EMBEDDED

л.3
Память та периферія мікропроцесора та мікроконтролера.

Палій Святослав

Український Католицький Університет
APPLIED SCIENCES FACULTY

Організація пам'яті

Основні області пам'яті

Флеш-пам’ять

Флеш-пам'ять забезпечує зберігання початкових даних і програм. Вміст зберігається при відсутності живлення. Читання швидке, запис не такий швидкий. Прямий запис неможливий тільки через контролер запису. Як правило запис цілими блоками, відсутня можливість запису окремого байта.

RAM

SRAM використовується для тимчасового зберігання даних під час виконання програм. Вміст втрачається без живлення. Запис та читання швидкі і прямі.

Регістри периферійних пристроїв

Регістри периферійних пристроїв забезпечують контроль над зовнішніми компонентами. В гарвардських архітектурах часто представлені в адресному просторі SRAM. Права доступу часто індивідуальні, можливі варіанти тільки читання, тільки запису чи повного доступу. Деколи зміст бітів на запис та читання є різним.

А також (опційно)

ROM

Забезпечує зберігання початкових даних та програм тільки для читання. Як правило формується (програмується) виробником в процесі виробництва чіпа. Читання швидке. Запис неможливий. Дешевша у виробництві порівняно з флеш-пам’яттю.

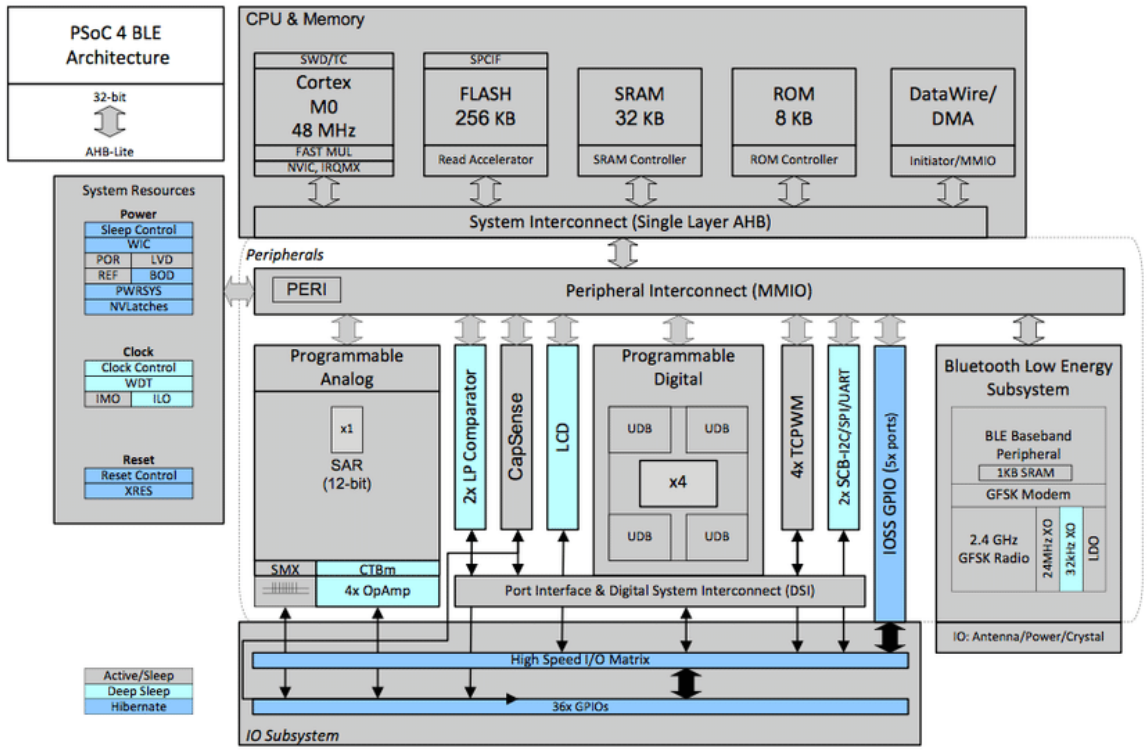
EEPROM

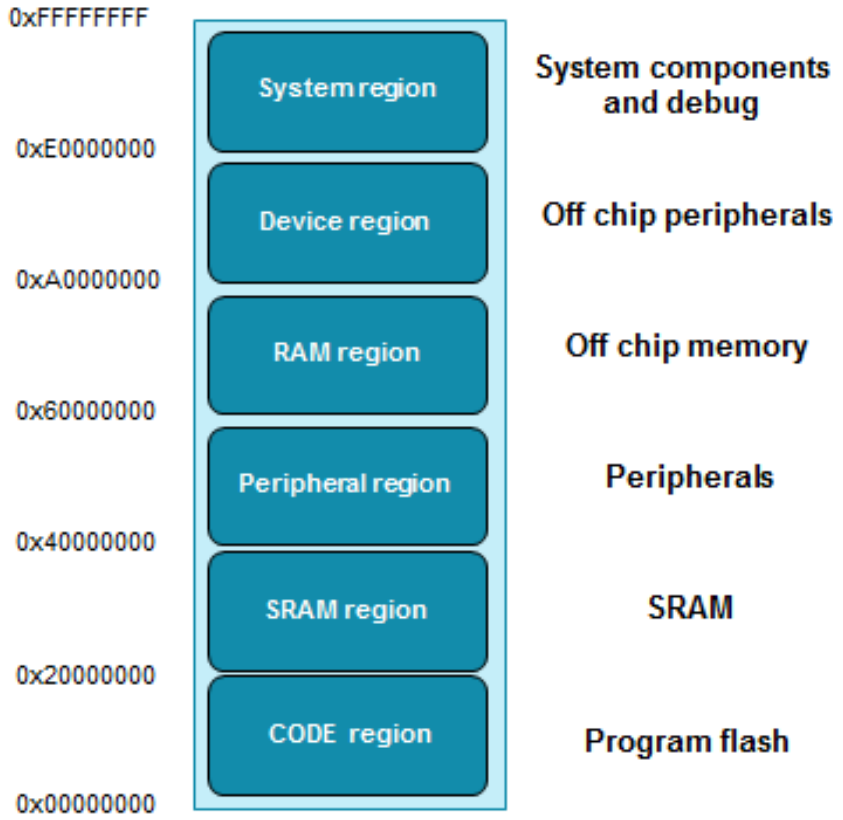
Подібна до флеш-пам’яті, але на відміну має більший ресурс. Доступна для побайтового запису, доступ на запис (а деколи і на читання) через контролер. Повільна. Використовується для зберігання конфігураційних даних, налаштувань користувача та інших невеликих обсягів даних, які потребують частого оновлення.

Конфігурація периферії

Дані для конфігурації конфігурованої периферії. Біти режиму роботи та захисту мікроконтролерів.

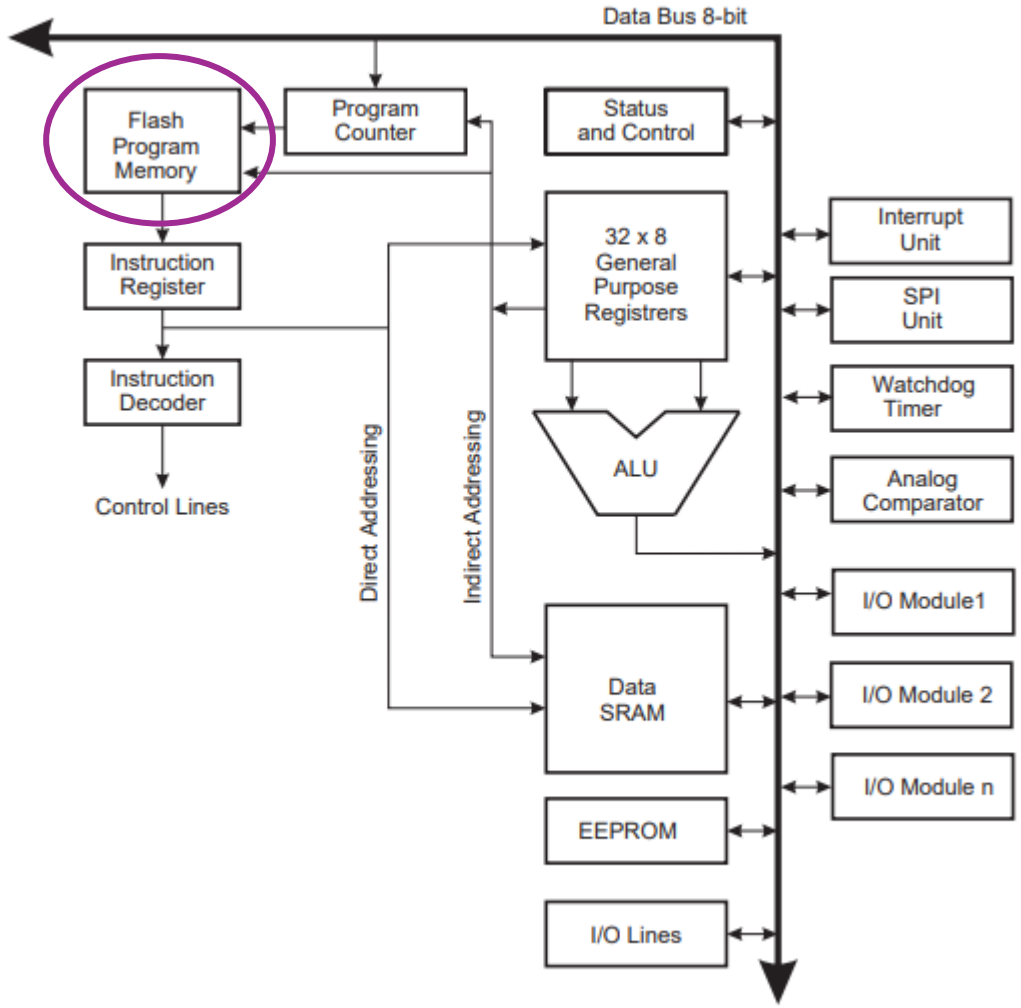
Приклад: ARM Cortex M0





Cortex-M3 core and its features - Architectures and Processors blog - Arm Community blogs - Arm Community

Приклад: ATmega 328P



Простір памяті програм окремий.

Простір I/O містить 64 адреси для периферійних функцій, таких як регістри управління, SPI, і т.д. Пам'ять вводу/виводу може бути доступна як місця в просторі даних, за адресами 0x20 - 0x5F. Крім того, ATmega8U2/16U2/32U2 має розширений простір вводу/виводу від 0x60 до 0xFF в SRAM

https://ww1.microchip.com/downloads/en/DeviceDoc/doc7799.pdf

Запис до флеш (ATmega328P)

Прямий запис неможливий:

```
a = 5;
```

```
void write_page_to_flash(uint32_t address, uint8_t *data) {
    // Вимикаємо переривання
    cli();

    // Стираємо сторінку Flash пам'яті
    boot_page_erase(address);
    boot_spm_busy_wait();

    // Заповнюємо сторінку даними
    for (uint16_t i = 0; i < PAGE_SIZE; i += 2) {
        uint16_t word = data[i] | (data[i + 1] << 8);
        boot_page_fill(address + i, word);
    }

    // Записуємо сторінку Flash пам'яті
    boot_page_write(address);
    boot_spm_busy_wait();

    // Вмикаємо переривання
    sei();
}
```

Arduino

```
int a;

void write() {
  // Записуємо значення 42 до адреси змінної
  //'a' у Flash пам'яті
  EEPROM.write(&a, 42);
}
```

Хоча бібліотека EEPROM зазвичай використовується для роботи з EEPROM, вона також може бути використана для запису даних до Flash пам'яті.

Запис до флеш (PSoC4)

```
#define FLASH_ROW_SIZE 128  // Розмір сторінки Flash пам'яті в байтах

void write_page_to_flash(uint32_t rowNumber, uint8_t *data) {

    // Вимикаємо переривання
    CyGlobalIntDisable;

    // Записуємо сторінку до Flash пам'яті
    CySysFlashWriteRow(rowNumber, data);

    // Вмикаємо переривання
    CyGlobalIntEnable;
}
```

Побайтна емуляція EEPROM

```
int main(void) {
    uint8_t eeprom_address = 0x00; // Адреса в EEPROM
    uint8_t eeprom_data = 0x42;    // Дані для запису

    // Ініціалізуємо компоненти
    Em_EEPROM_Start();

    // Записуємо байт до EEPROM
    Em_EEPROM_Write(&data, address, 1);
}
```

Доступ до RAM та периферії

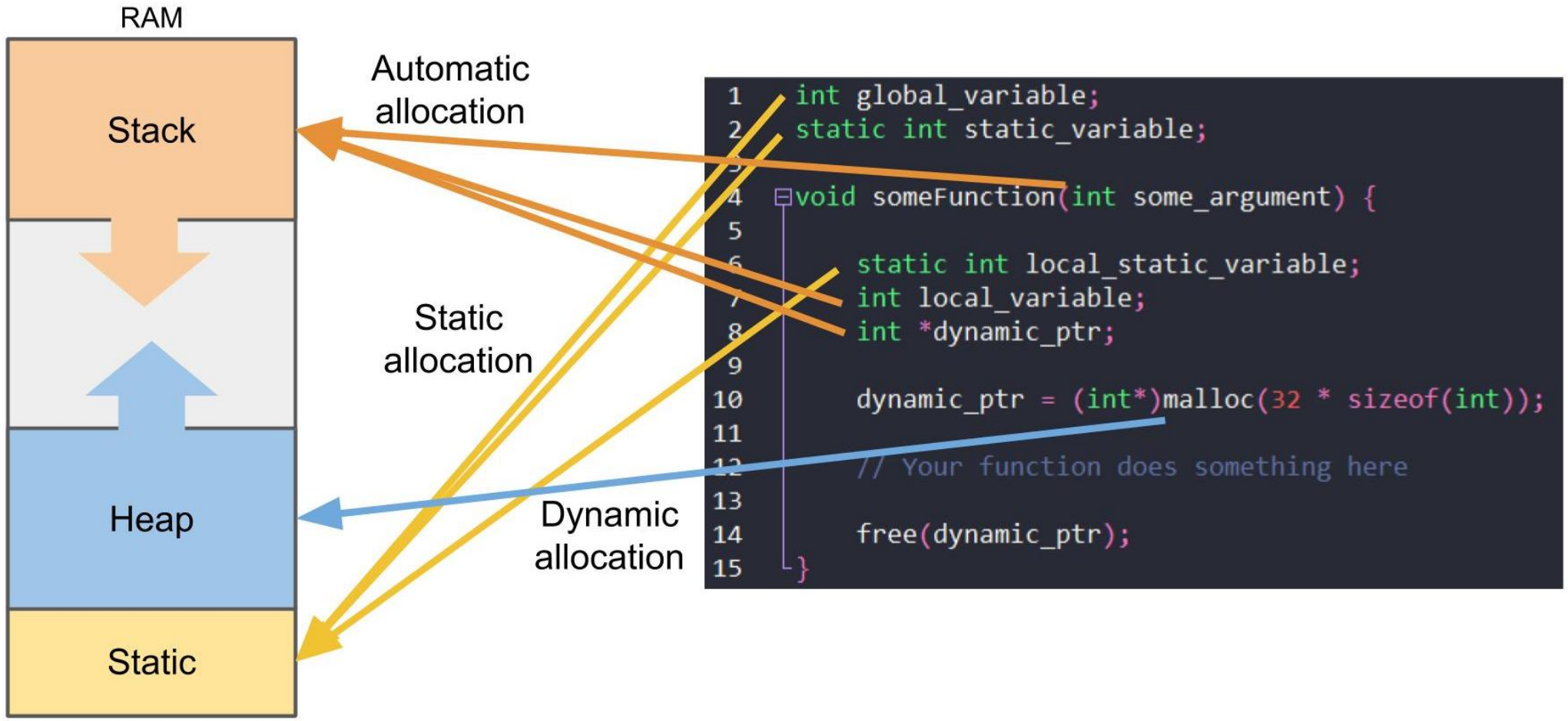
До RAM доступ вільний і безпосередній для запису та читання.

```
// Безпосередній доступ
varInRam = 5;
secondVar = varInRam;
```

Периферійні регістри можуть мати обмеження на запис чи читання.
Також можливі випадки що значення котрі пишуться та читаються не зовсім ідентичні.

```
// Безпосередній доступ
ioReg = 5;
varInRam = ioReg;

// в деяких випадках операції читання-модифікації-запису дають неочікувані результати
ioReg = ioReg;
ioReg |= ioReg;
```

Логічна організація памяті



FreeRTOS Memory Management

Автор: Шон Хаймел та Digi-Key Electronics.

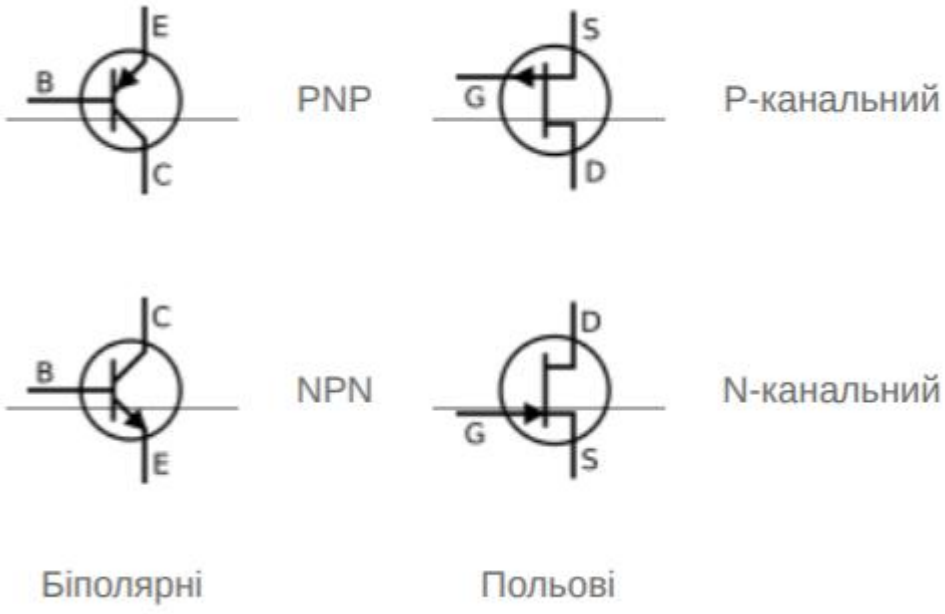
Приклад для дискусії

```
bool btnPressed(int btnNum)
{
  const int buttonPin[2] = {BTN1_PIN, BTN2_PIN};
  int reading[2] = {HIGH, HIGH};
  static int buttonState[2] = {HIGH, HIGH};
  static int lastButtonState[2] = {HIGH, HIGH};
  static int lastDebounceTime[2] = {0,0};
  int debounceDelay = 50;
  bool retVal = false;

  if (btnNum > 1)
  {
    return false;
  }

  reading[btnNum] = digitalRead(buttonPin[btnNum]);

  // Якщо стан кнопки змінився
  if (reading[btnNum] != lastButtonState[btnNum])
  {
    lastDebounceTime[btnNum] = millis();
  }

  // Якщо пройшло достатньо часу з моменту останньої зміни стану кнопки
  if ((millis() - lastDebounceTime[btnNum]) > debounceDelay)
  {
    // Якщо стан кнопки змінився
    if (reading[btnNum] != buttonState[btnNum])
    {
      buttonState[btnNum] = reading[btnNum];

      // Якщо кнопка натиснута
      retVal = (buttonState[btnNum] == LOW);
    }
  }
  lastButtonState[btnNum] = reading[btnNum];
  return retVal;
}
```

Базовий ввід/вивід I/O pin

Транзистор

Транзистор (transfer, resistor, той що переносить опір)- напівпровідниковий елемент електронної техніки, який дозволяє керувати струмом, що протікає крізь нього, за допомогою зміни вхідної напруги або струму, поданих на базу, або затвор. Невелика зміна вхідних величин, може призводити до суттєво більшої зміни вихідної напруги та струму.

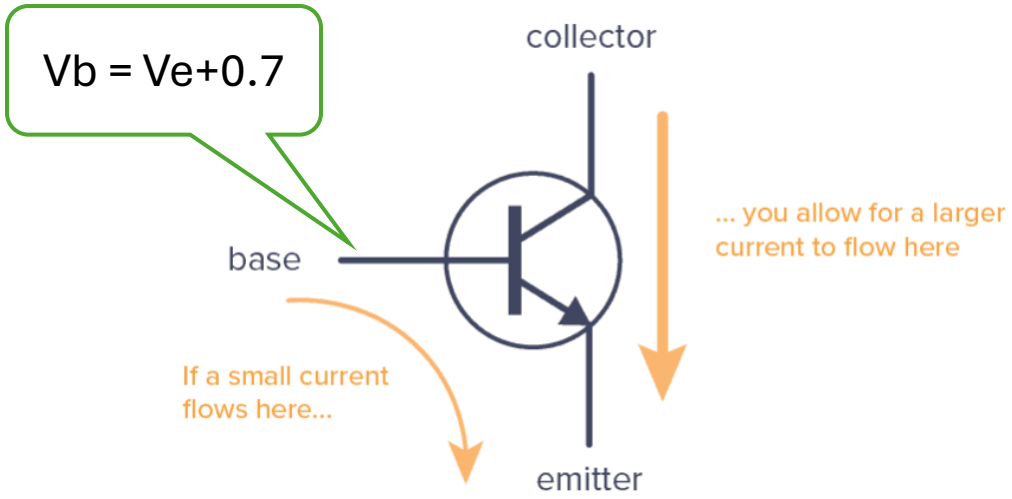


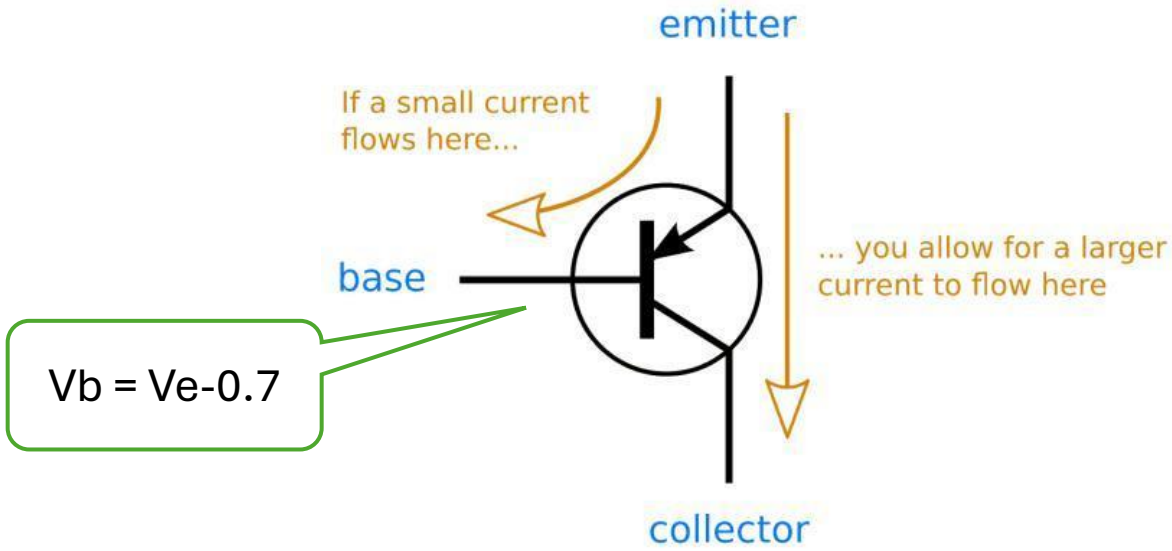
Транзистор має два основні застосування: як підсилювач і як перемикач.

Підсилювальні властивості транзистора зв'язані з його здатністю контролювати великий струм за допомогою малого струму. Таким чином, малі зміни величини сигналу в одному електричному колі, можуть відтворюватися з більшою амплітудою в іншому колі.

Використання транзистора як перемикача пов'язане з тим, що приклавши відповідну напругу або струм, можна максимально можливо відкрити або закрити (заперти) транзистор. Цю властивість використовують для керування логічними сигналами.

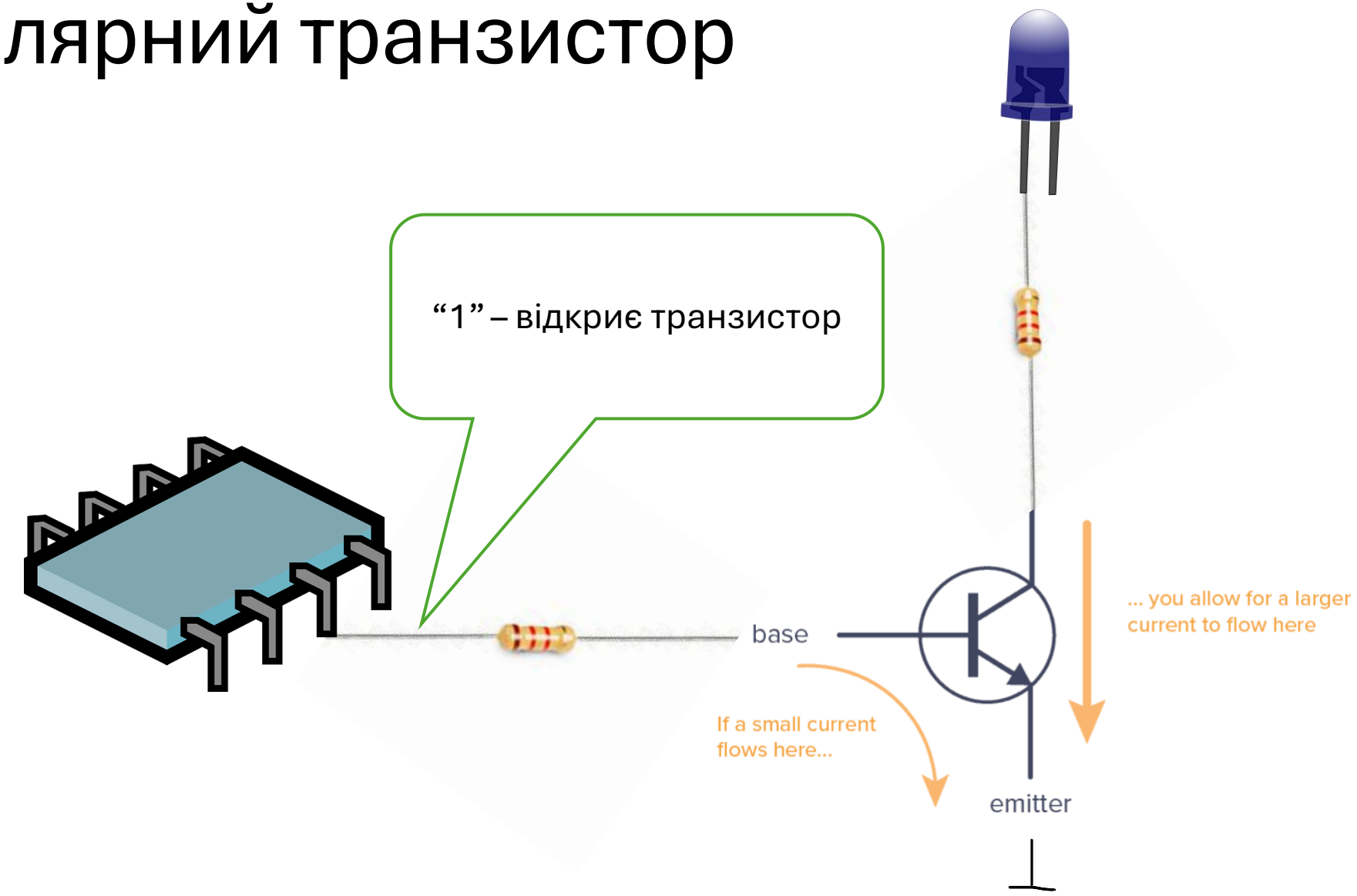
Біполярний транзистор



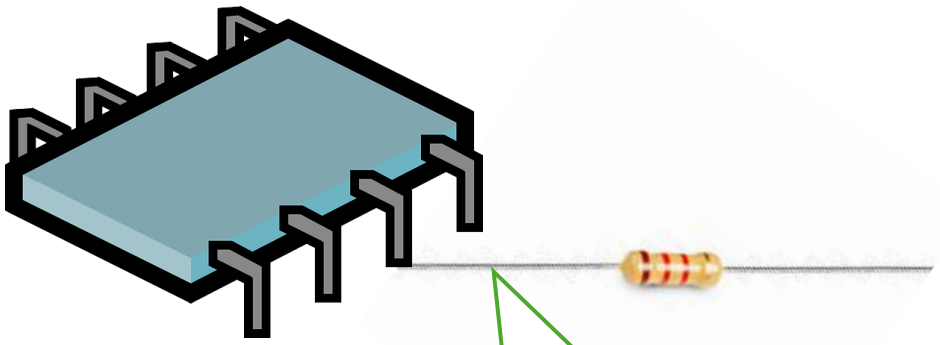


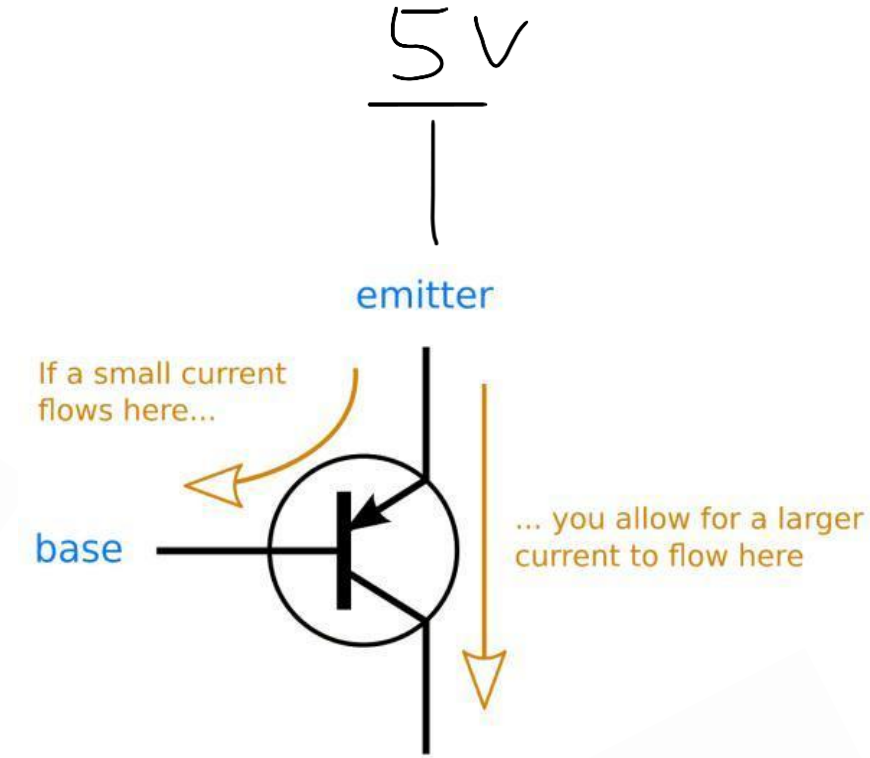
PNP Transistor – How Does It Work? - duino
by Ibrar Ayyub

Біполярний транзистор



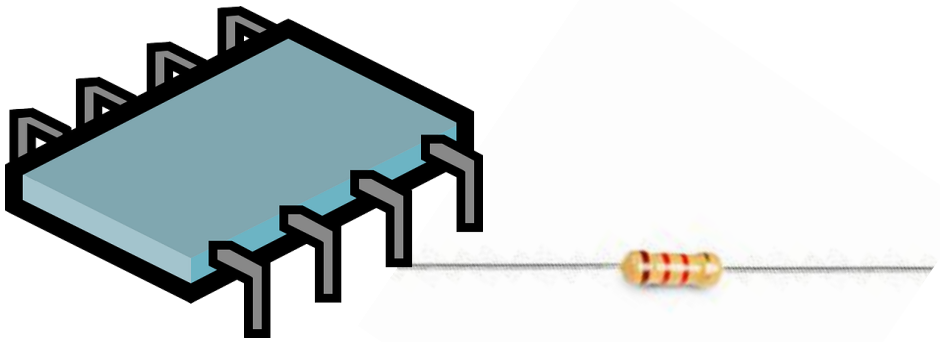
Біполярний транзистор

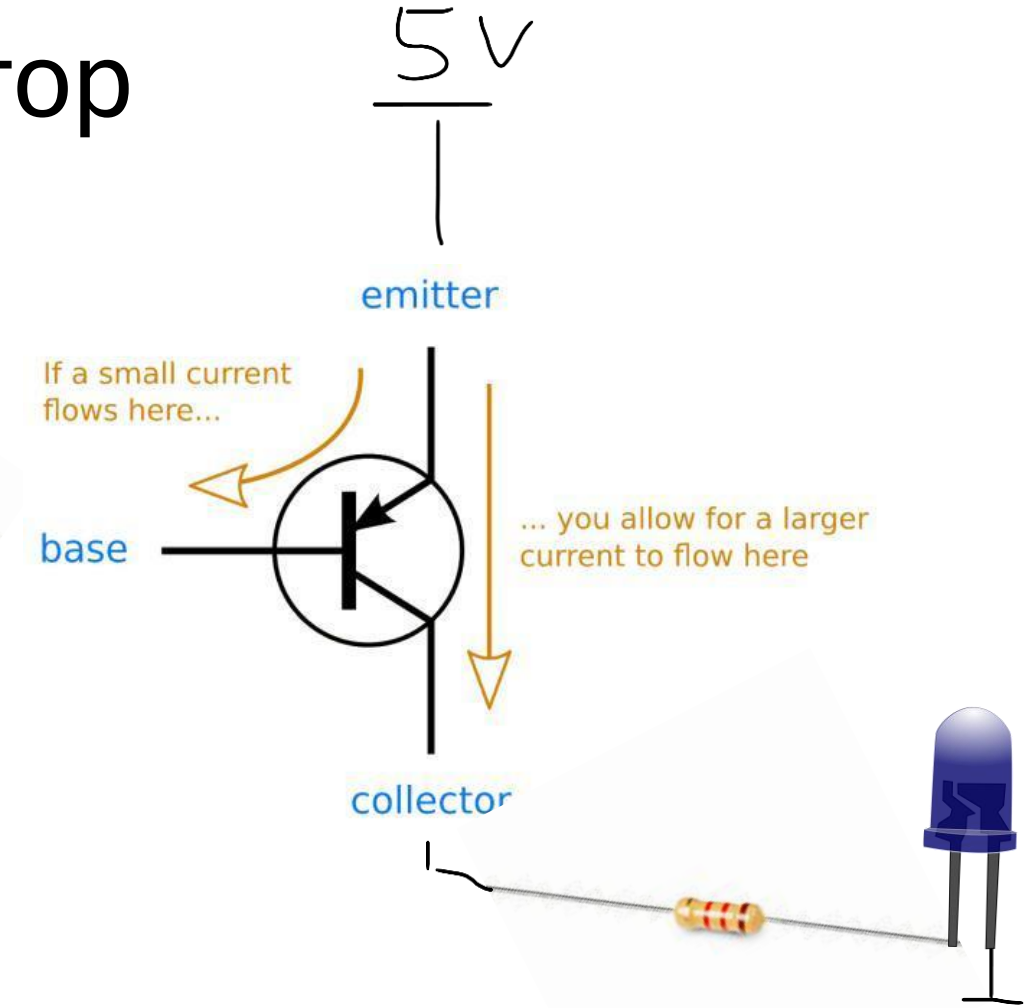




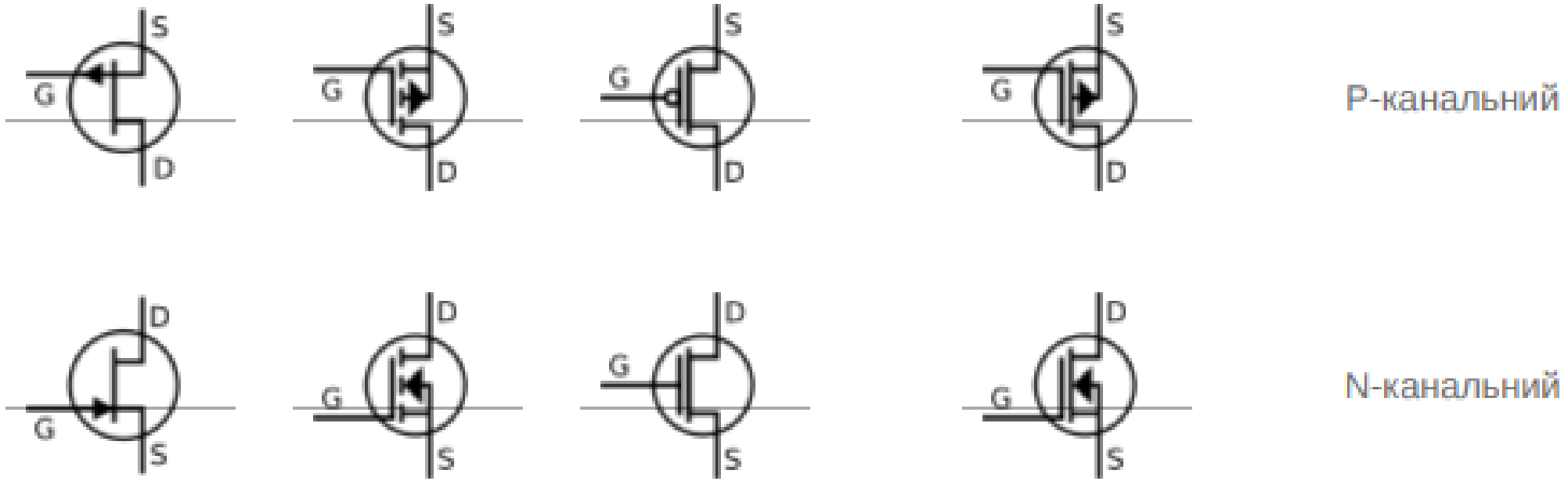
“0” - відкриє транзистор

Біполярний транзистор





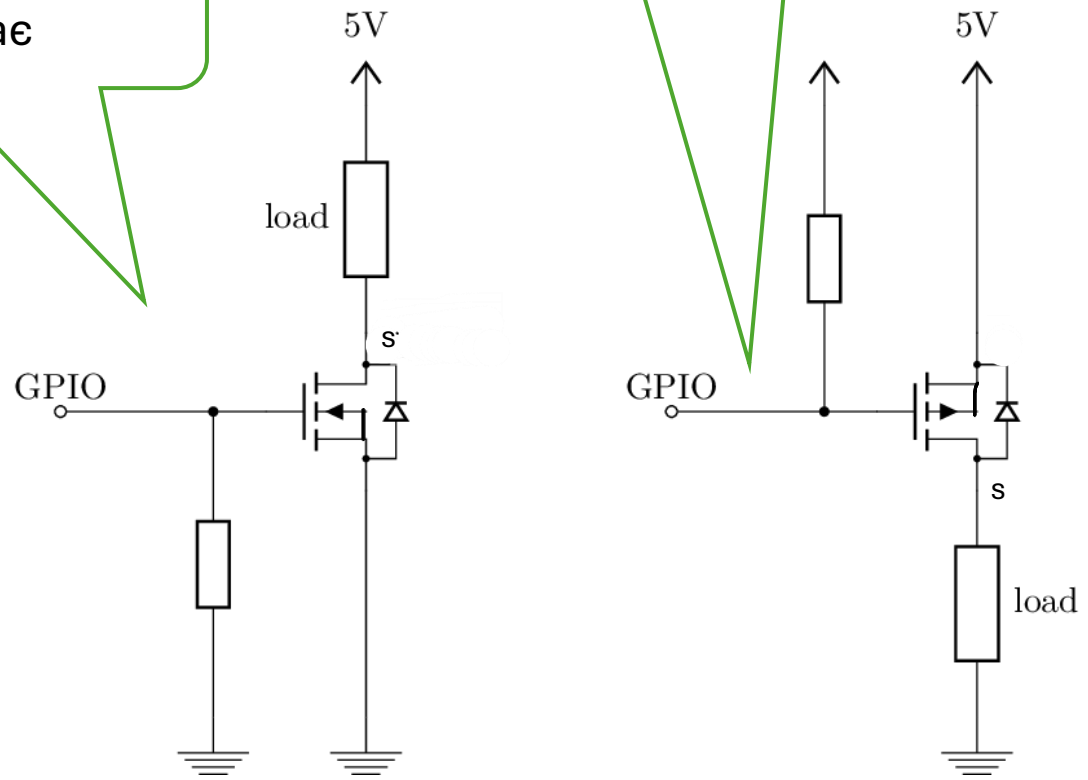
Польовий транзистор



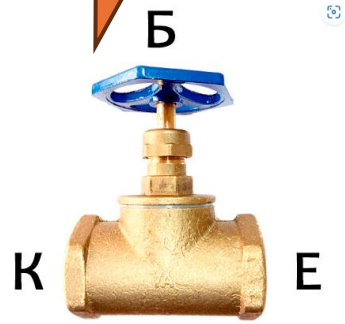
Польовий транзистор

N-канальний
“1” відкриває

P-канальний
“0” відкриває

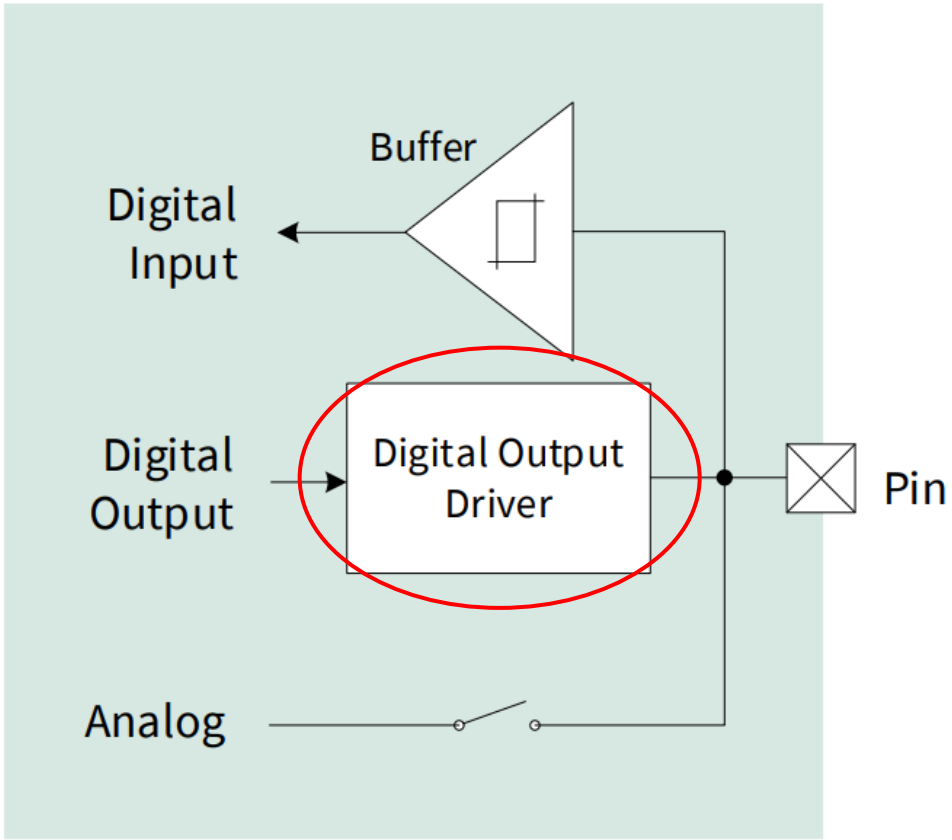


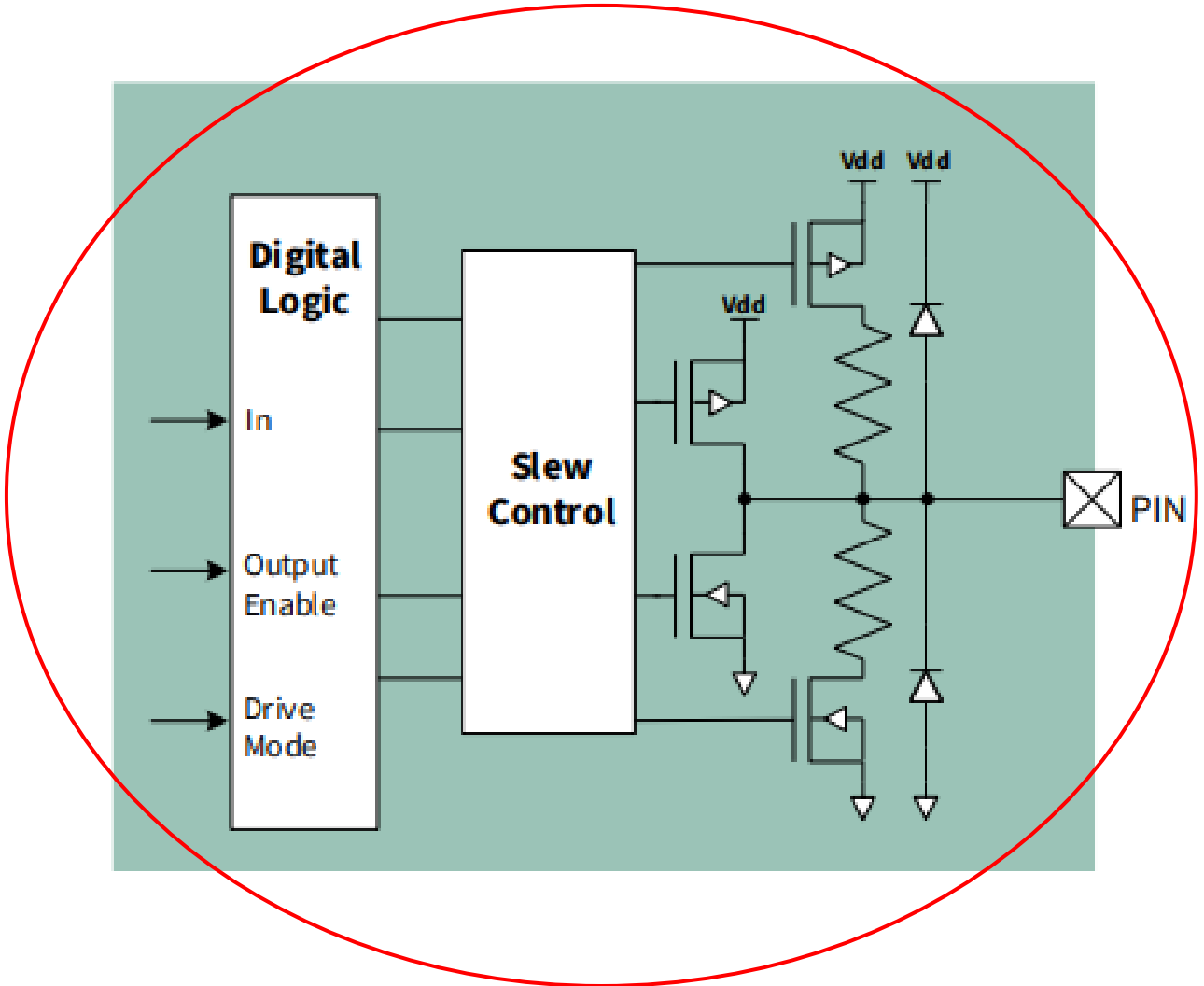
Прикладати напругу



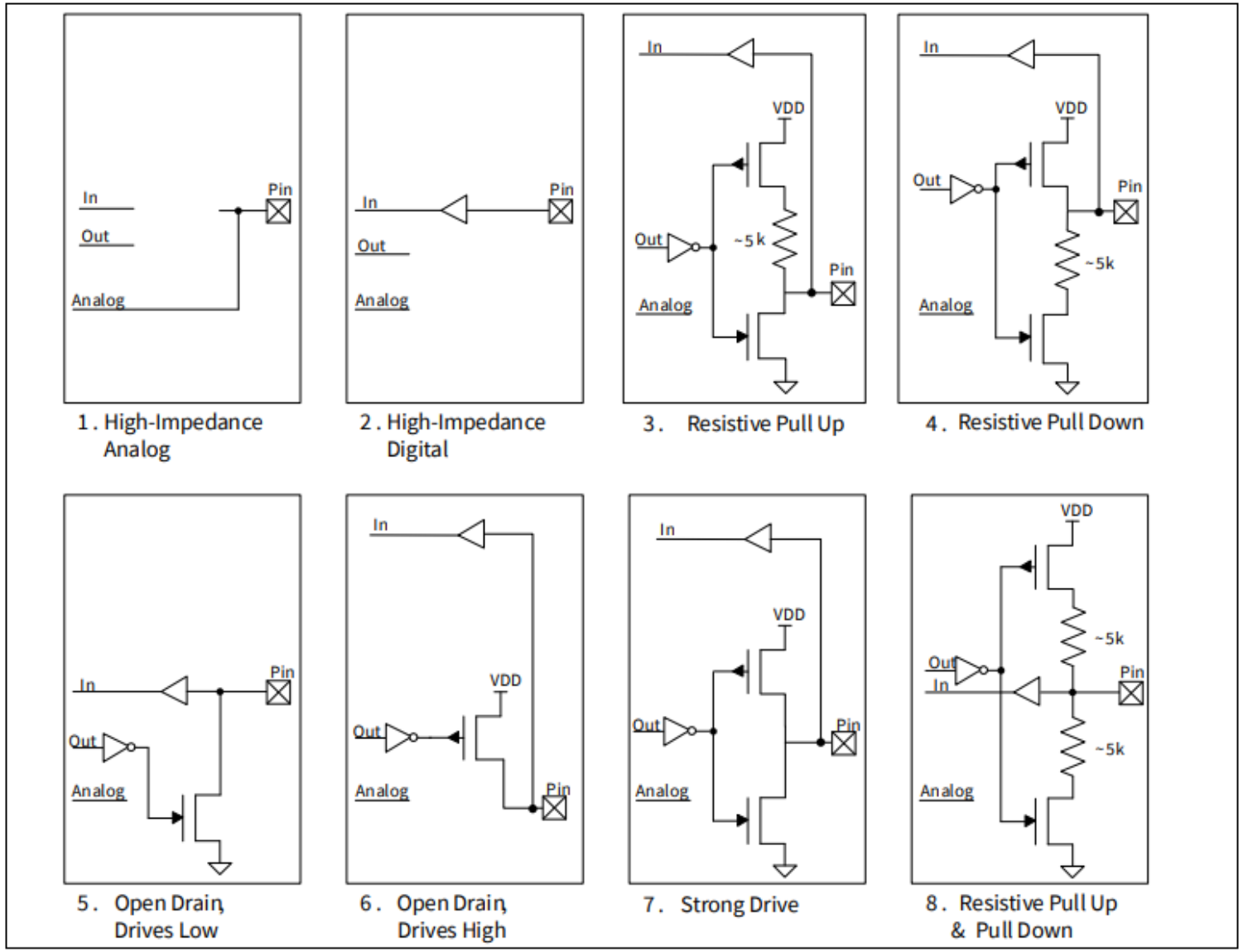
Структура вводу виводу

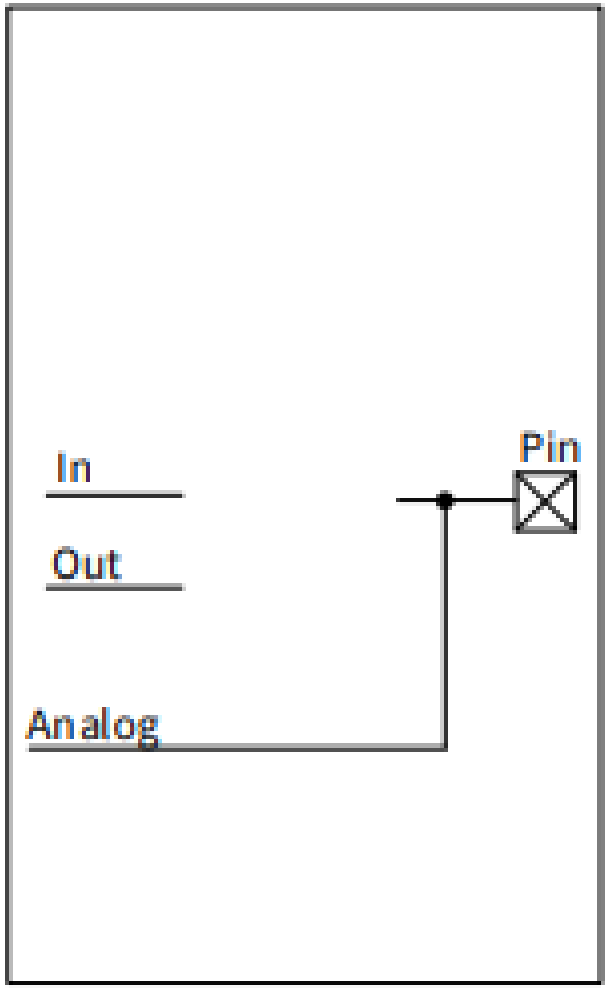
AN86439 - PSoC™ 4 MCU - Using GPIO pins - Infineon Technologies



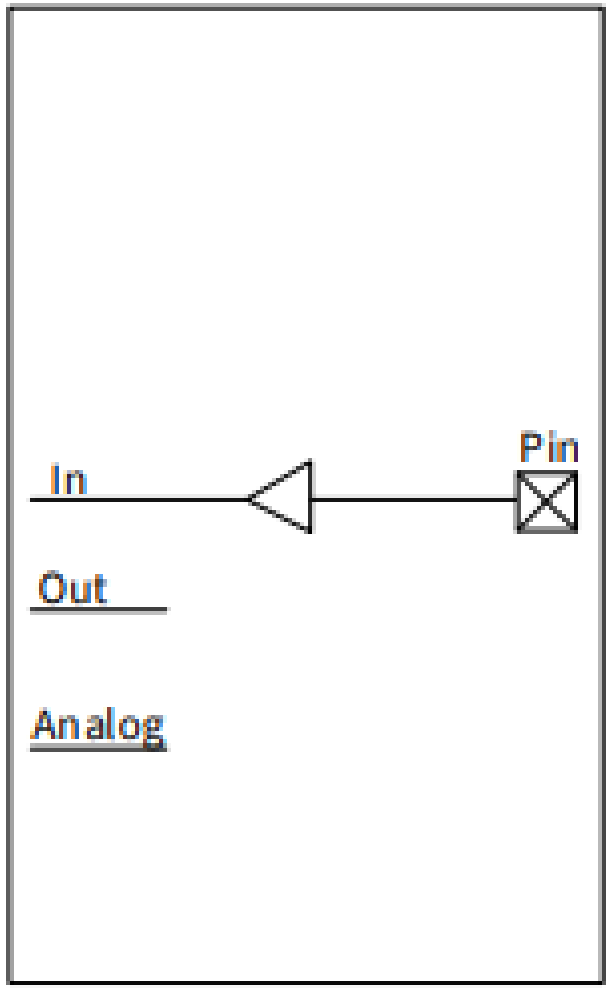


Варіанти конфігурації піна PSoC4





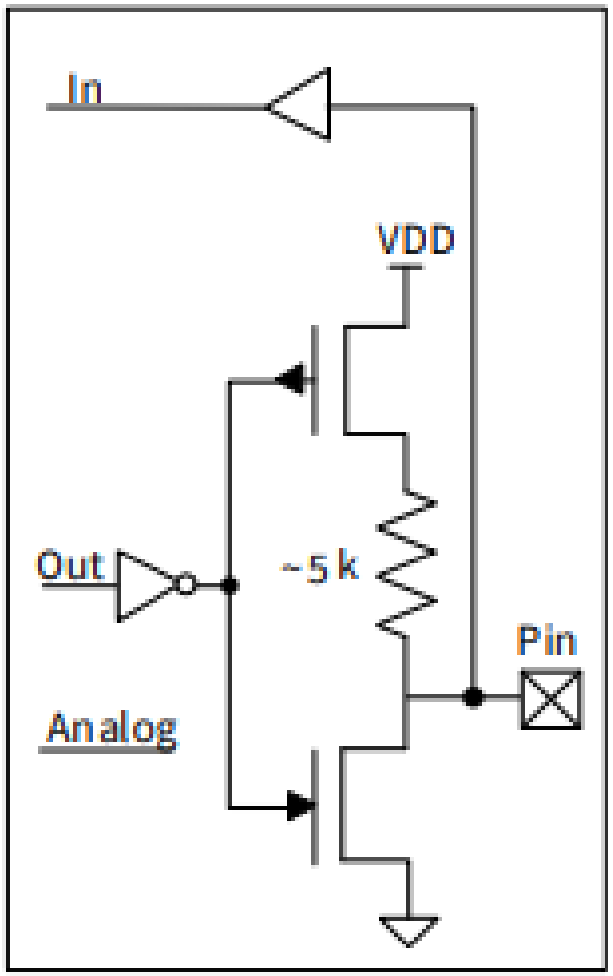
1 . High-Impedance Analog



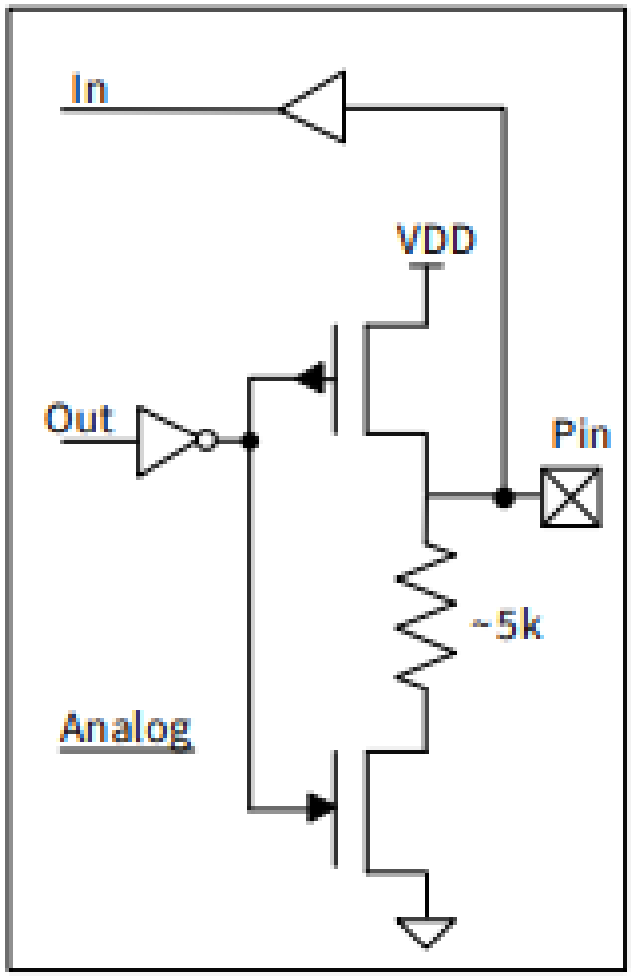
2 . High-Impedance Digital

High-impedance (High-Z) analog
Аналоговий вхід або вихід. Цифрові схеми не під'єднано.

High-impedance (High-Z) digital
Під'єднано тільки цифровий вхід. Потенціал на виводі задається винятково зовнішніми факторами.



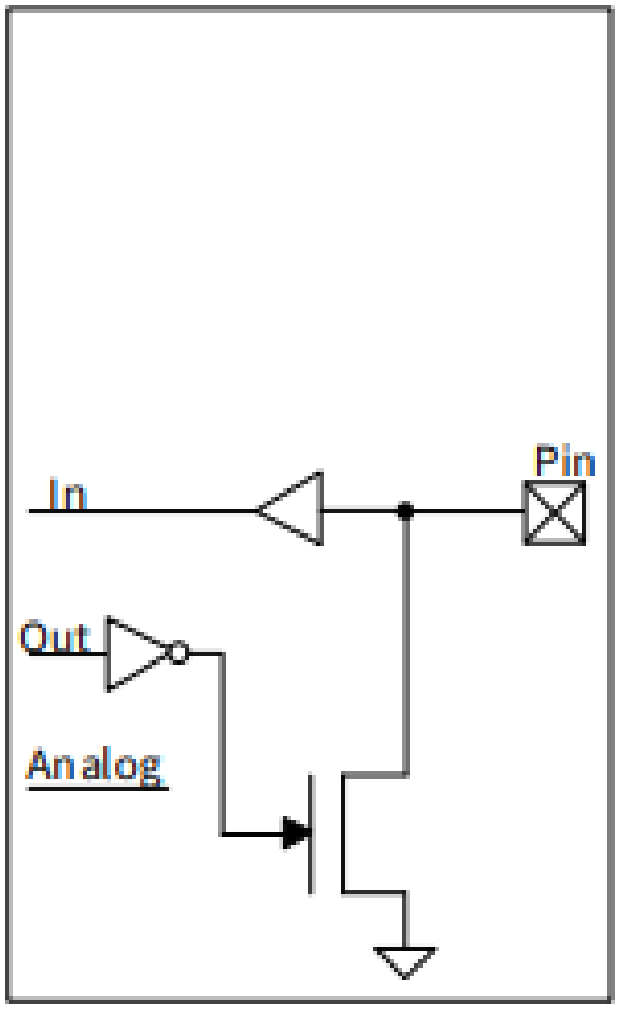
3 . Resistive Pull Up



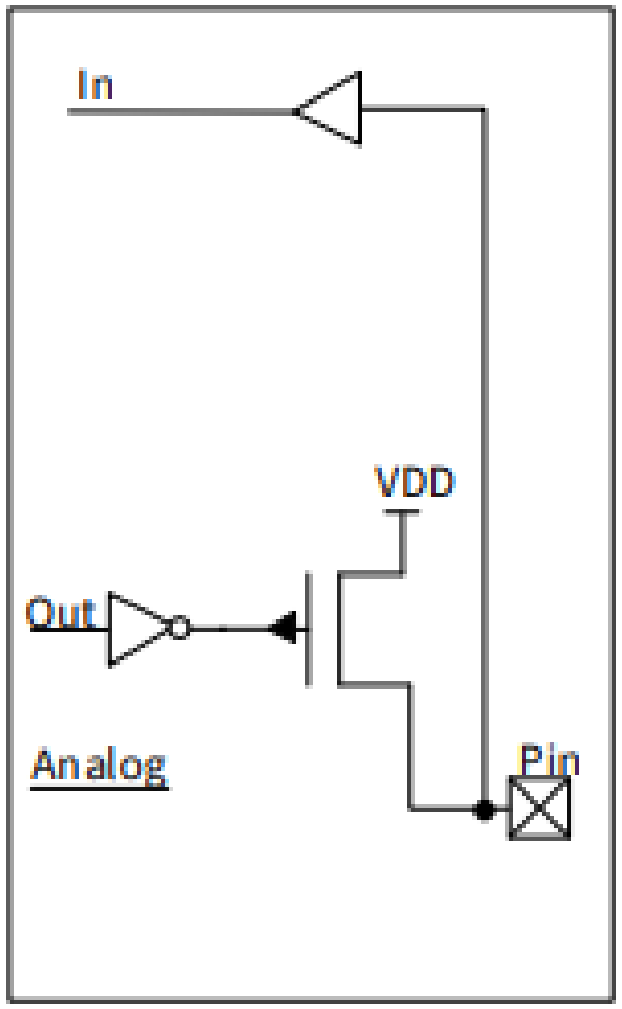
4 . Resistive Pull Down

Resistive pull up
Інтерфейс до входу з відкритим стоком, що має низький рівень, наприклад, вихід тахометра від двигунів або перемикач, підключений до землі. Його також можна використовувати для керування світлодіодами.

Resistive pull down
Інтерфейс до входу з відкритим стоком, що має високий рівень, або перемикач, підключений до VDD. Його можна використовувати як вихід для інтерфейсу світлодіодів у режимі струмового стоку.



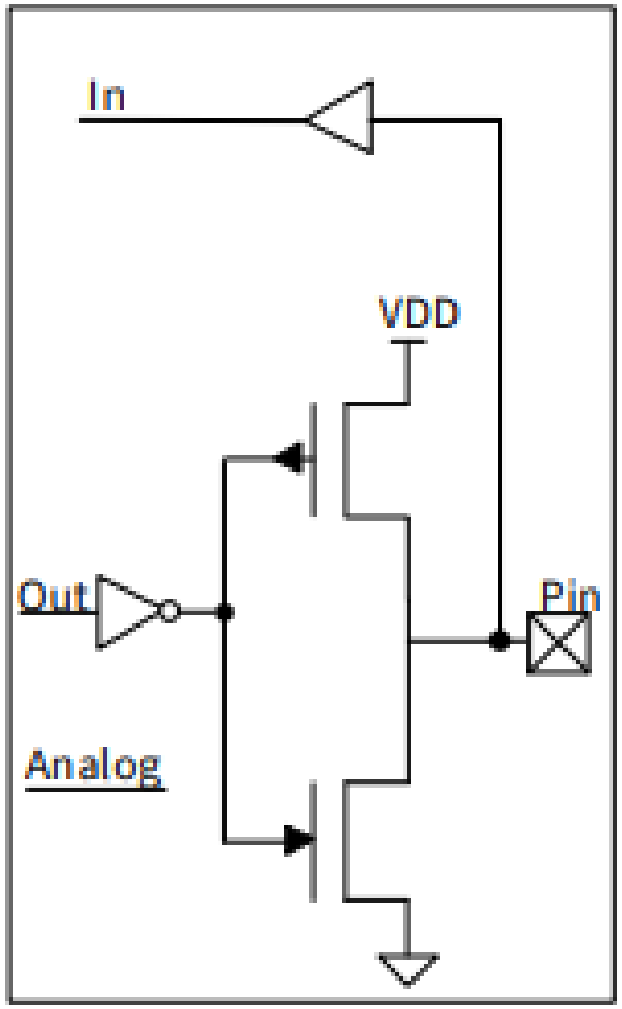
5. Open Drain, Drives Low



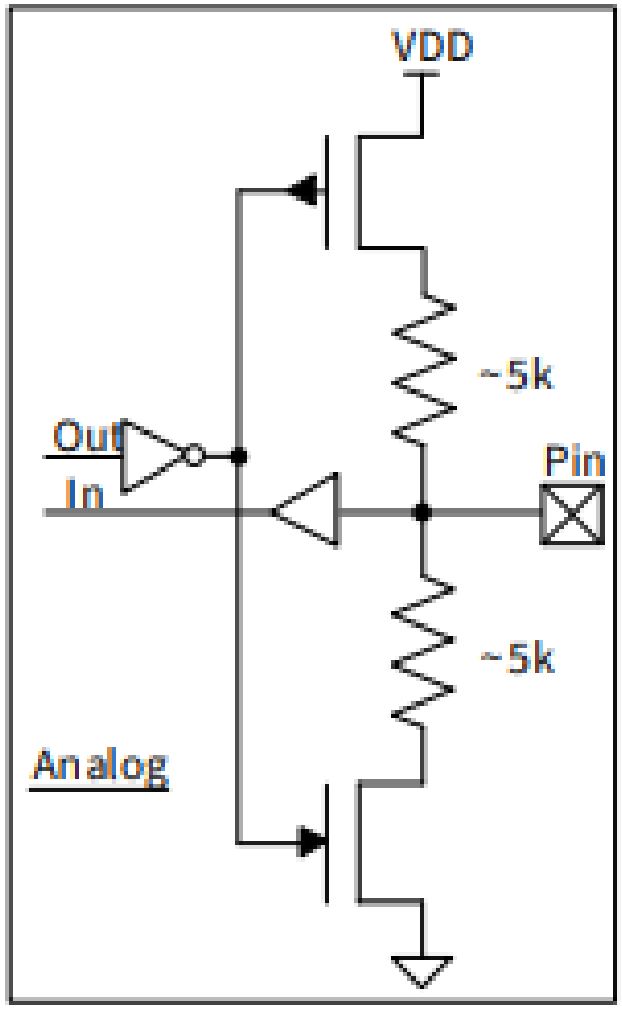
6. Open Drain, Drives High

Open drain, drives low
Цей режим забезпечує високий імпеданс у високому стані та строгий драйв до низького рівня; ця конфігурація використовується наприклад для I2C виводів. Цей режим працює у поєднанні з зовнішнім підтягувальним резистором

Open drain, drives high
Забезпечує строгий драйв у високому стані та високий імпеданс у низькому стані. Цей режим працює у поєднанні з зовнішнім підтягувальним резистором до низького рівня.



7. Strong Drive



8. Resistive Pull Up & Pull Down

Strong drive
Типовий режим коли чітко виводиться строгі високий та низький рівні.

Resistive pull up and pull down
Дуже нетиповий режим. В обох станах додано послідовний резистор, котрий робить обидва стани м'яко підтягнутими.

Режими роботи пінів конфігуруються 3-ма DM регістрами

www.infineon.com - AN32399

User Interface – LCD Driving Methods using PSoC®

AN32399

Author: Svyatoslav Paliy
Associated Project: Yes
Associated Part Family: All
Software Version: PSoC® Designer™ 5.0
Associated Application Notes: None

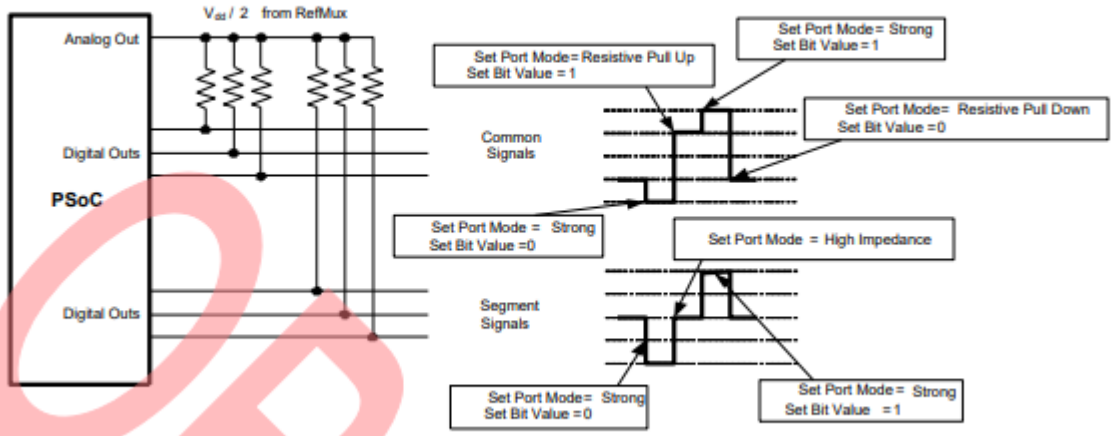
introduction to LCD construction, working principles, and driving methods. The attached
multiplexed LCD.

LCD Basics

An LCD panel is constructed of many layers. Figure 1
shows the typical construction of an LCD panel. The first
layer is called a Front Polarizer.

are widely used in
provide good graphic
rgy, and are easy to use.

Figure 19. 1/4 Bias Signal Generation



ATMega16U2 не має так багато режимів

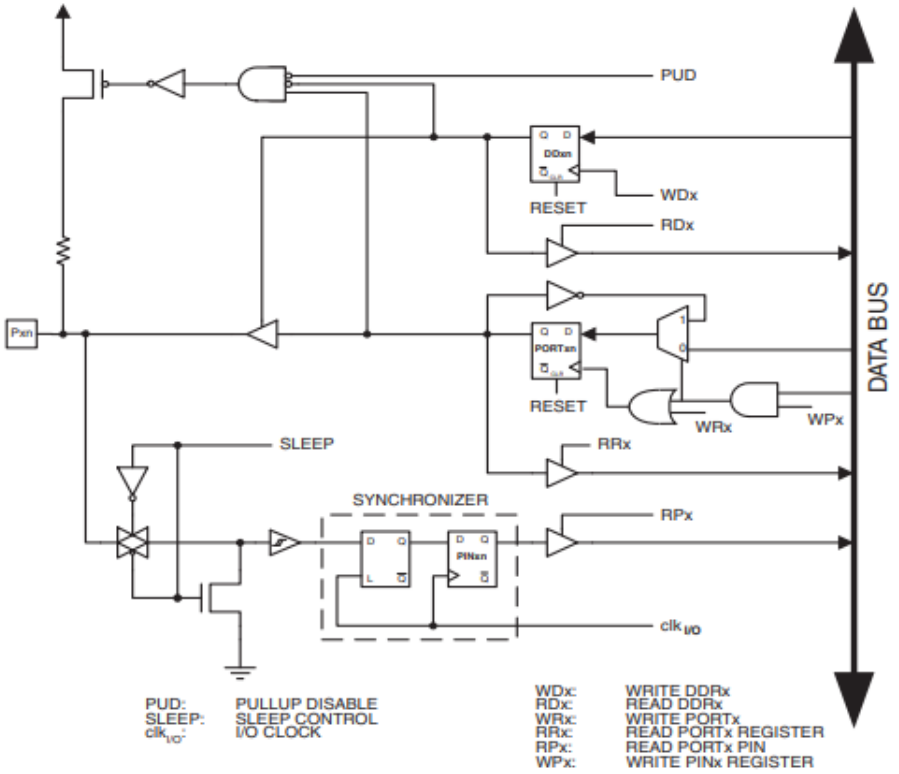
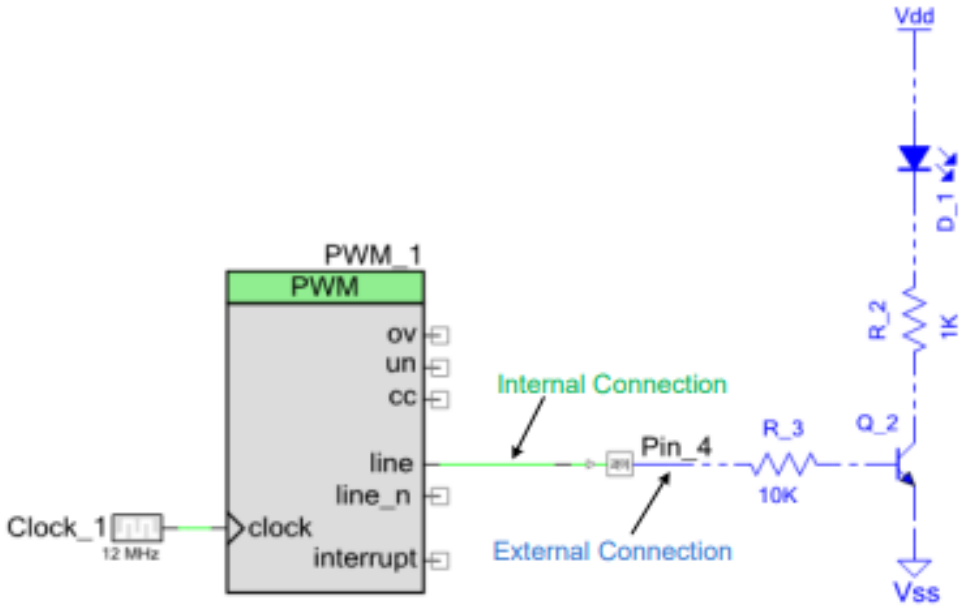


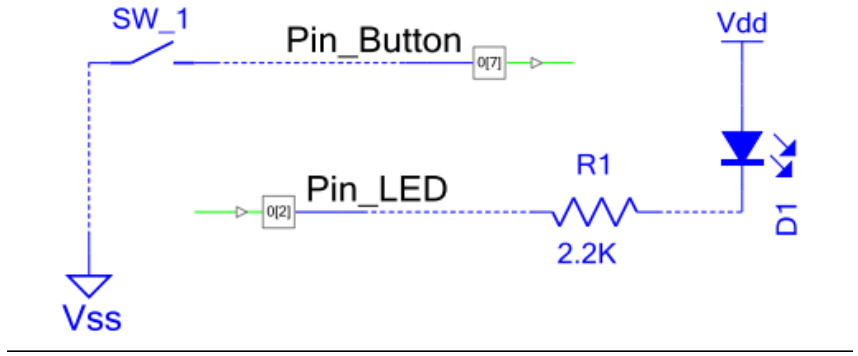
Table 12-1. Port Pin Configurations

DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

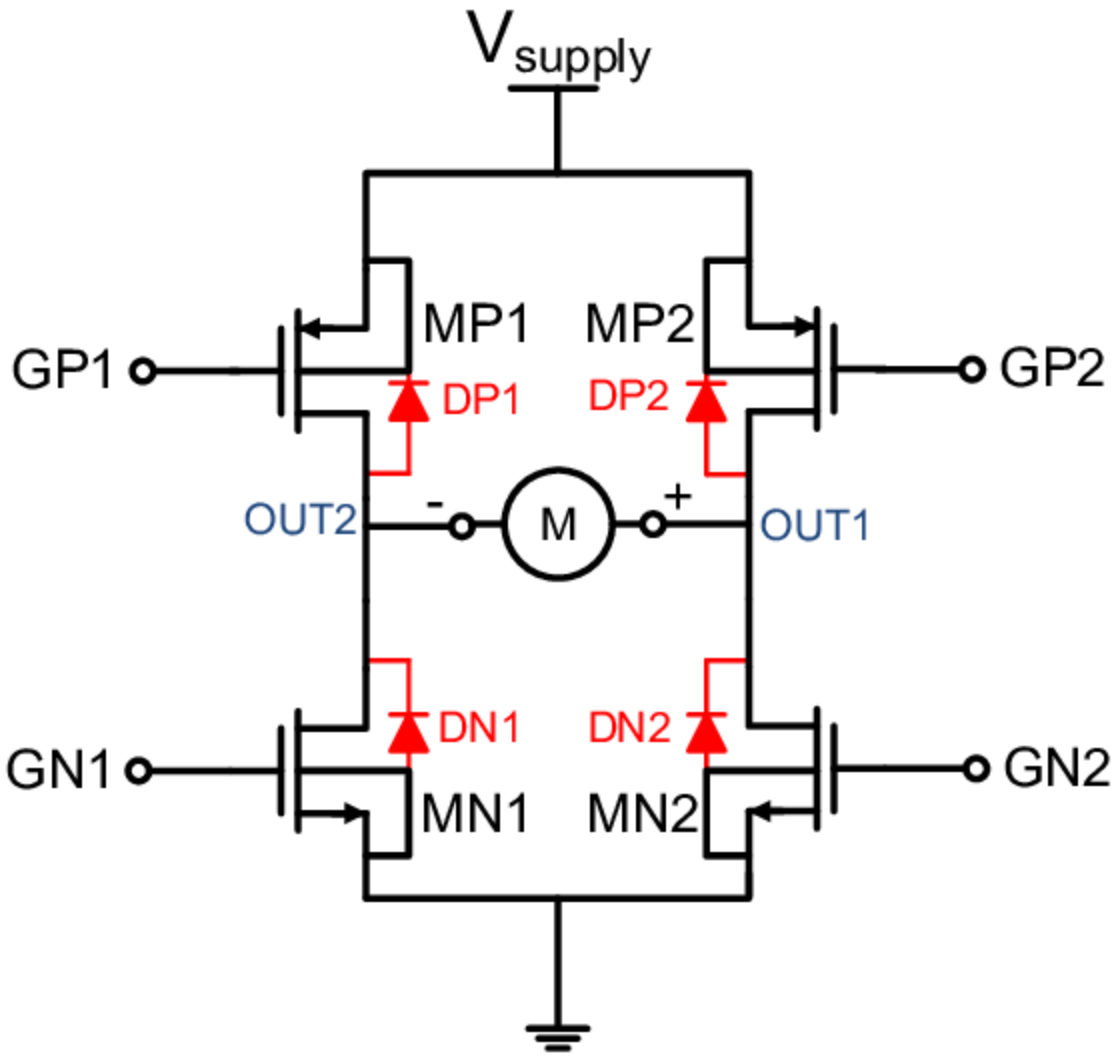
https://ww1.microchip.com/downloads/en/DeviceDoc/doc7799.pdf

Типове використання GPIO





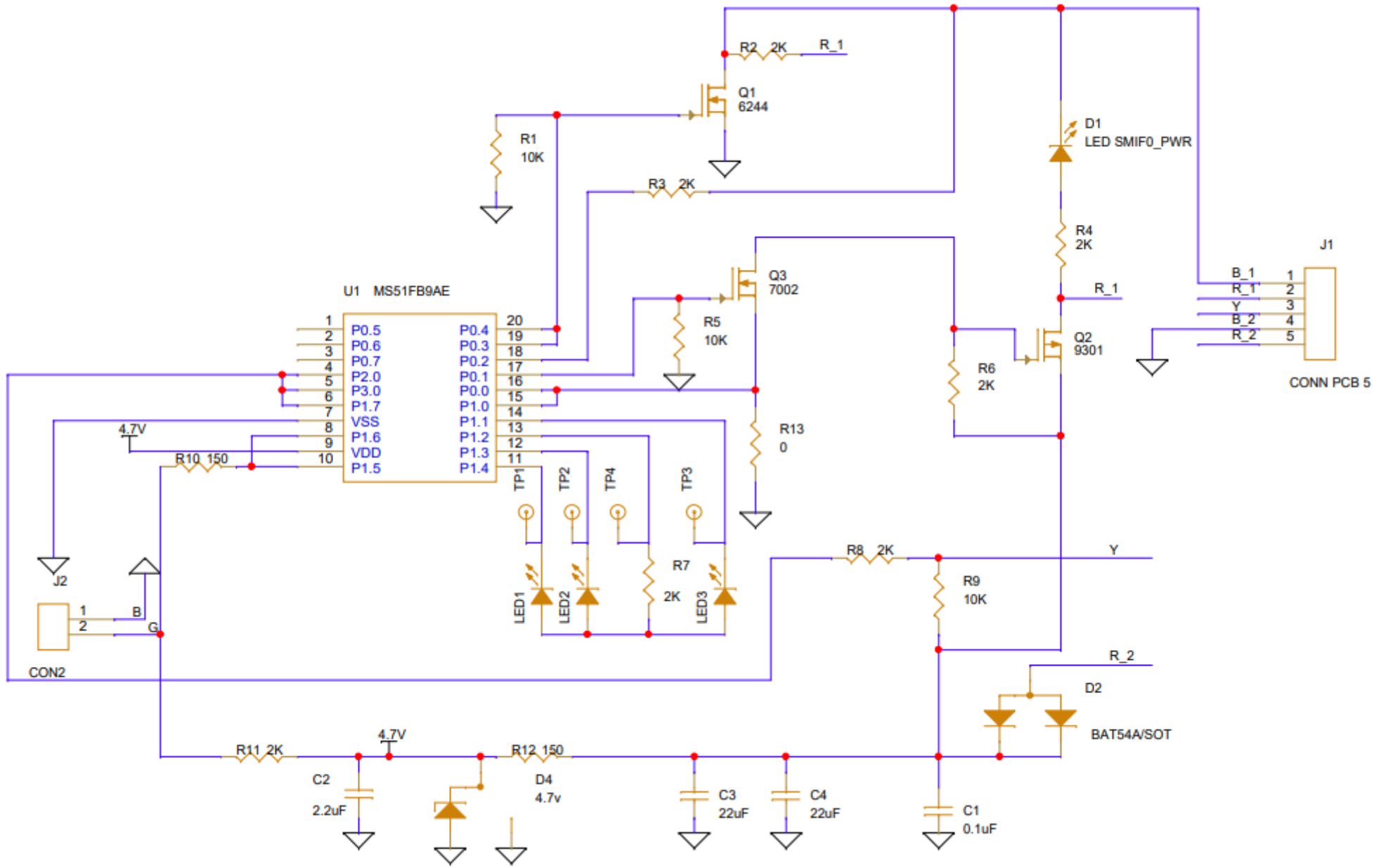
H-міст


V supply
MP1
MP2
GP1
GP2
DP1
DP2
OUT2
-
M
+
OUT1
DN1
DN2
GN1
GN2
MN1
MN2

Одночасність виводу в GPIO

```
// Встановлюємо значення на порти GPIO для PSoC
// Припустимо, що ми використовуємо порт 2 для виводу
CY_SET_REG8(CYREG_PRT2_DR, 0x0F);


// Щось подібне для Arduino
#include <avr/io>

PORTD = 0x0F;

// В наступному прикладі піни будуть перемикатися послідовно,
// це можна побачити за допомогою швидкого логічного аналізатора
digitalWrite(9,1);
digitalWrite(10,1);
digitalWrite(11,1);
digitalWrite(12,1);
```

Не забуваємо що процесор виконує команди одна за одною а час тим часом іде!


R2 2K
R_1
Q1
6244
R1
10K
R3 2K
D1
LED SMIF0_PWR
R4
2K
J1
B_1
R_1
Y
B_2
R_2
CONN PCB 5
U1 MS51FB9AE
P0.5
P0.6
P0.7
P2.0
P3.0
P1.7
VSS
P1.6
VDD
P1.5
P0.4
P0.3
P0.2
P0.1
P0.0
P1.0
P1.1
P1.2
P1.3
P1.4
Q3
7002
R5
10K
R_1
Q2
9301
R6
2K
R13
0
4.7V
R10 150
TP1
TP2
TP4
TP3
J2
CON2
B
G
LED1
LED2
LED3
R7
2K
R8 2K
Y
R9
10K
R_2
D2
BAT54A/SOT
R11 2K
4.7V
R12 150
C2
2.2uF
D4
4.7v
C3
22uF
C4
22uF
C1
0.1uF

Щиро дякую!