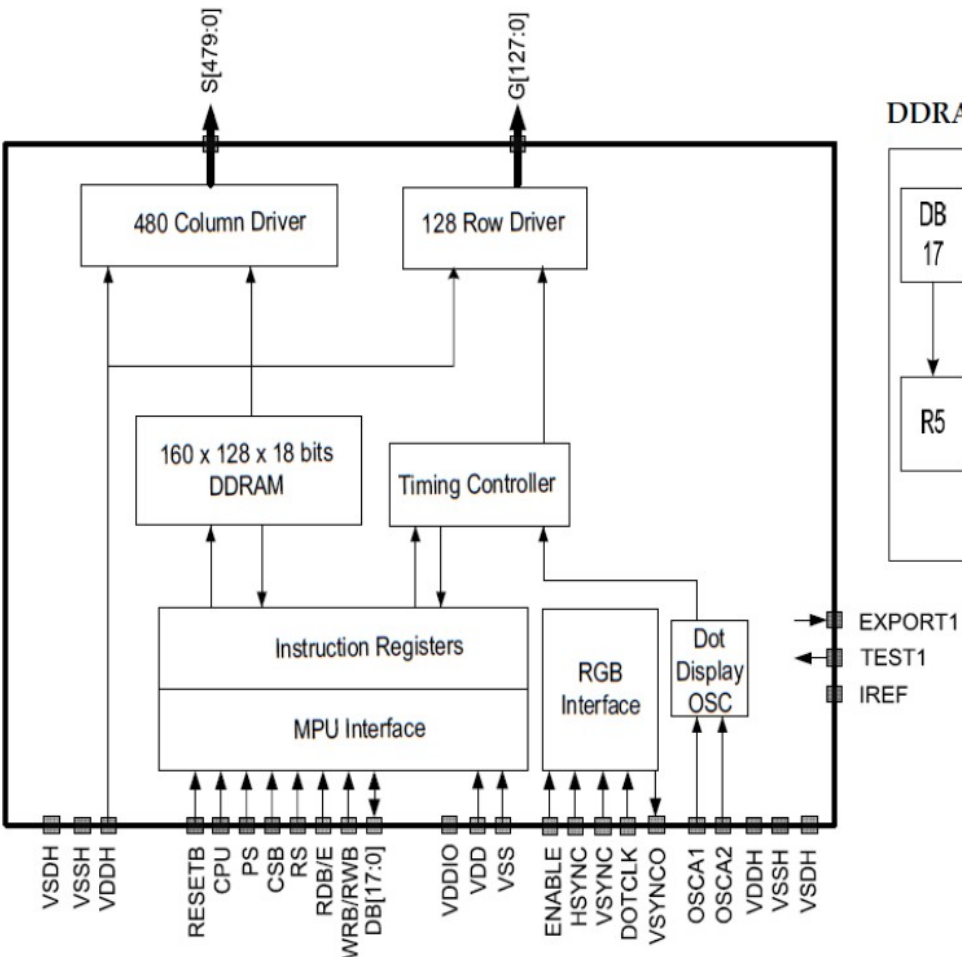
**\*HSYNC** – сигнал за хоризонтална синхронизация. Използва се за рестартиране на адресния брояч на колоните на дисплея.

**\*VSYNC** – сигнал за вертикална синхронизация. Използва се за рестартиране на адресния брояч на редовете на дисплея.<sup>80/111</sup>

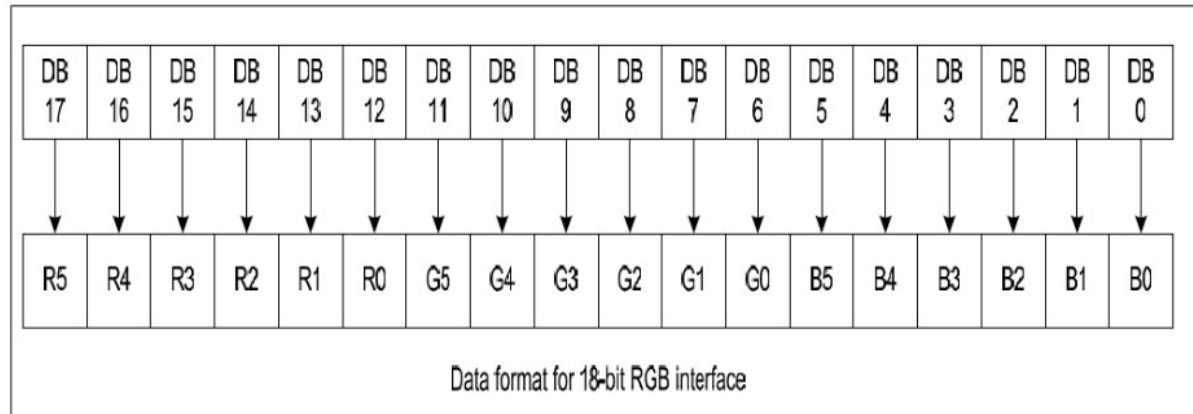


# Управление на OLED дисплеи

*Пример – вътрешната структура на графичен OLED драйвер с RGB интерфейс SEPS525[8].*



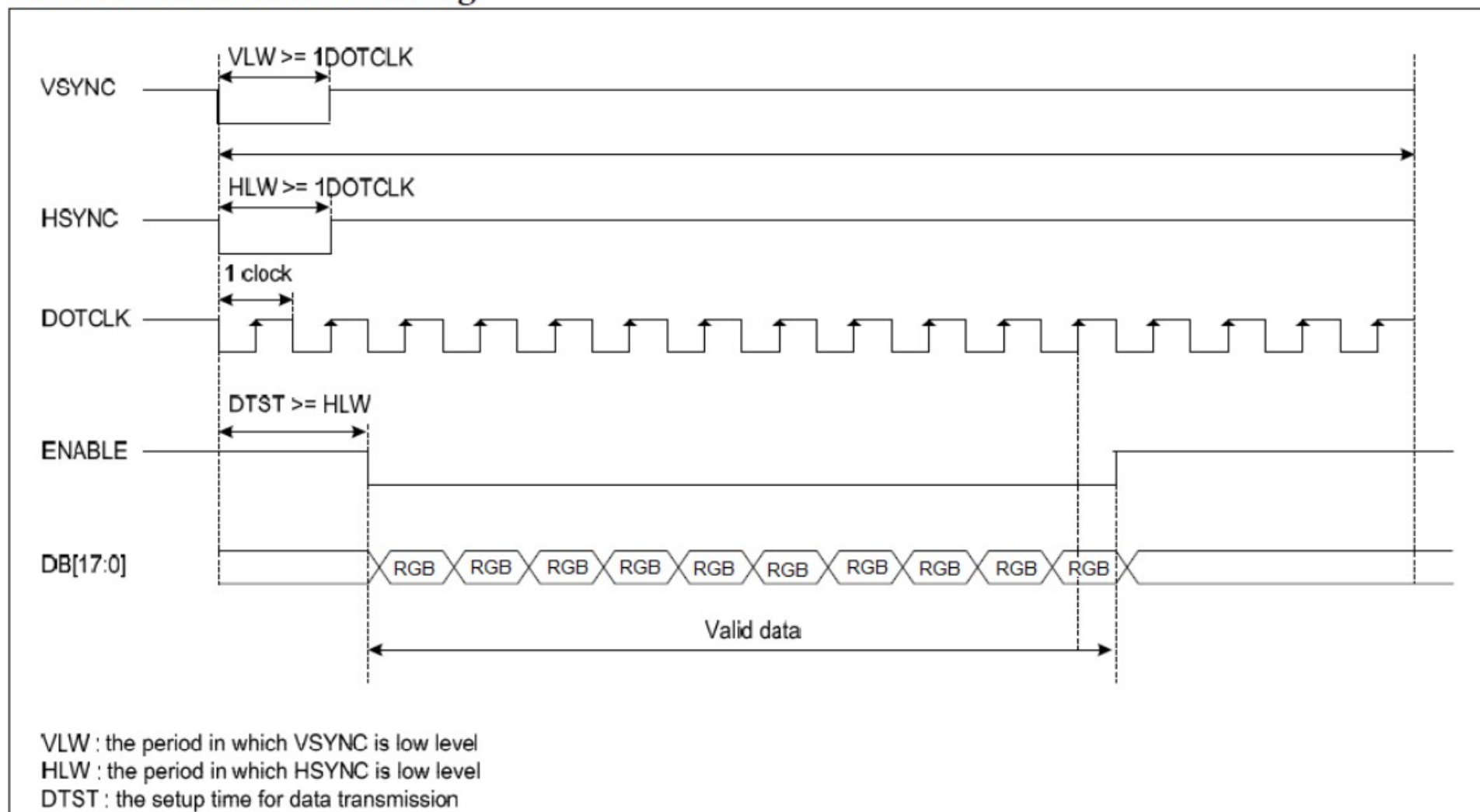
DDRAM Write



# Управление на OLED дисплеи

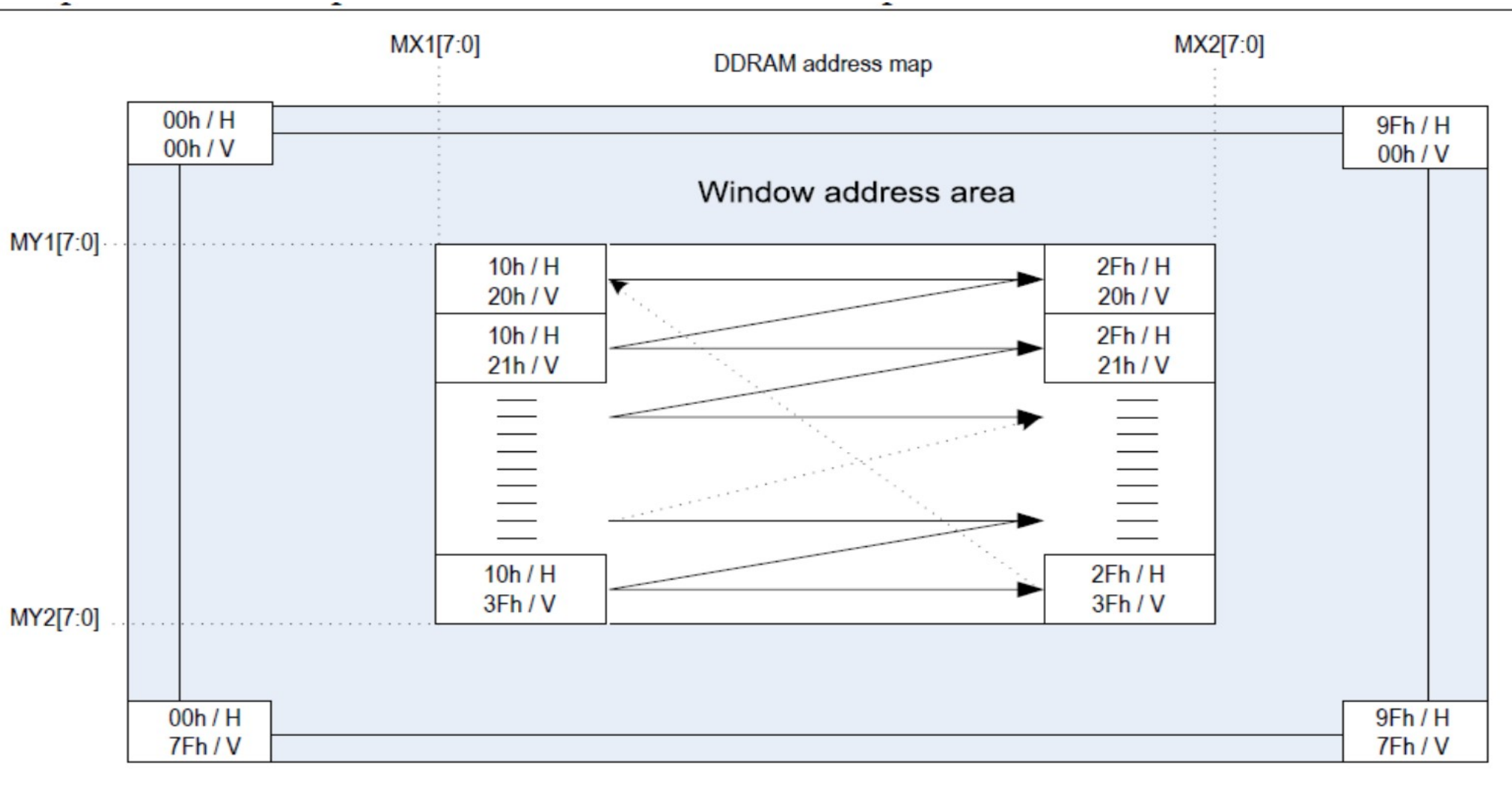
*Пример* – времедиаграма на изпращане на данни за 1 пиксел по RGB интерфейс [8].

18/16-bit RGB interface timing



# Управление на OLED дисплеи

Опресняването на дисплея става от горен-ляв ъгъл надолу. Когато се достигне адреса на последния пиксел, адресните броячи (хоризонтален и вертикален) се нулират.

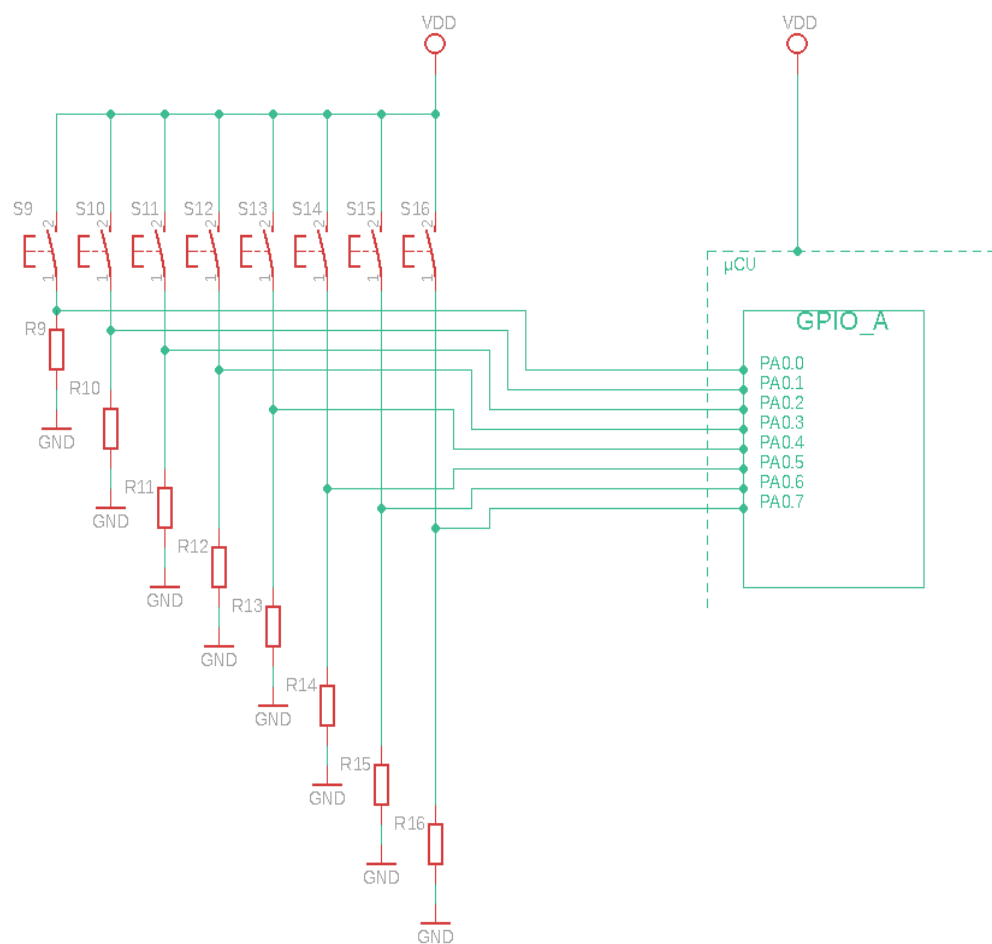
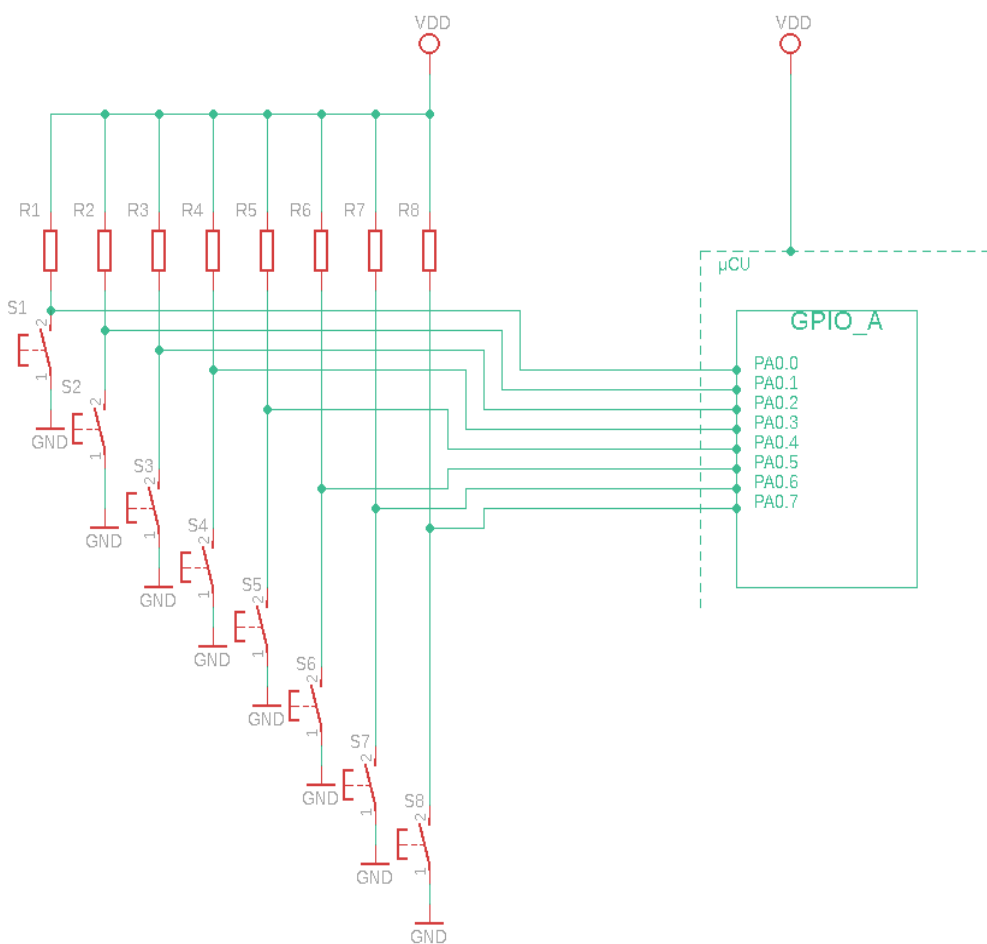


# Бутони и клавиатури

**Механични бутони и ключове** може да се четат от един  $\mu\text{CU}$  посредством GPIO модула. Изводите, на които има свързани бутони, трябва да бъдат конфигурирани като входове с или без издърпващи резистори.

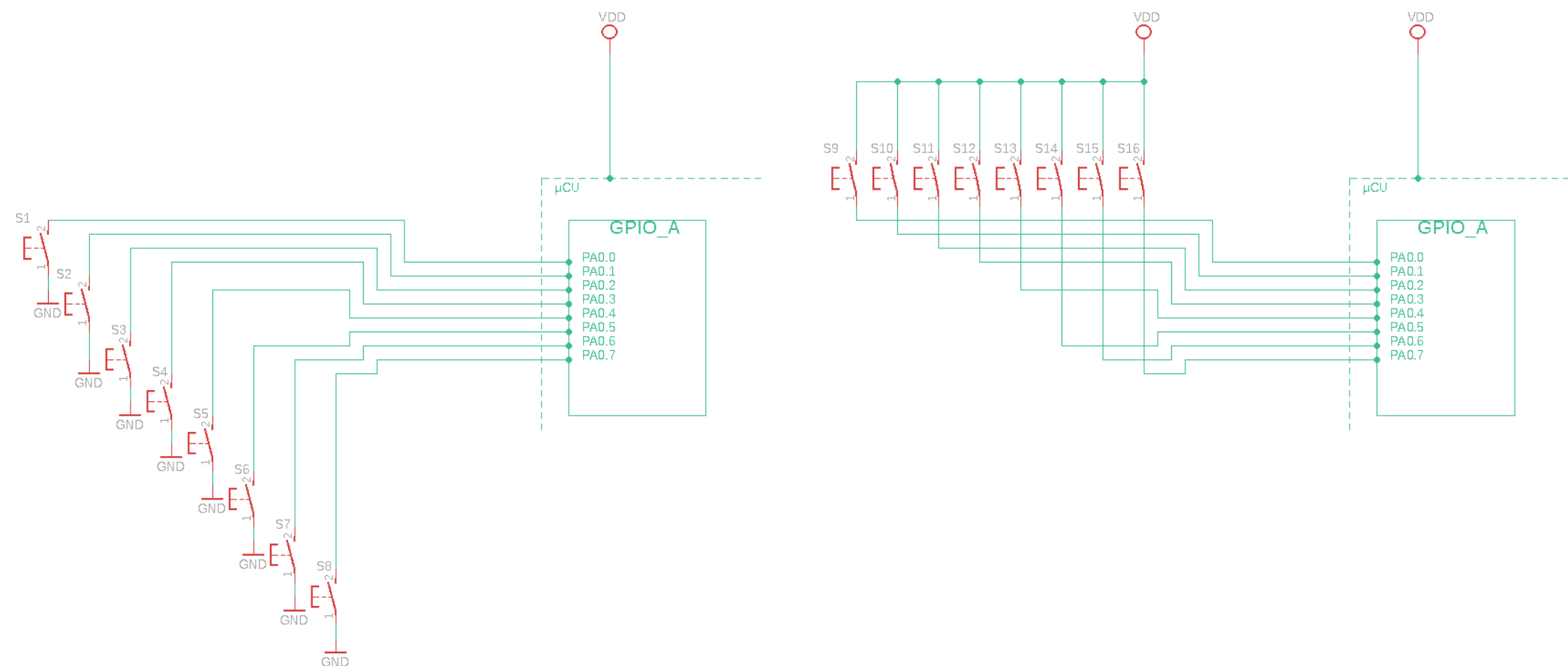
# Бутони и клавиатури

Свързване на механични бутони без включване на вътрешните издърпващи резистори. S1 ÷ S8 задават логическа 0 при натиск. S9 ÷ S16 задават логическа 1 при натиск.



# Бутони и клавиатури

Свързване на механични бутони с включване на вътрешните издърпващи резистори. S1 ÷ S8 задават логическа 0 при натиск. S9 ÷ S16 задават логическа 1 при натиск.



# Бутони и клавиатури

При натискане на механичен бутон се появяват **микромеханични трептения на контакта**, които може да доведат до регистриране на фалшиви натискания от програмата на  $\mu$ CU.

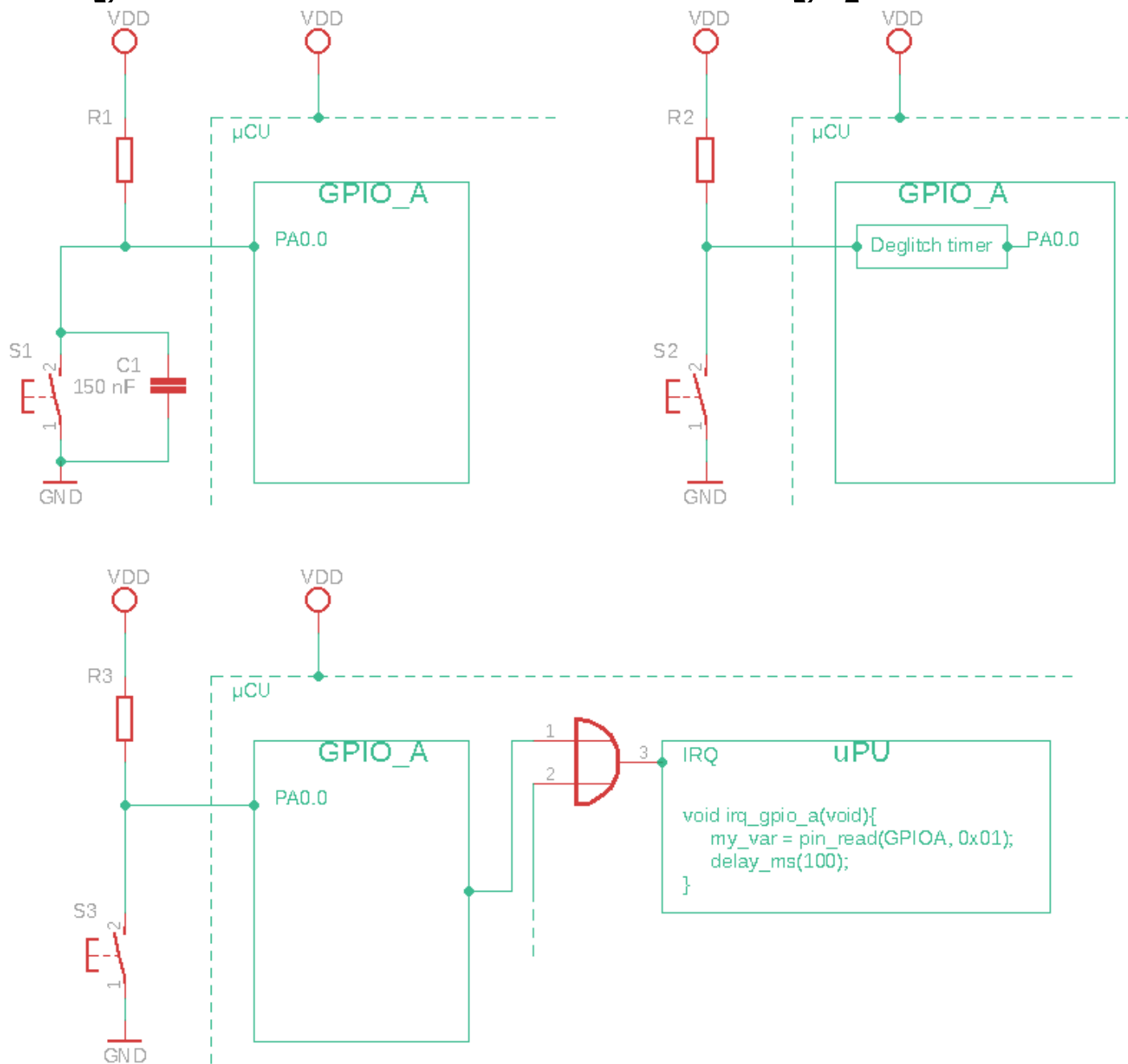
Затова те трябва да бъдат **филтрирани** (pin debounce):

- \*хардуерно с външен кондензатор, паралелно свързан на бутона

- \*хардуерно с таймер за deglitch

- \*софтуерно (чрез изчакване)

# Бутони и клавиатури





# Бутони и клавиатури

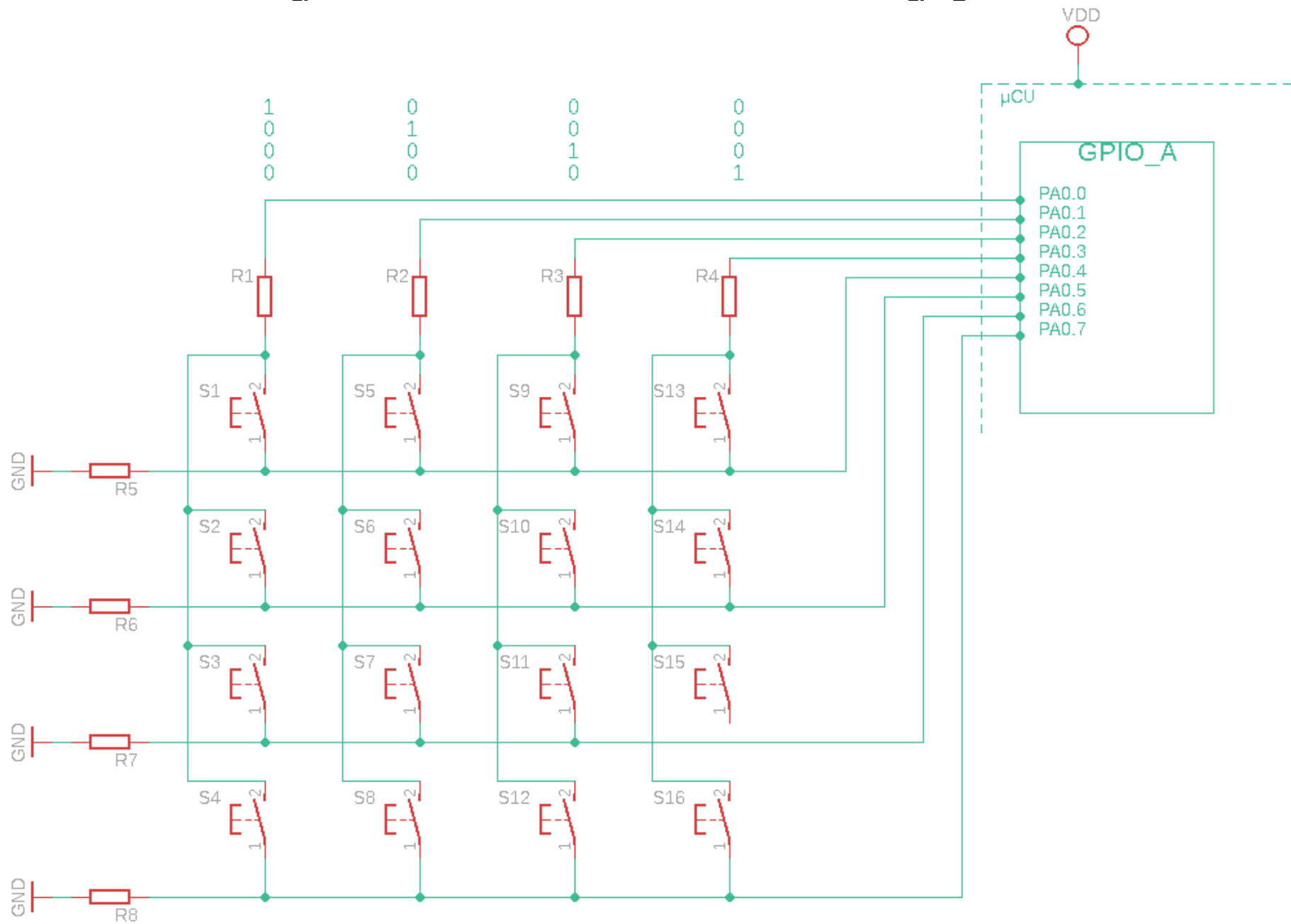
За да се намали броя на използваните изводи, клавиатурите може да бъдат **четени динамично**. Използва се т.нар. бягаща единица, която захранва всяка колона само в даден момент от времето.

Принципна схема е показана на следващия слайд.

Резистори  $R5 \div R8$  са издърпващи към маса.

Резистори  $R1 \div R4$  са защитни, за случаите когато бъдат натиснати два бутона от един и същ ред едновременно. Тези резистори трябва да са с много по-малки стойности от  $R5 \div R8$ , защото в противен случай нивото на логическата единица може да се окаже твърде ниско.

# Бутони и клавиатури



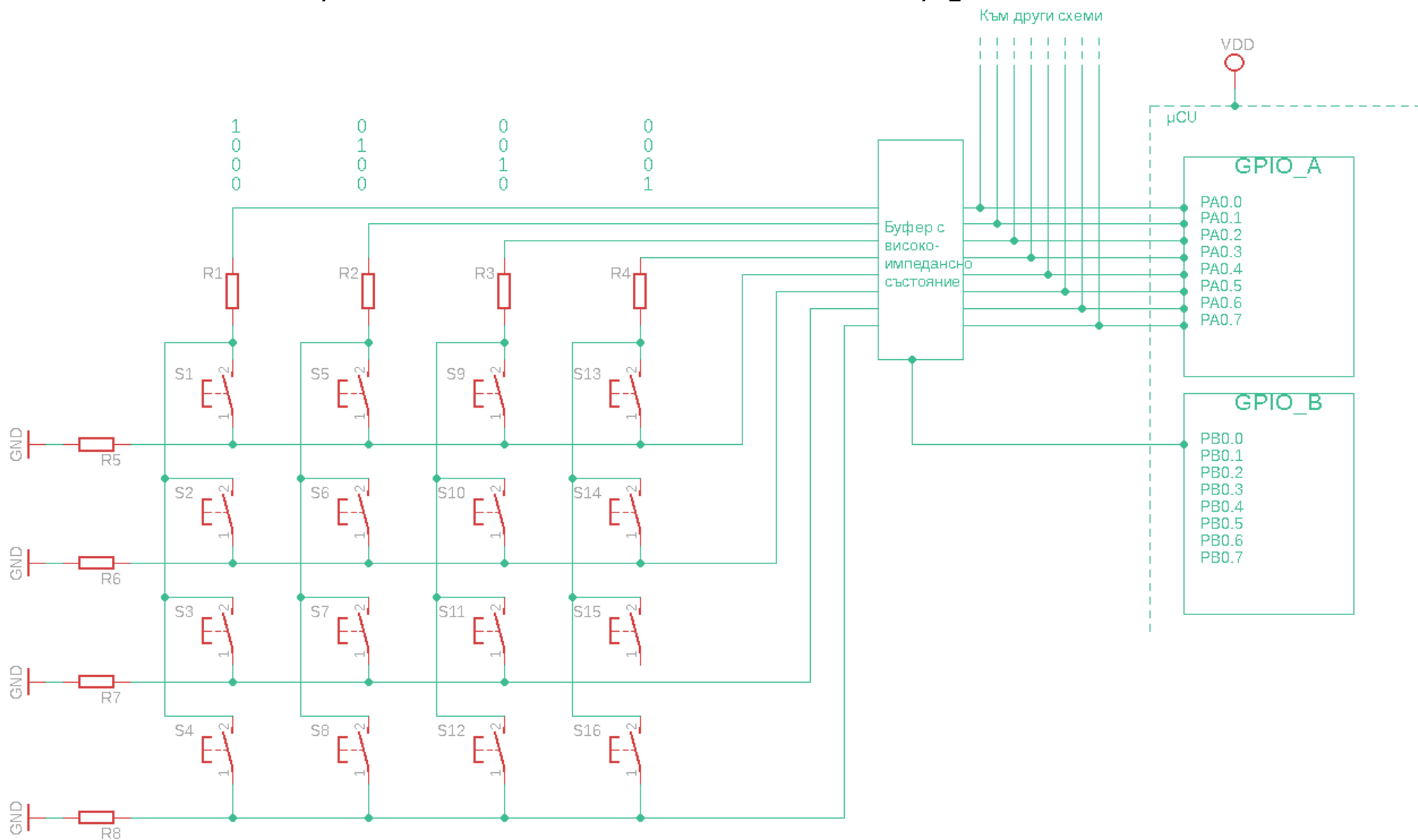
# Бутони и клавиатури

Изводите, с които се чете клавиатурата **може да се използват и за други периферни схеми**, понеже натискането на бутон е **бавен процес** от гледна точка на  $\mu\text{CU}$ .

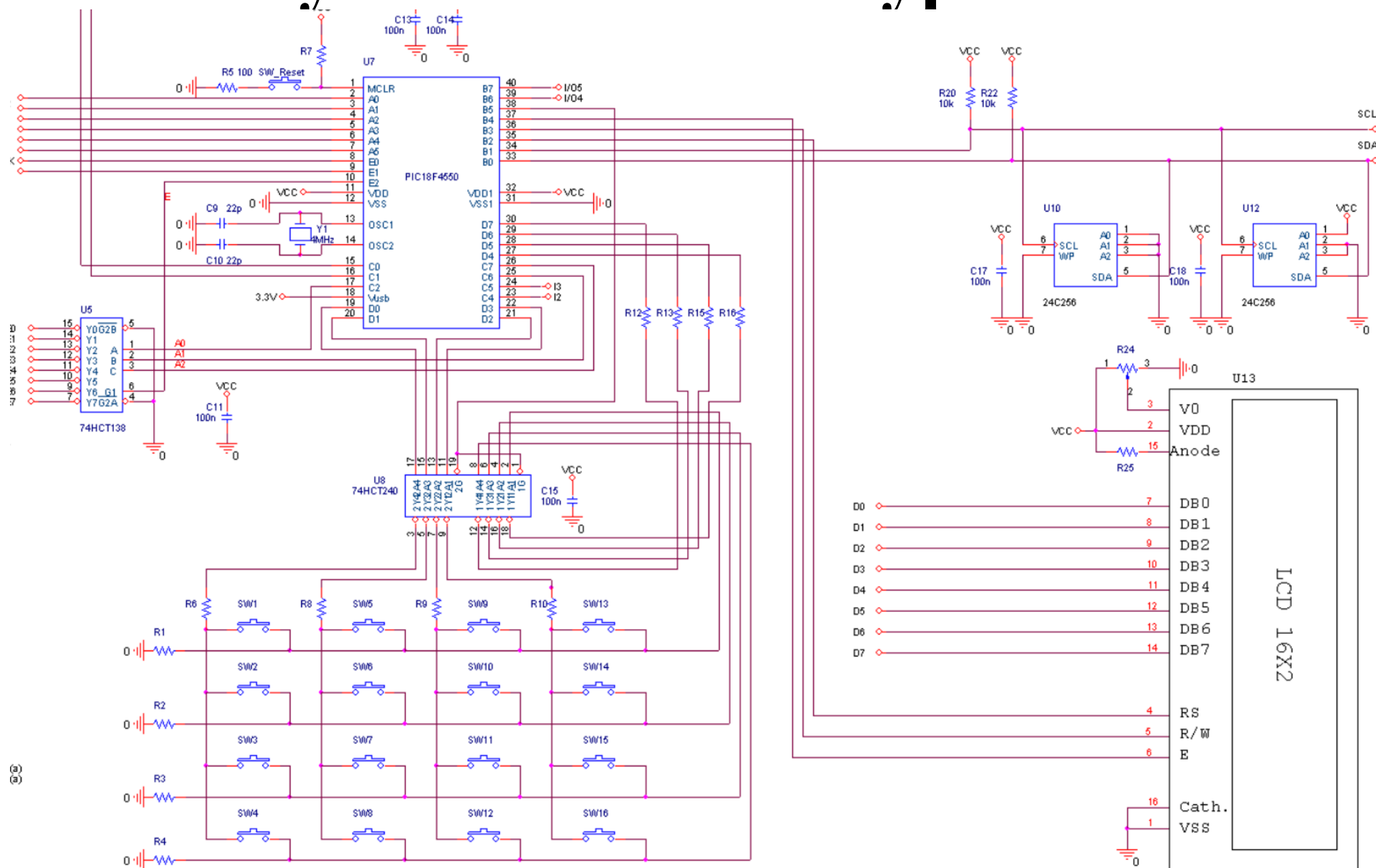
Тогава може да се добавят буфери с високоимпедансно състояние, чрез които да се “развърже” клавиатурата от другата периферия.

*Пример* – на по-следващият слайд е показан пример с 8-инвертора в един корпус 74НСТ240. Те се разрешават с 1G и 2G сигналите, т.е. чрез тези сигнали се избира високоимпедансното състояние. Добавянето на тази схема е оправдано понеже всички изводи на  $\mu\text{CU}$  са заети с други функции и двуредовия LCD екран няма къде да се свърже. Резистори R12, R13, R15, R16 са защитни, в случай че инверторите са разрешени и почне да се подават данни към LCD.

# Бутони и клавиатури



# Бутони и клавиатури



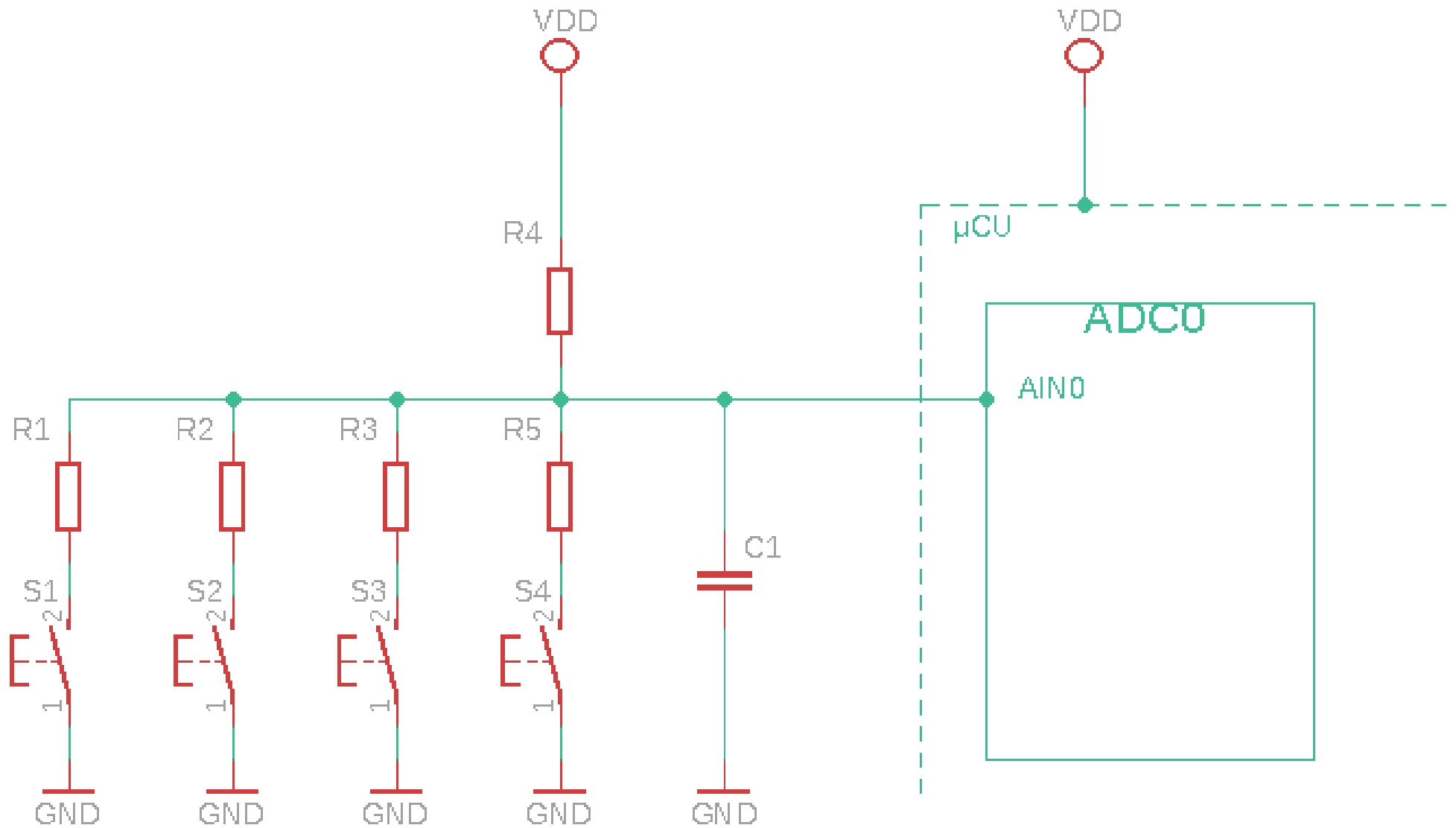
(a)  
(a)

# Бутони и клавиатури

По изключение може да се използва и вход на АЦП за четене на клавиатура с немного бутони. Схемата реализира резисторен делител, като резисторът към маса се включва от бутоните и е с различна стойност за всеки бутон. Така на входа на АЦП се подават различни напрежения, от които може да се познае кой бутон е натиснат.

Това обаче е аналогов метод за четене и е податлив на смущения.

# Бутони и клавиатури



# Тъч сензори

**Капацитивни тъч сензори** – регистрират натискане на бутон (или допир) чрез измерване на капацитета на площадка. Използват се също и за измерване нивото на течности, движение на течности и т.н.

Характерното за тъч сензорите е, че са чувствителни към смущения, защото трябва да се мерят промени в капацитета, които са части от пикофарада.

При директен допир с пръст е възможно сензорът да отчете промяна с до 50 % от началното си показание. Когато обаче се сложи зад защитен панел и солдер маска чувствителността му значително намалява. Капацитивните тъч сензори **регистрират изменения от порядъка на  $1 \div 5$  %**, което означава, че сензорът трябва да работи много добре при ненатиснат бутон.<sup>96/111</sup>



# Тъч сензори

За измерване на капацитет се използват най-често два метода:

- \*трансфер на заряд
- \*измерване на период

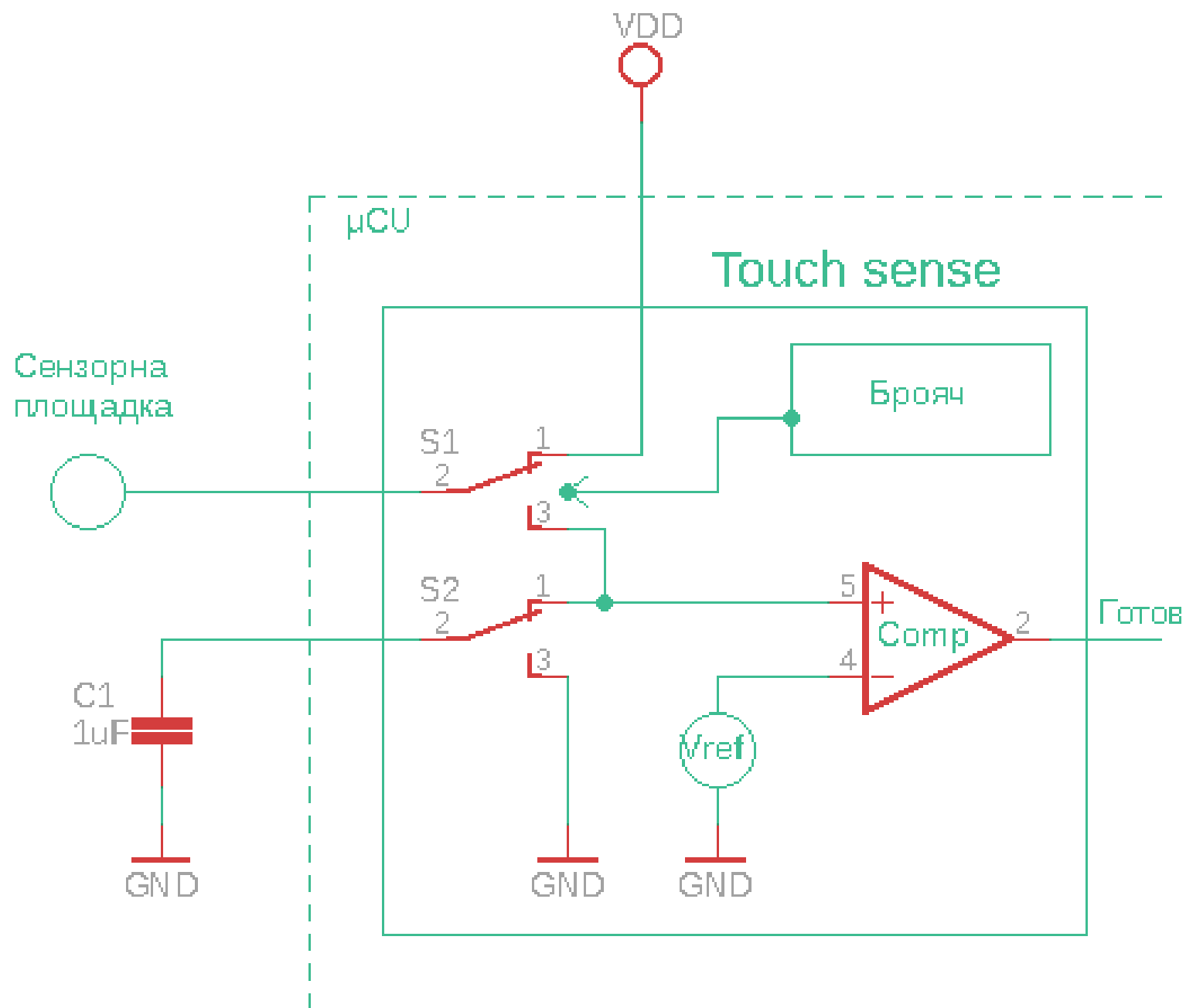
# Тъч сензори

**Метода на трансфер на заряд** (charge transfer) – сензорната площадка се зарежда до някакъв потенциал от стабилен източник на напрежение, след което площадката се свързва към кондензатор. Процесът се повтаря хиляди пъти и когато кондензаторът се зареди до определена стойност, измерването спира. Броят на зарежданията на площадката е пропорционален на капацитета ѝ.

Този вид сензори използват **варираща честота** (spread spectrum) за превключване на площадката, за да намалят влиянието на външни смущения.

*Разчита се, че смущението е с постоянна честота. Ако тази честота съвпадне с честотата на превключване, ще се насложи върху полезния сигнал. Ако честотата на превключване се променя в рамките на едно измерване, тогава и “вредния” сигнал ще се насложи за по-кратко време.*

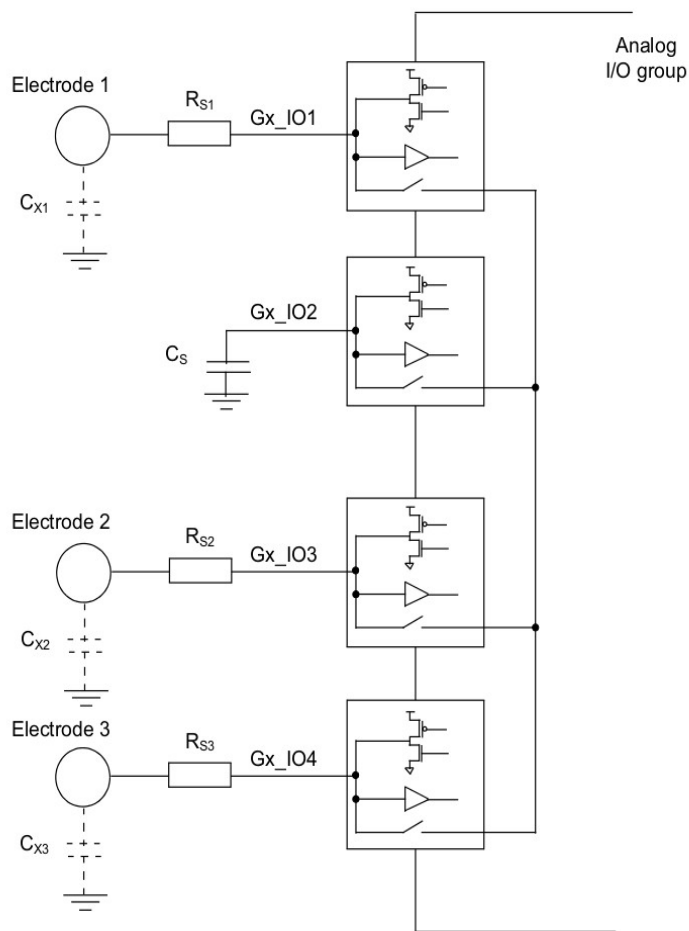
# Тъч сензори



# Тъч сензори

*Пример* – микроконтролерите от подфамилията STM32L05x и нагоре имат модул за капацитивни бутони, използващ метода с трансфер на заряда. Блокова схема на модула е дадена по-долу.

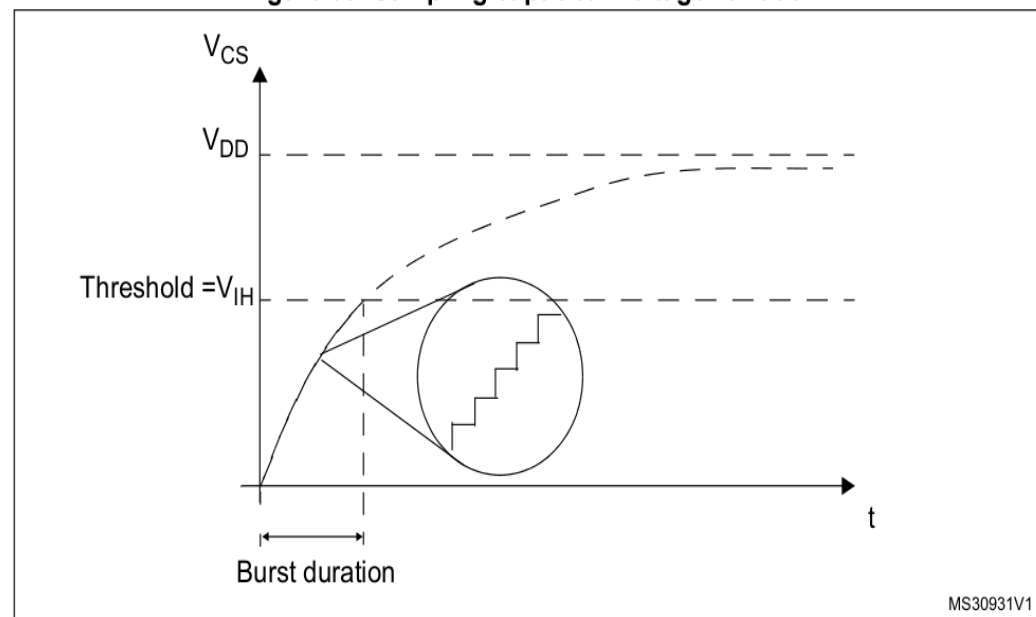
Figure 68. Surface charge transfer analog I/O group structure



MSv30930V2

The voltage variation over the time on the sampling capacitor  $C_S$  is detailed below:

Figure 69. Sampling capacitor voltage variation



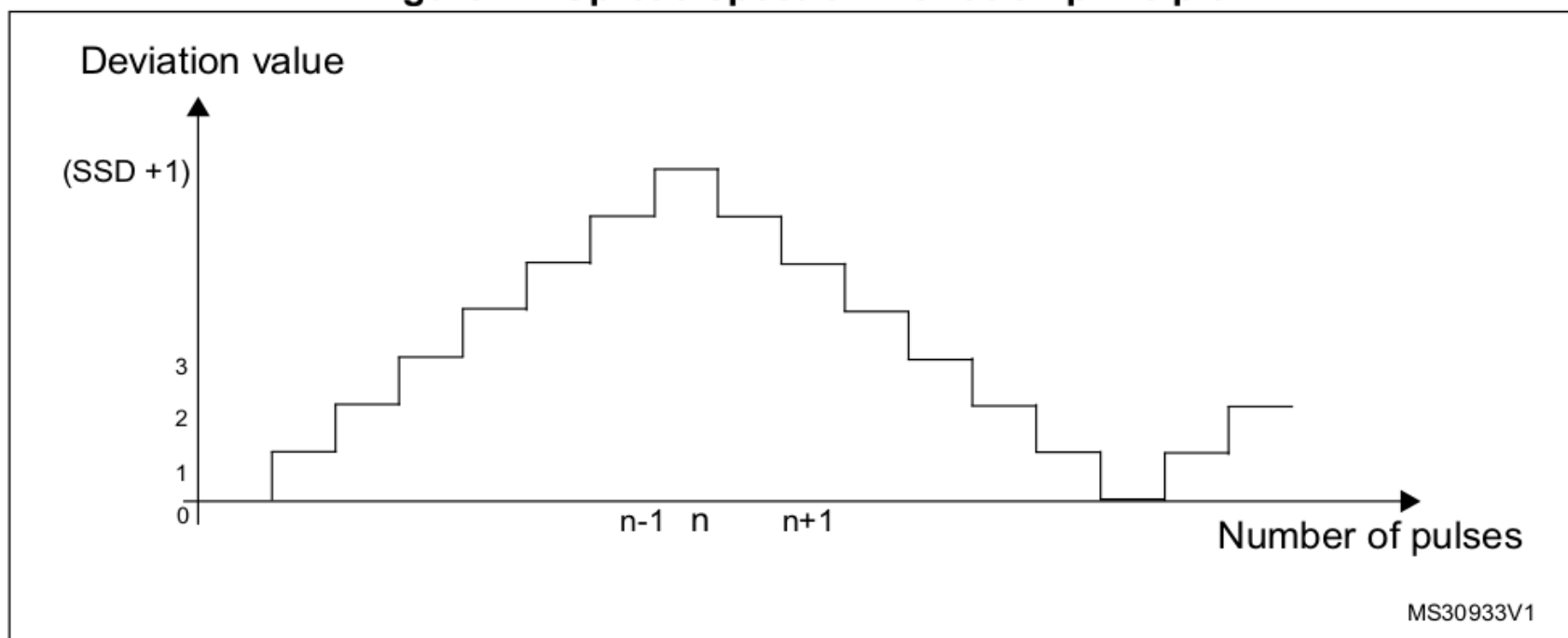
MS30931V1

$Gx\_IOy$  where  $x$  is the analog I/O group number and  $y$  the GPIO number within the selected group.

# Тъч сензори

Описаният модул (Touch Sensing Controller, TSC) има възможност и за вариране на честотата (spread spectrum).

Figure 71. Spread spectrum variation principle



The table below details the maximum frequency deviation with different HCLK settings:

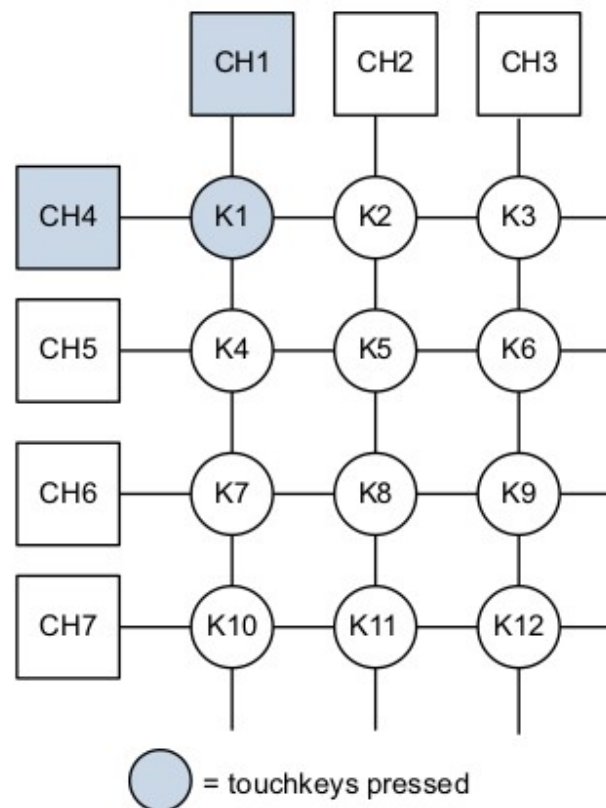
Table 74. Spread spectrum deviation versus AHB clock frequency

$f_{HCLK}$	Spread spectrum step	Maximum spread spectrum deviation
24 MHz	41.6 ns	10666.6 ns
32 MHz	27.7 ns	7111.1 ns

# Тъч сензори

Тъч сензорните модули обикновено са многоканални, което позволява да се реализират клавиатури и плъзгачи [9].

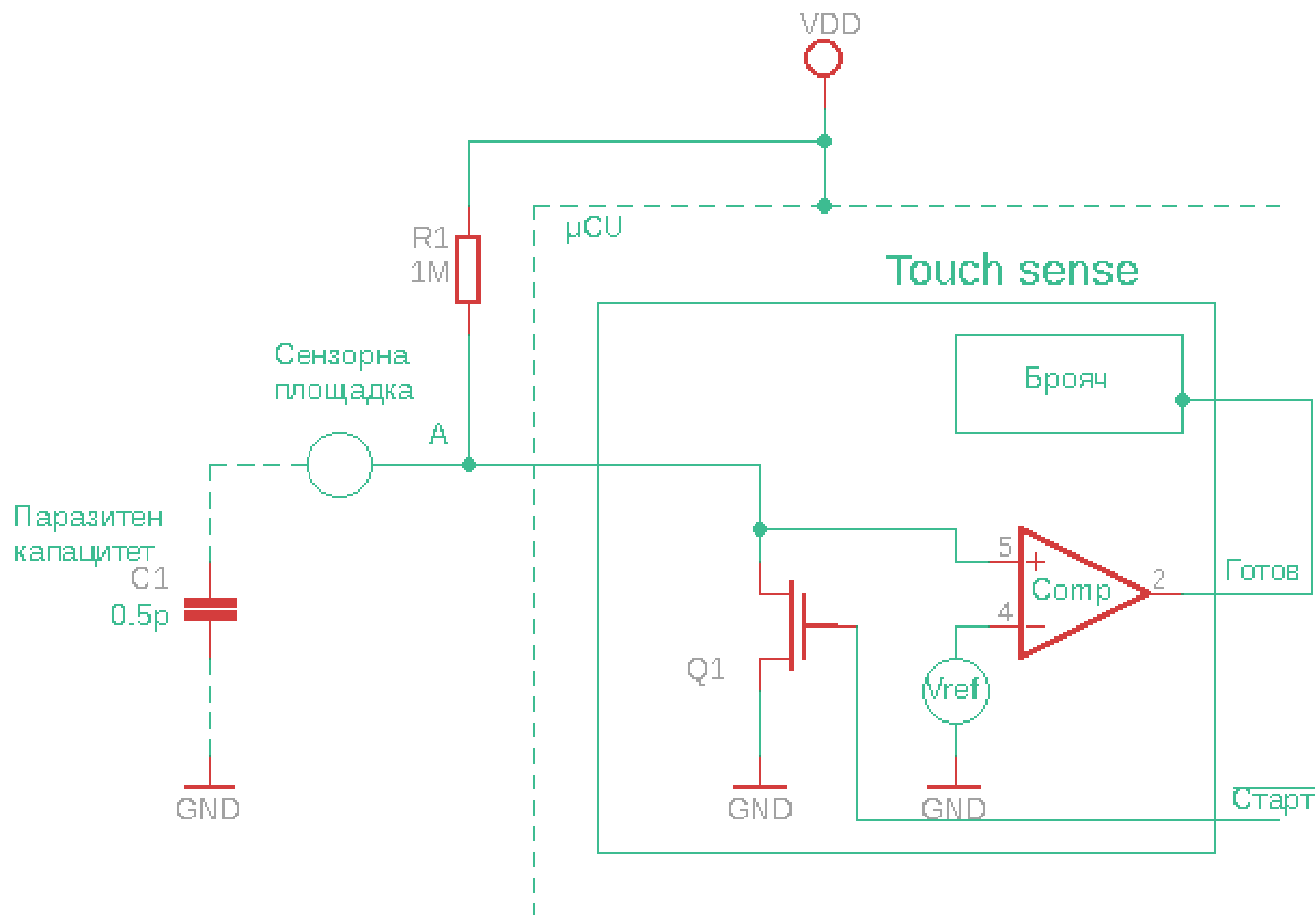
Figure 14. Simple matrix implementation



# Тъч сензори

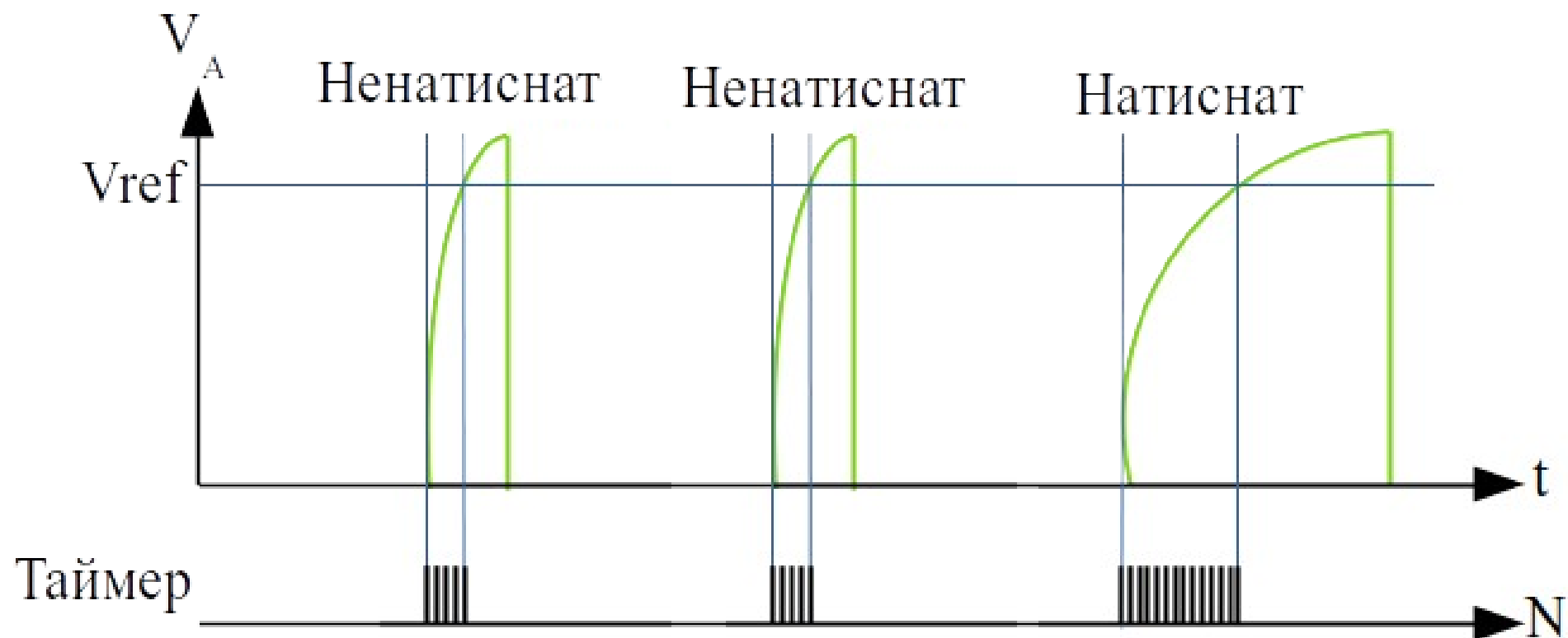
**Метода с измерване на период (RC touch sensor)** – сензорната площадка се свързва към захранване през високоомен издърпващ резистор. Паразитният капацитет на площадката и резистора формират RC верига. Измерва се времето, което ще отнеме на сигнала в средната точка да премине определен праг. Това време се мери с таймер в Capture режим.

# Тъч сензори





# Тъч сензори



# Тъч сензори

*Пример* – всички MSP430, които имат таймер и аналогов компаратор могат да реализират тъч сензор. При тях се използва модифициран вариант на метода с измерване на период – RC веригата е част от обратната връзка на генератор на правоъгълни импулси. Използват се наличните GPIO, Timer и Comparator модули.

The Capacitive Touch I/O module allows implementation of a simple capacitive touch sense application. The module uses the integrated pullup and pulldown resistors and an external capacitor to form an oscillator by feeding back the inverted input voltage sensed by the input Schmitt triggers to the pullup and pulldown control. [Figure 9-1](#) shows the capacitive touch I/O principle.

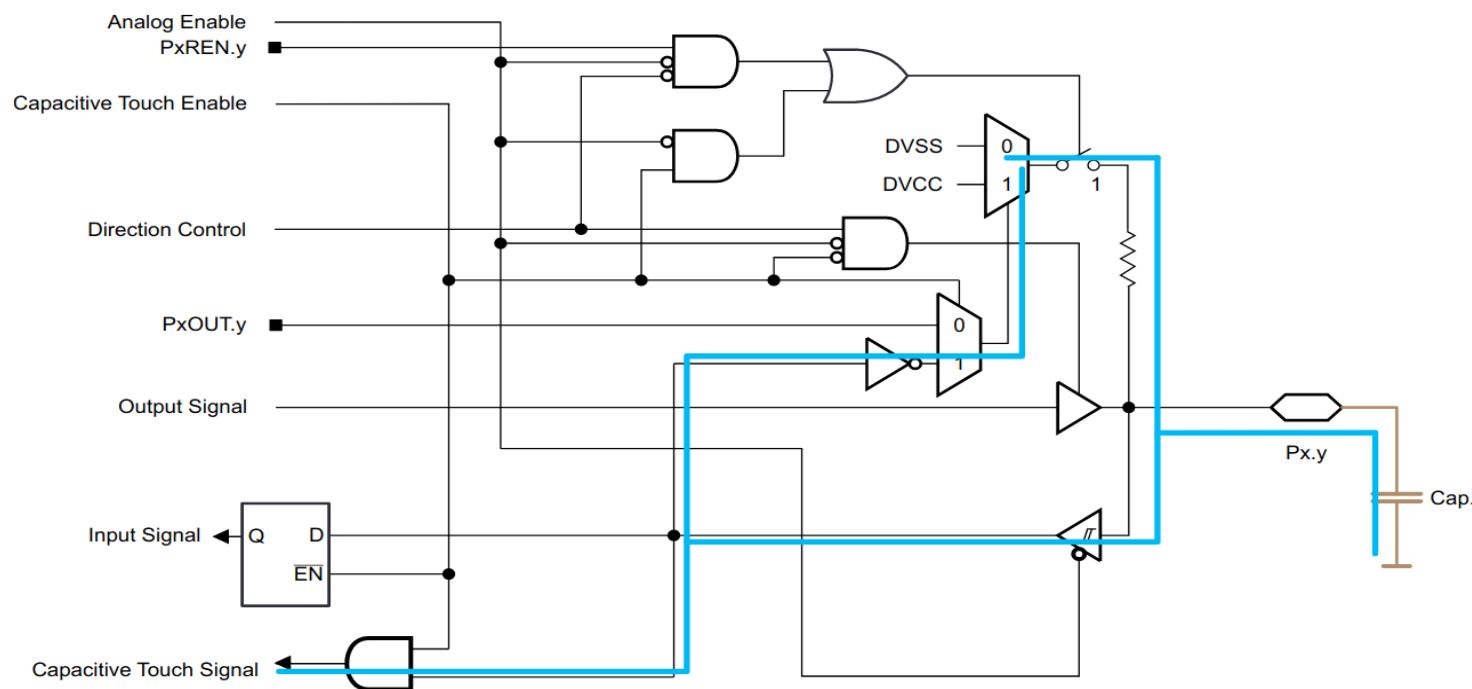


Figure 9-1. Capacitive Touch I/O Principle

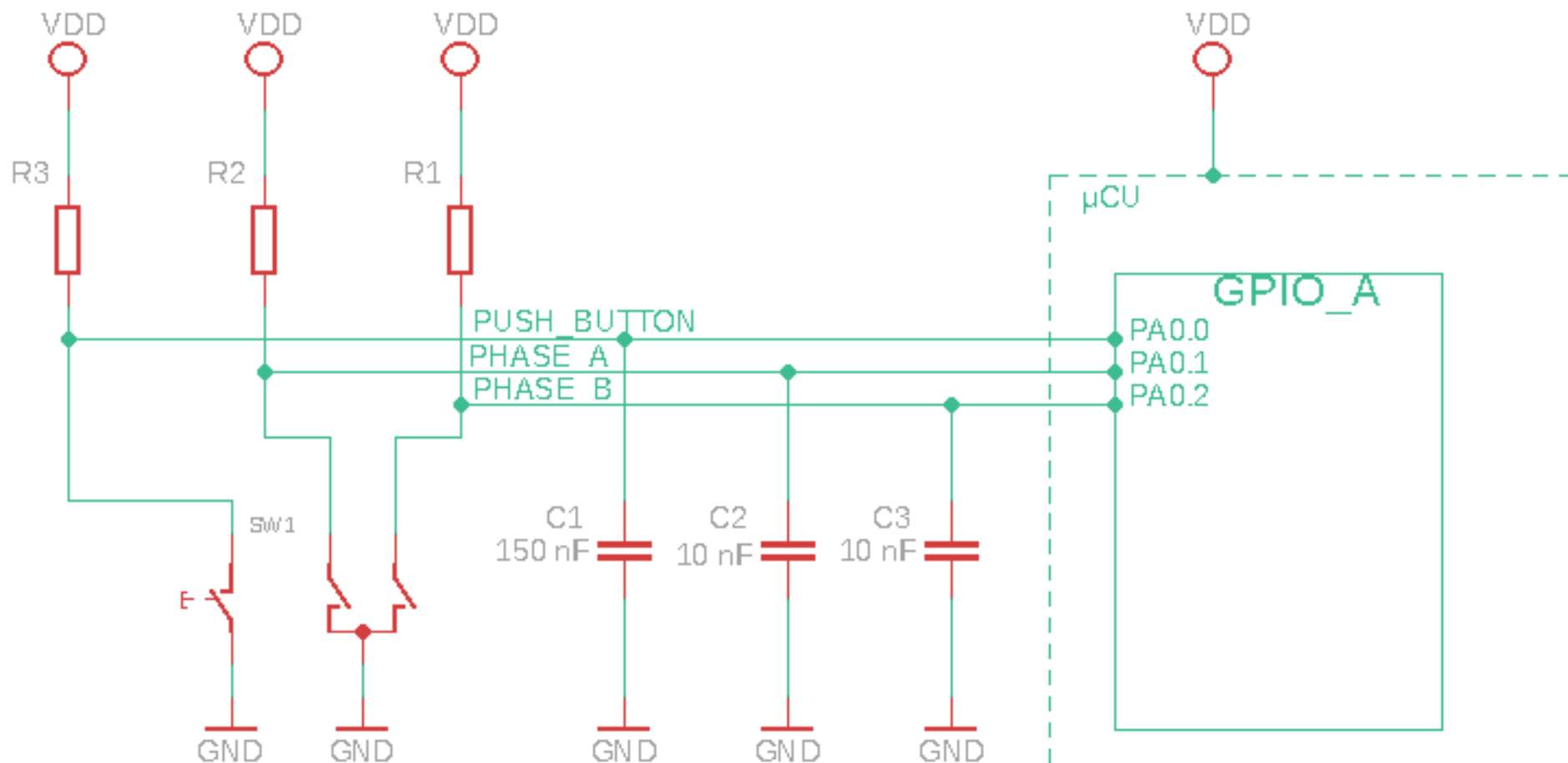
# Ротационни енкодери

**Ротационен енкодер** — електромеханичен елемент, който преобразува ъгъла на завъртане на ос в цифров вид. Най-често се използва за определяне позицията на ротора на електромотори и като заместител на потенциометри в електро-табла.

Ротационните енкодери може да се четат със специализирани модули от  $\mu$ CU като QE1 и ESI (виж лекцията за таймери) или от GPIO входове, работещи с прекъсване.

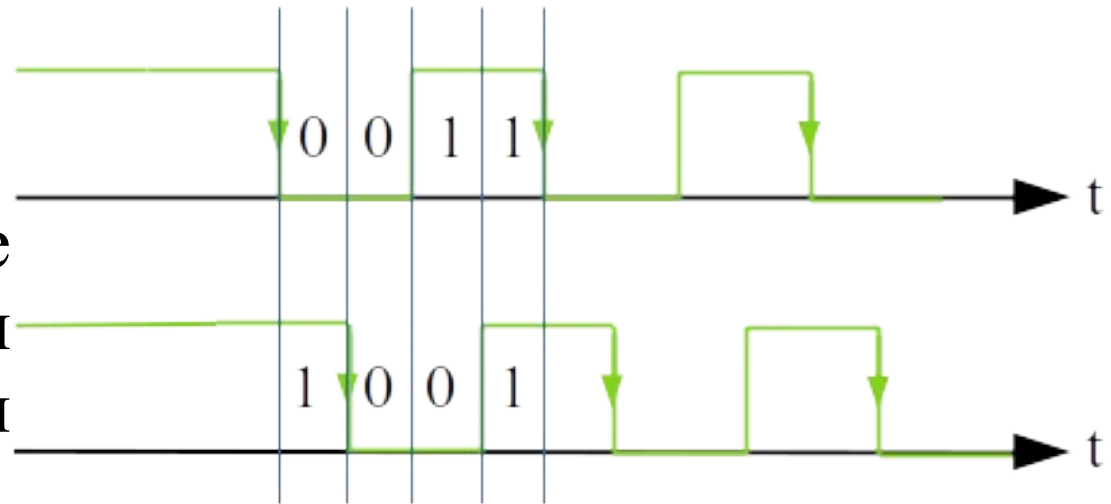
# Ротационни енкодери

Схема на свързване на механичен ротационен енкодер. Както при бутоните, така и тук трябва да се филтрират микромеханичните притрепвания на контакта. Много енкодери позволяват да се натисне оста им, която е свързана към бутон в единия си край. Такива енкодери изискват 3 GPIO входа.



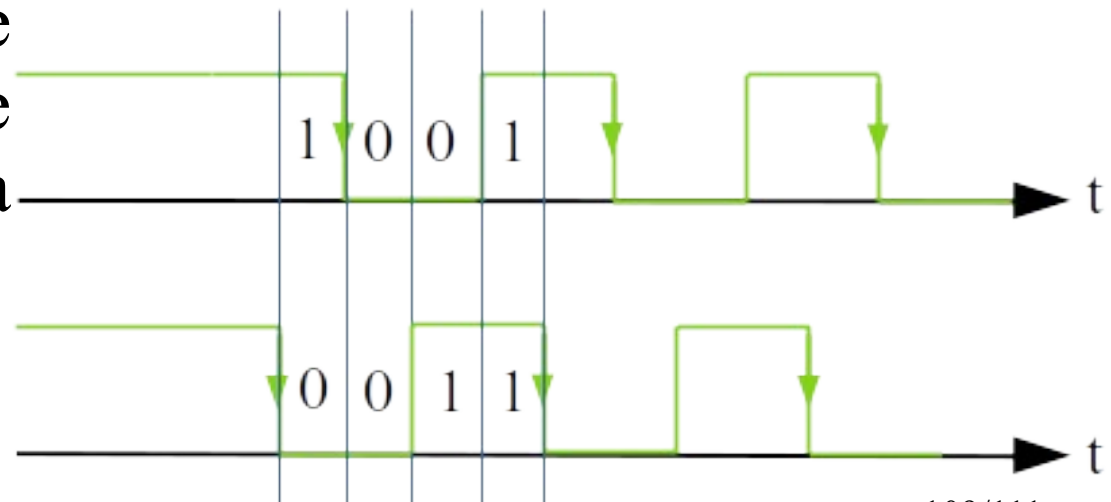
# Ротационни енкодери

Въртене надясно



По първия падащ фронт може да се разбере **накъде** се върти енкодера (наляво или надясно).

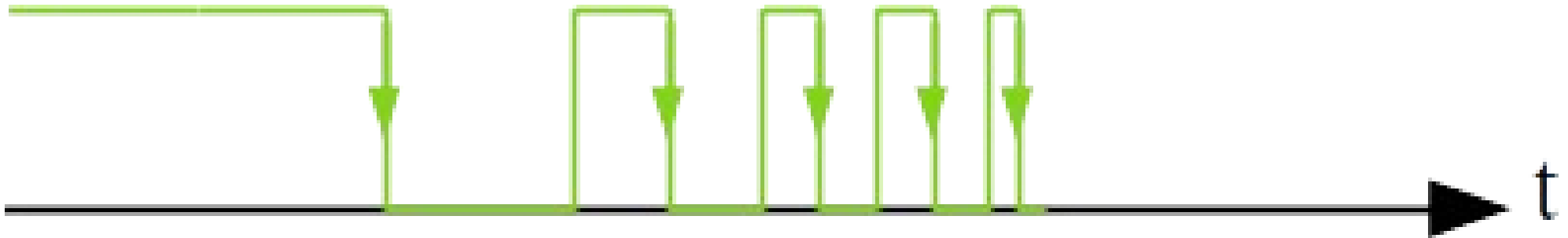
Въртене наляво



По броя на падащите фронтове може да се разбере **колко** завъртания са направени.

# Ротационни енкодери

Осцилограмите от миналия слайд са идеализирани. Реално, обаче, **сигналят е апериодичен**. Колкото по-бързо се върти енкодера, толкова по-кратък става периода на фазите.  $\mu\text{CU}$  трябва да може да регистрира дори най-бързите завъртания. На осцилограмата по-долу е дадена една фаза на енкодер, който се завърта първоначално бавно, след това бързо.



# Литература

- [1]Г. Михов, “Цифрова схемотехника”, ТУ-София, 1999.
- [2]L-115WGYW Datasheet, Kingbright, 2003.
- [3]CLM1B-RKW/AKW Product Family Datasheet, CREE, 2011.
- [4][http://www.fujitsu.com/downloads/MICRO/fma/pdf/LCD\\_Backgrounder.pdf](http://www.fujitsu.com/downloads/MICRO/fma/pdf/LCD_Backgrounder.pdf)
- [5]HD44780U (LCD-II), Hitachi, ADE-207-272(Z), rev. 0.0.
- [6]MC21605A6WD-BNMLW Specification, MIDAS, 2011.
- [7]MSP430FR6989, User’s Guide SLAU367N, Texas Instruments, 2017.
- [8]SEPS525, Syncoam Co., Ltd, 2006.
- [9]“Guidelines for designing touch sensing applications with surface sensors”, AN4312, ST Microelectronics, 2016.