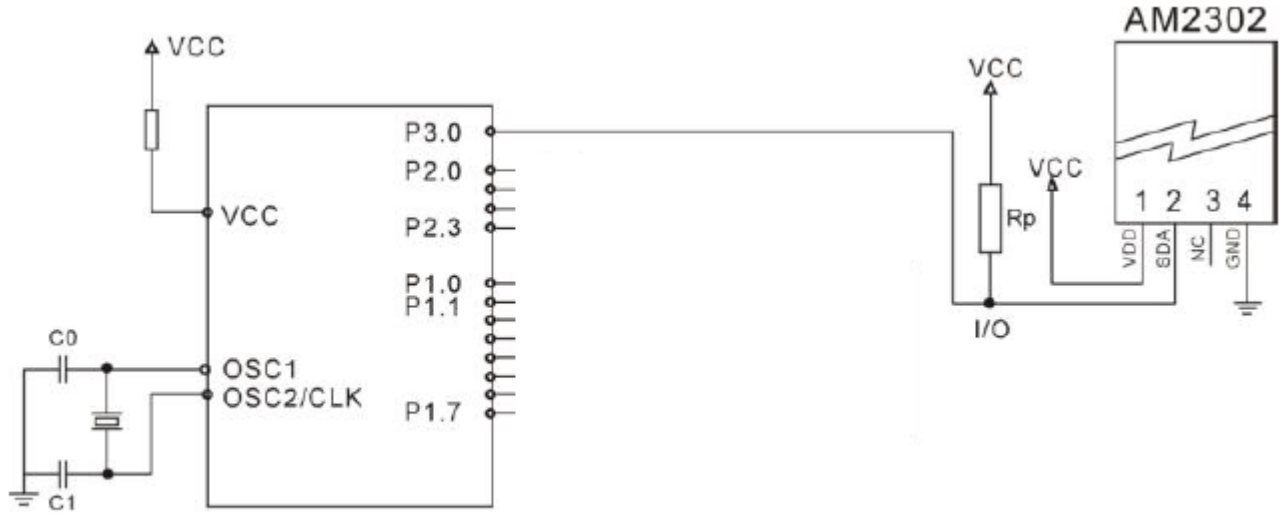
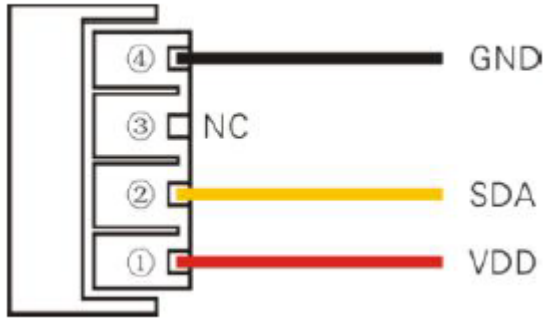
AM2302 (DHT22) – czujnik temperatury i wilgotności na magistrali 1-wire

Procesory nie posiadają interfejsów do magistrali 1-wire – urządzenie Slave można podłączyć na dowolnej linii GPIO. Protokół komunikacyjny jest w całości softwarowy.

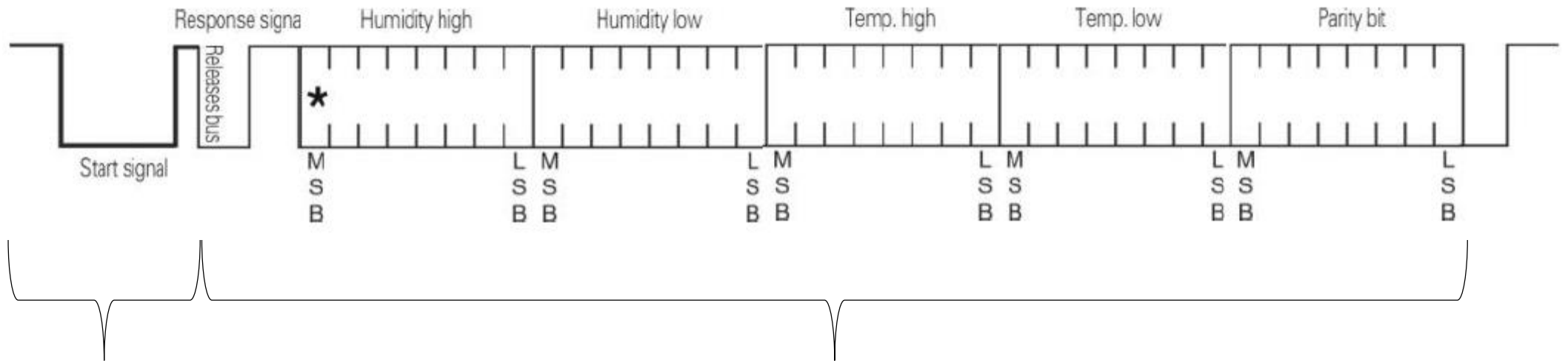




Pin	Name	Description
①	VDD	Power (3.3V–5.5V)
②	SDA	Serial data, bidirectional port
③	NC	Empty
④	GND	Ground

AM2302 (DHT22) – protokół komunikacyjny

Czujnik temperatury i wilgotności DHT22 nie posiada pamięci ROM – zatem na wybranej linii można podłączyć tylko jeden taki czujnik działający w układzie Master-Slave z procesorem Master. W stanie bezczynnym magistrala jest podciągnięta rezystorem pod zasilanie. Układ DHT22 nie przyjmuje żadnych rozkazów oprócz sprawdzenia obecności (Response signal). Pomiary temperatury i wilgotności wykonuje automatycznie i w każdej chwili, po odebraniu impulsu resetu (Start signal) automatycznie przesyła wyniki pomiarów po magistrali 1-wire.

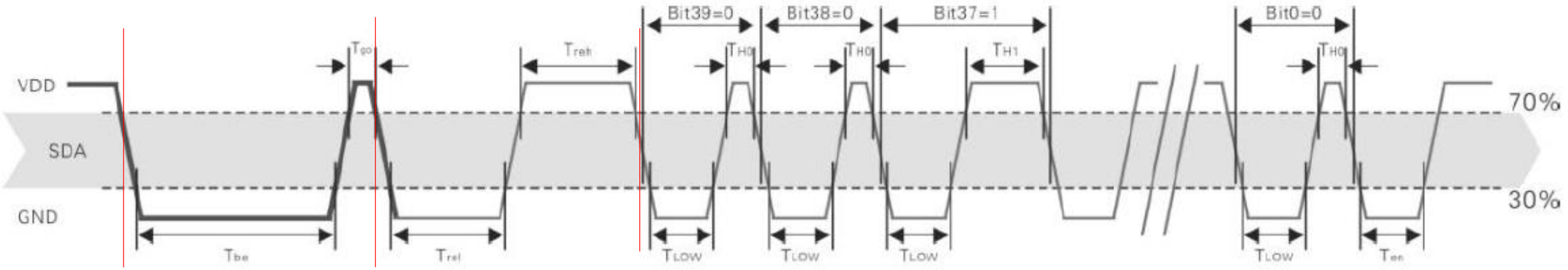


Master Write: impuls resetu

Master Read: impuls obecności + 40 bits (5 bytes)

Master rozpoczyna transmisję danych nadając impuls resetu. Układ Slave DHT22 odpowiada impulsem obecności, po czym nadaje kolejnych 40 bitów (5 bajtów) z danymi: dwa bajty wilgotności, dwa bajty temperatury i jeden bajt sumy kontrolnej CRC.
Na bajtach humidity high i humidity low zapisana jest 10-cio krotność wilgotności w procentach
Wilgotność [%] = (humidity high<<8 | humidity low) /10. Rozdzielczość wynosi 1/10 %.
Na bajtach temp high i temp low zapisana jest 10-cio krotność temperatury w stopniach °C
temperatura [°C] = (temp high<<8 | temp low) /10. Rozdzielczość wynosi 1/10 °C.

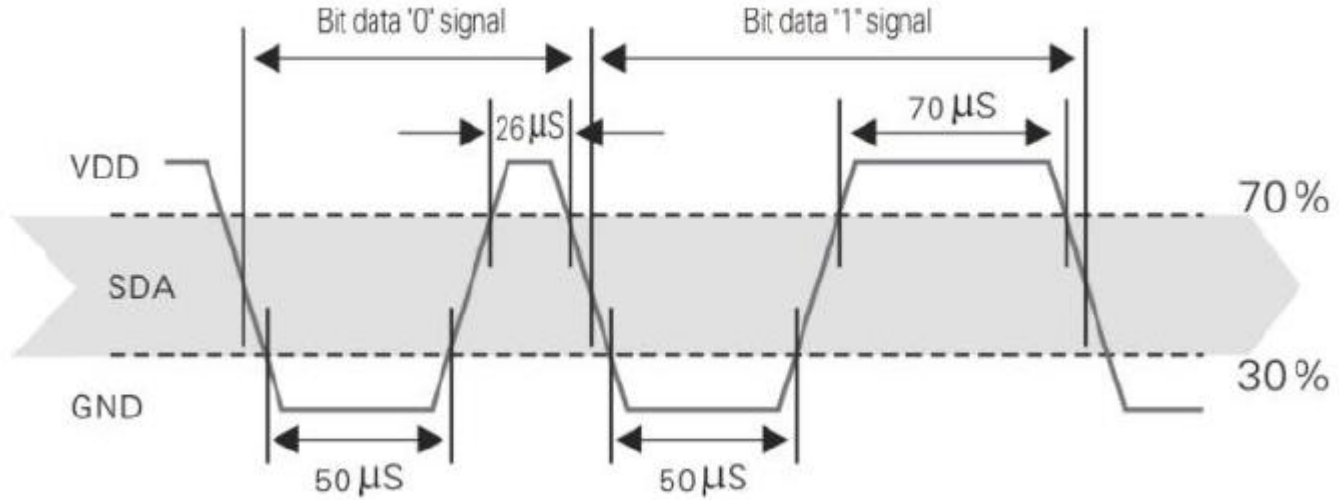
AM2302 (DHT22) – czasy trwania impulsów



start signal response signal

Symbol	Parameter	min	typ	max	Unit
T_{be}	Host the start signal down time	0.8	1	20	mS
T_{go}	Bus master has released time	20	30	200	μS
T_{rel}	Response to low time	75	80	85	μS
T_{reh}	In response to high time	75	80	85	μS
T_{LOW}	Signal "0", "1" low time	48	50	55	μS
T_{H0}	Signal "0" high time	22	26	30	μS
T_{H1}	Signal "1" high time	68	70	75	μS
T_{en}	Sensor to release the bus time	45	50	55	μS

AM2302 (DHT22) – impulsy logicznego 0 i 1



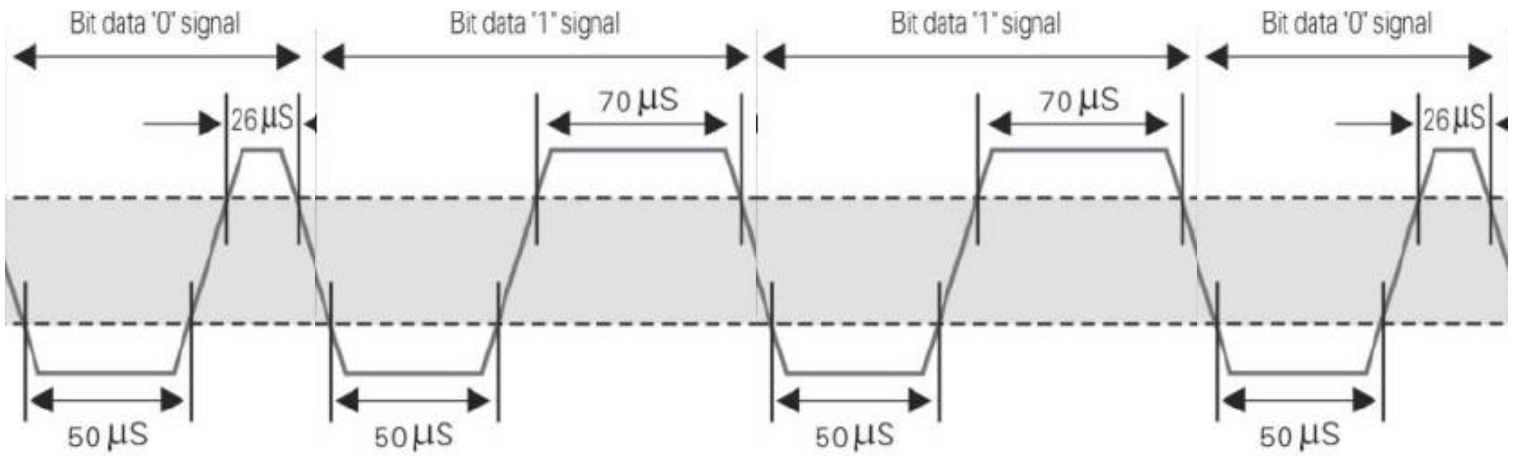

```
F_CPU            // cykli w sekundzie
F_CPU/1000       //cykli w ms
//timeout po 1ms
uint32_t cycles_in_ms = F_CPU / 1000;

Przykład: taktowanie z kwarcu 16MHz
F_CPU=16000000 // cykli w sekundzie
16000                //cykli w milisekundzie
Timeout  po 16000 cyklach

#define pinn 5
#define PINmy PINB

//procedury pochodzą z biblioteki
//DHT library
//MIT license written by Adafruit Industries
```

```
uint32_t expectLow(void)
{
uint32_t count = 0;
while ((PINmy & (1<<pinn)) == 0)
        {
        if (count++ >= cycles_in_ms)
        return 0;
        }
return count;
}
```

```
uint32_t expectHigh(void)
{
uint32_t count = 0;
while ((PINmy & (1<<pinn)) == (1<<pinn))
        {
        if (count++ >= cycles_in_ms)
        return 0;
        }
return count;
}
```

Pętla *while ((PINmy & (1<<pinn)) == 0) {count++;}* oraz *while ((PINmy & (1<<pinn)) == (1<<pinn)) {count++;}* zlicza obroty pętli, które odpowiadają odcinkom czasu po 1/16 µs (przy taktowaniu 16MHz), ponieważ każdy obrót zachodzi w jednym cyklu zegarowym.
Jeśli oczekiwany stan napięciowy (low/high) nie pojawił się wcale to funkcja expectLow (/expectHigh) zwraca zero (na zmiennej count).
Jeśli oczekiwany stan napięciowy pojawił się ale trwał dłużej niż 16000 cykli zegarowych (co odpowiada czasowi 1ms), to zwracane jest 0, w przeciwnym wypadku zwracana jest czas trwania stanu jako krotność 1/16 µs (zapisany na zmiennej count).

AM2302 (DHT22) – fragment kodu programu



Każdy bit logiczny składa się z dwóch stanów napięciowych.
Na bit logicznego zera wypada 50 µs stanu niskiego i 26 µs stanu wysokiego.
Na bit logicznej jedynki wypada 50 µs stanu niskiego i 70 µs stanu wysokiego.
Bity logicznego zera i jedynki nie są tej samej długości, co uniemożliwia wykorzystanie funkcji opóźnienia _delay w procesie odczytu.

```
uint32_t stany [80];
 for (int i = 0; i < 80; i += 2)
 {
  stany[i]   = expectLow();
  stany[i + 1] = expectHigh();
 }

for (int i = 0; i < 40; i++)
 {
  uint32_t lowCycles  = stany[2 * i];
  uint32_t highCycles = stany[2 * i + 1];
  if ((lowCycles == 0) || (highCycles == 0))
  {
   // błędny odczyt
  }
  //bity nadawane od MSB do LSB
  data[i / 8] <<= 1;
  if (highCycles > lowCycles)
  {
   data[i / 8] |= 1;
  }
 }
```

Dwa stany napięciowe low/high przypadają na jeden bit logiczny. 80 odebranych stanów napięciowych low/high odpowiada 40-stu odebranym bitom.
Zmienna lowCycles (highCycles) przechowuje długość stanu napięciowego low (high) zliczoną w cyklach zegarowych czyli jako krotność 1/16 µs.
Wnioskowanie jest proste: jeśli highCycles > lowCycles to bit jest logiczną jedynką, w przeciwnym razie bit jest logicznym zerem.