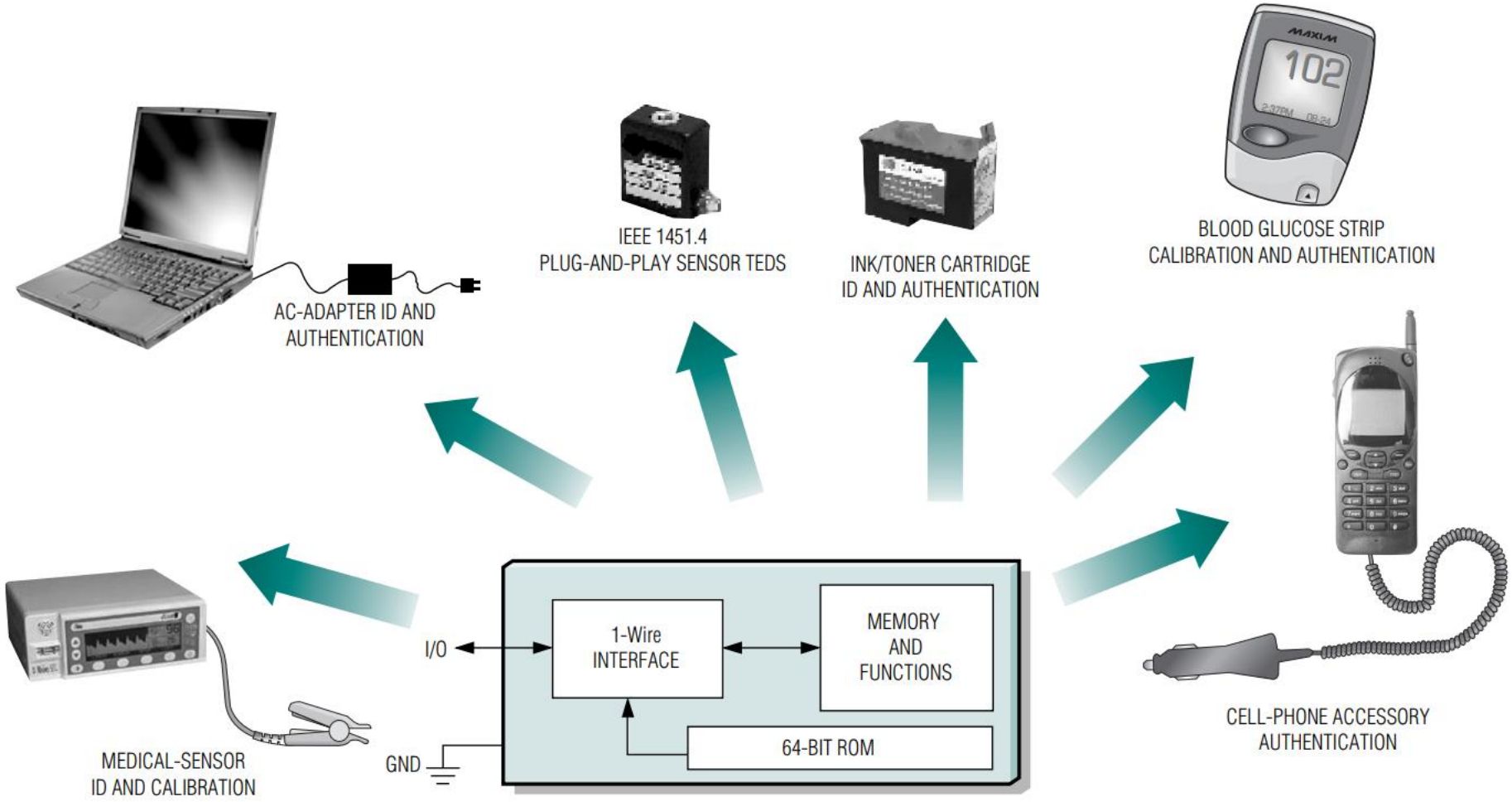
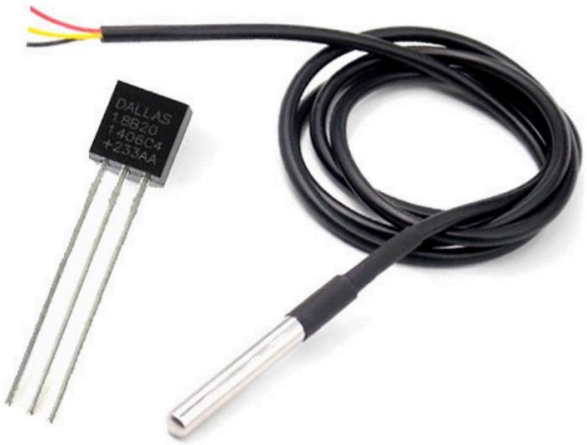
Вбудовані системи

Шина 1-Wire

Приклади використання пристроїв з інтерфейсом 1-Wire



iButton ключ (DS1990)



Термометр DS18B20

Шина 1-Wire

Шина **1-Wire** є основою мереж ***MicroLAN*** і розроблена наприкінці 90-х років фірмою **Dallas Semiconductor**. В даний час фірма **Dallas Semiconductor** є дочірнім підприємством фірми **Maxim Integrated**. Особливістю цієї шини є те, що для живлення підключених пристроїв, і для обміну даними використовується одна пара проводів.

Основні характеристики шини:

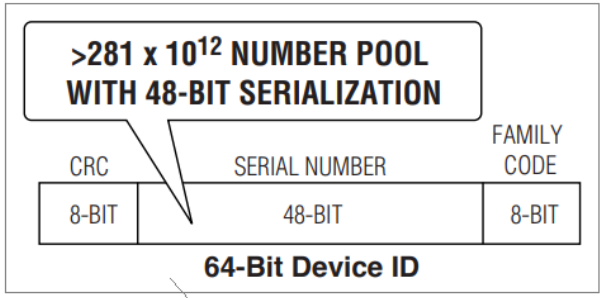
- максимальна протяжність шини до 300 м;
- швидкість передачі інформації 16,3 Кбіт/с;
- максимальна кількість адресованих елементів на шині 256;
- рівні напруги на шині відповідають стандартним рівням КМОП/ТТЛ;
- напруга живлення компонентів мережі 2,8...6 В;
- для з'єднання елементів мережі може застосовуватися звичайний телефонний кабель або вита пара.

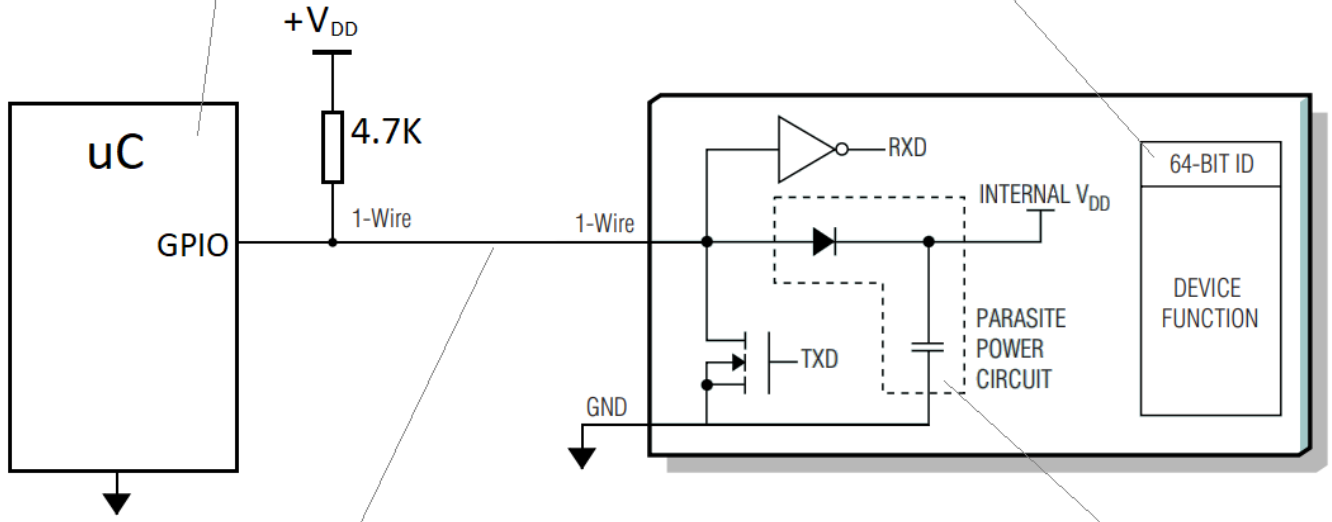
Існують і модифікації шини **1-Wire** які підтримують швидкісний режим роботи шини (Overdrive, 142 Кбіт/с). Проте такі мікросхеми можуть працювати лише на шині малої протяжності і за умови, коли рівень зовнішніх електричних завад зведений до мінімуму. Шина ***MicroLAN*** побудована за технологією Master/Slave. На шині має бути хоча б один ведучий пристрій (Master), а всі мають бути веденими (Slave). Ведучий пристрій ініціює всі процеси передачі інформації в межах шини. Master може прочитати дані з будь-якого Slave-пристрою або записати їх туди. Передача інформації від одного Slave до іншого безпосередньо неможлива. Для того, щоб Master міг звертатися до будь-якого з ведених пристроїв по шині, кожен ведений пристрій містить в собі індивідуальний код (IDкод). Цей код заноситься в спеціальну область мікросхеми за допомогою лазера.

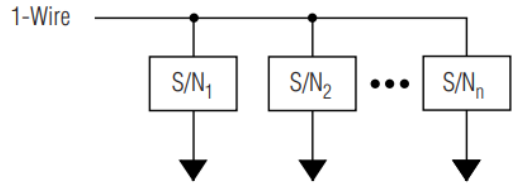
Інтерфейс 1-Wire

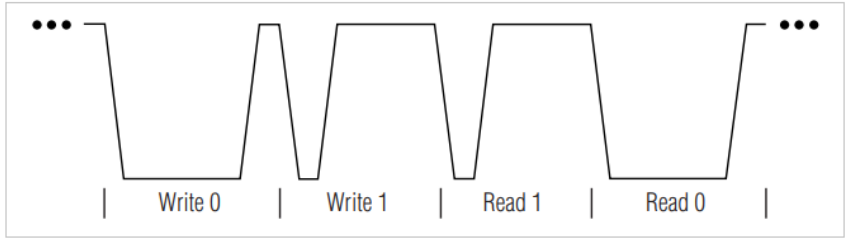
* Проста реалізація ведучого з використанням одного порта вводу-виводу

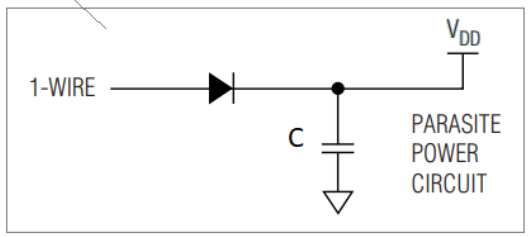
* Унікальний 64-бітний ідентифікатор для кожного пристрою







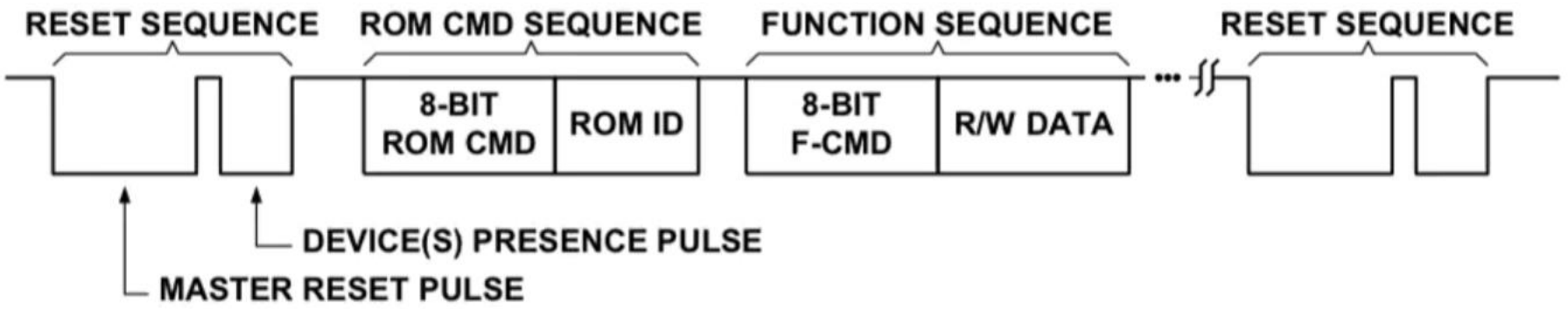


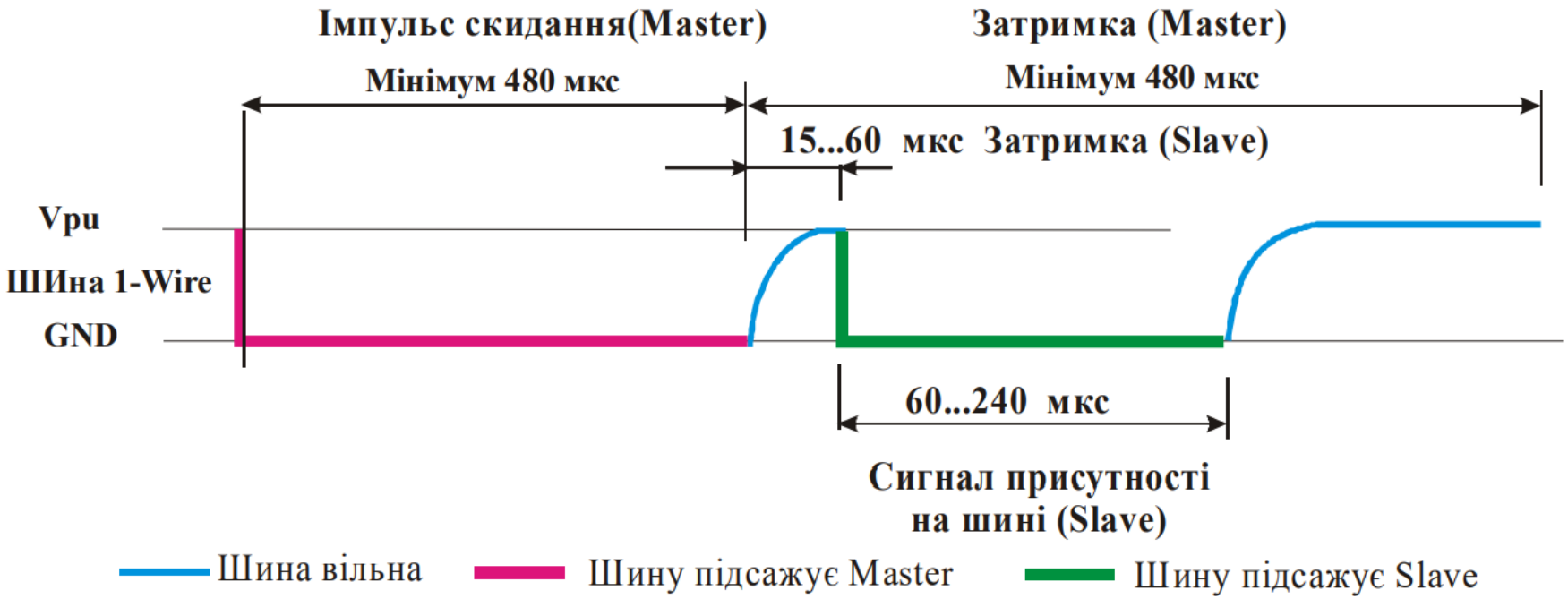


* Двохнаправлена послідовна передача даних по одній лінії
* Один ведучий, багато ведених

* Паразитне живлення від лінії даних

Передача і прийом даних по шині 1-Wire

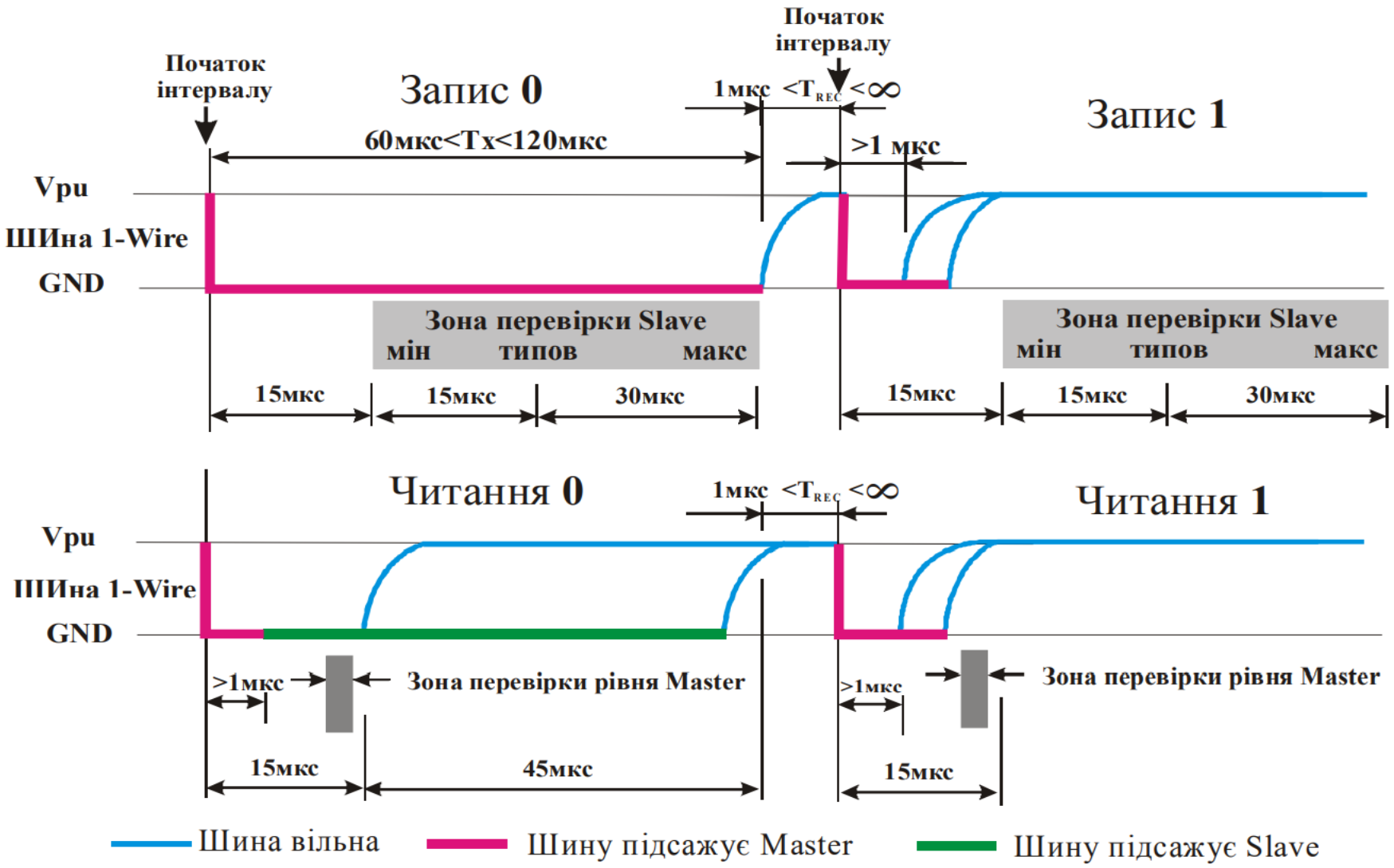




Часова діаграма процесу скидання (Reset) та зчитування сигналу присутності пристроїв на шині.

Передача і прийом даних по шині 1-Wire

Часові характеристики протоколу 1-Wire



Алгоритм взаємодії з пристроєм 1-Wire

1. Ведучий посилає на лінію сигнал reset "Скидання". Після чого лінія «звільняється» для отримання сигналу присутності. Якщо на шині присутній ведений, то протягом 60 мкс він повідомляє про "присутність". Якщо ведучий не отримує сигналу - "присутності", то він вважає, що підключених до шини пристроїв немає.

2. Далі ведучий повинен визначити, до якого з пристроїв на шині даних він буде далі звертатися. Даний вибір забезпечується надсиланням однієї з ROM-команд (довжиною в 1 байт), які працюють з унікальними кодами пристроїв (мережевий рівень протоколу):

Search ROM (0xF0) - "пошук ROM". Якщо ID коди підключених пристроїв не відомі, то ця команда дозволяє ведучому їх визначити.
Read ROM (0x33) - "читання ROM" - команда використовується тоді, коли до шини підключений тільки один ведений пристрій. При отриманні даної команди все ведені пристрої на шині відсилають свій унікальний ID код.
Skip ROM (0xCC) - "пропуск ROM". команда використовується коли необхідно відправити функціональну команду всім пристроям на шині.
Match ROM (0x55) - "збіг ROM". Використовується для вибору конкретного підлеглого пристрою на шині. Після відправки команди ведучий передає 64-розрядний ID код (ROM). Якщо на шині присутній ведений пристрій з даним ID, то він залишиться активним на шині, а всі інші переходять в неактивний стан до наступного імпульсу скидання.

3. Ведучий відправляє функціональну команду - це вже транспортний рівень протоколу; при цьому набір функціональних команд і подальша поведінка залежить від конкретного пристрою 1-Wire.

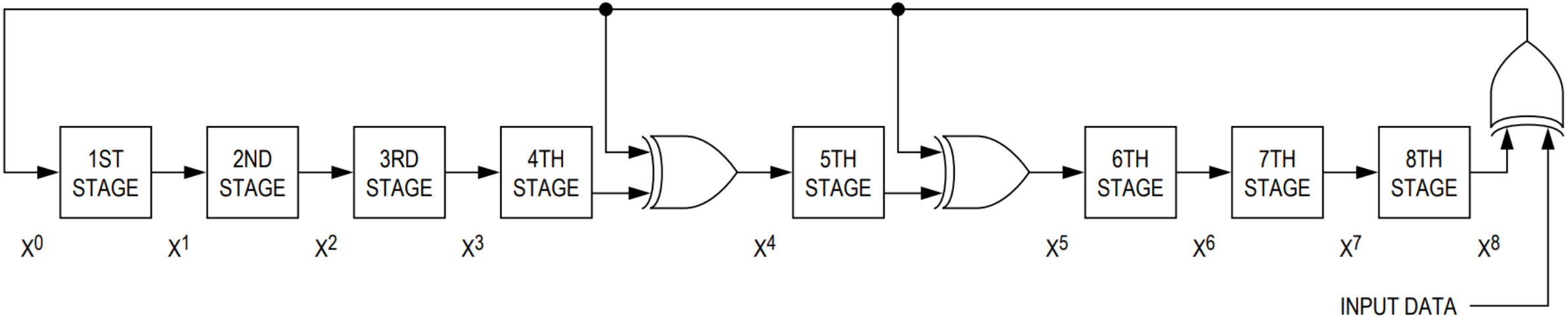
Програмна реалізація протоколу 1-Wire

```c
#include <avr/io.h>
#include <util/delay.h>

#define OW_PIN PB0
#define OW_IN PINB
#define OW_OUT PORTB
#define OW_DDR DDRB

uint8_t OW_Reset(void)
{
  uint8_t err;

    OW_OUT &= ~(1 << OW_PIN);
    OW_DDR |= 1 << OW_PIN;
    _delay_us(480);
    cli();
    OW_DDR &= ~(1 << OW_PIN);
    _delay_us(70);
    err = OW_IN & (1 << OW_PIN); // presence detect
    sei();
    _delay_us(410);
    if ((OW_IN & (1 << OW_PIN)) == 0) {
        err = 1;
    }
  return err;
}
```

```c
uint8_t OW_IO_Bit(uint8_t b)
{
    cli();
    OW_DDR |= 1 << OW_PIN;
    _delay_us(6);  // 1
    if (b) {
        OW_DDR &= ~(1 << OW_PIN);
    }
    _delay_us(9);  // 14
    if ((OW_IN & (1 << OW_PIN)) == 0) {
        b = 0;
    }
    _delay_us(55);
    OW_DDR &= ~(1 << OW_PIN);
    sei();
  return b;
}

uint8_t OW_RW_byte(uint8_t b)
{
  uint8_t i, t;
    for (i =0; i < 8; i++)
    {
        t = OW_IO_Bit(b & 1);
        b >>= 1;
        if (t) b |= 0x80;
    }
  return b;
}
```

Алгоритм розрахунку контрольної суми CRC

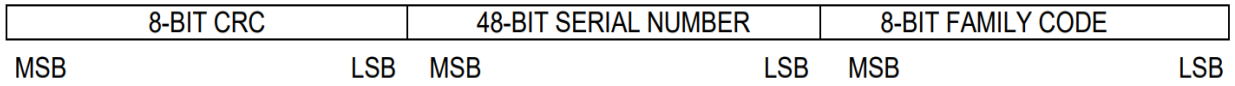
$$CRC = X^8 + X^5 + X^4 + 1$$




```
uint8_t OW_UpdateCRC(uint8_t crc, uint8_t b)
{
    for (uint8_t p = 0; p < 8; p++)
    {
        crc = ((crc ^ b) & 1)? (crc >> 1) ^ 0b10001100: (crc >> 1);
        b >>= 1;
    }
  return crc;
}
```

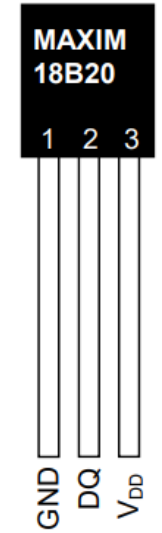
Приклад програми (Читання ID)

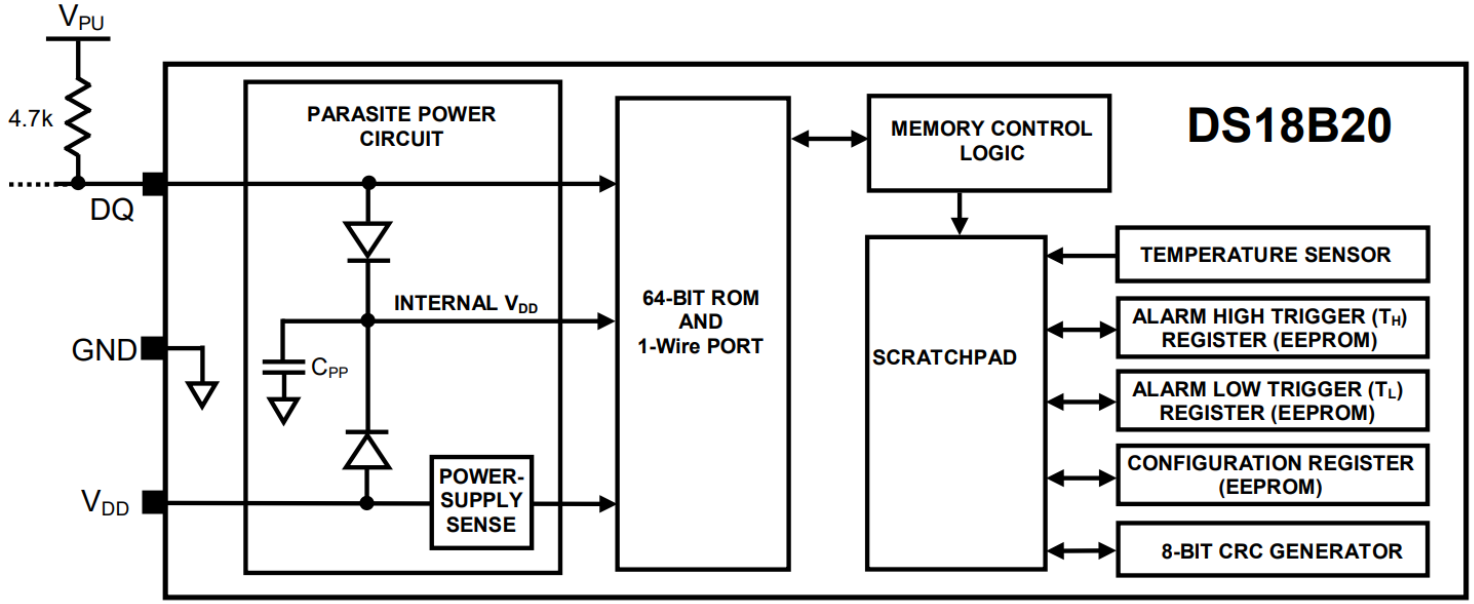
```
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#define OW_PIN PB0
#define OW_IN PINB
#define OW_OUT PORTB
#define OW_DDR DDRB

uint8_t OW_Reset(void); //…
uint8_t OW_IO_Bit(uint8_t b); //…
uint8_t OW_RW_byte(uint8_t b); //…
uint8_t OW_UpdateCRC(uint8_t crc, uint8_t b); // …

uint8_t OW_Get_ROM(uint8_t *id)
{
  uint8_t i, crc = 0;

    if (OW_Reset()) {
      return 0xFF;      // error, device not found
    }
    OW_RW_byte(0x33);  //command get ROM
    for (i = 0; i < 7; i++)
    {
      id[i] = OW_RW_byte(0xFF);
      crc = OW_UpdateCRC(crc, id[i]);
    }
    id[7] = OW_RW_byte(0xFF);
    if (id[7] != crc) return 0xFE; // error CRC
  return 0;
}
```



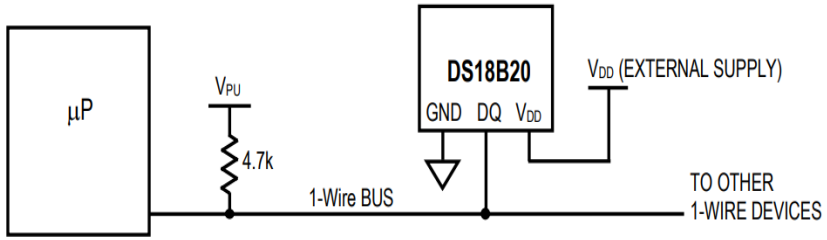

```
// продовження ...

int main(void)
{
  uint8_t id[8];

  if ( OW_Get_ROM(buff) == 0 ) {
   // printf("ROM: %02X%02X%02X%02X%02X%02X%02X%02X\n",
   //   id[7],id[6],id[5],id[4],id[3],id[2],id[1],id[0]);
  } else {
   //   printf("Error!\n");
  }

  // mail loop =============
  for (;;)
  {
    _delay_ms(10);
  }

  return 0;
}
```

DS18B20 (Термометр)





Структурна схема DS18B20



	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LSB	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}

	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MSB	S	S	S	S	S	2^6	2^5	2^4

S = SIGN

Організація пам’яті DS18B20

SCRATCHPAD (POWER-UP STATE)

			EEPROM
Byte 0	Temperature LSB (50h)	(85°C)	
Byte 1	Temperature MSB (05h)		
Byte 2	T_H Register or User Byte 1*	↔	T_H Register or User Byte 1
Byte 3	T_L Register or User Byte 2*	↔	T_L Register or User Byte 2
Byte 4	Configuration Register*	↔	Configuration Register
Byte 5	Reserved (FFh)		
Byte 6	Reserved		
Byte 7	Reserved (10h)		
Byte 8	CRC*		

**Power-up state depends on value(s) stored in EEPROM.*

Опис алгоритму роботи з термометром DS18B20

1. Ведучий відсилає сигнал скидання.
2. Датчик відповідає сигналом «присутності».
3. Ведучий відсилає адресну команду "ROM" (якщо датчик на лінії один, то відправляємо команду «пропуск ROM» **0xCC**)
4. Ведучий відсилає функціональну команду "Convert T **0x44**" - по цій команді датчик температури почне одноразове вимірювання температури; результат перетворення буде записаний в пам'ять датчика «Scratchpad».
5. Ведучий чекає поки датчик завершить перетворення (приблизно 1 секунду).
6. Ведучий відсилає сигнал скидання.
7. Датчик відповідає сигналом «присутності».
8. Ведучий відсилає адресну команду "ROM" після якої слідує функціональна команда "Read Scratchpad" **0xBE,** після цієї команди датчик відсилає 9-байт даних.
9. Ведучий зчитує «Scratchpad» та переходить до обробки результату.

```
uint8_t t_dt, dt;    //Temperature format:  raw_t -> t.dt C
int8_t t;
uint16_t raw_t = readRawTemp();
   if ( raw_t & 0x8000) {
       raw_t = ~raw_t + 1; t_dt = raw_t & 0x0F; t = -(raw_t >> 4);
   } else {
       t_dt = raw_t & 0x0F; t = raw_t >> 4;
   }
   dt = ((t_dt << 1) + (t_dt << 3)) >> 4;
```