Микропроцесорна системотехника
Лабораторно упражнение №6
"Изследване на сериен интерфейс I²C на микроконтролера MSP430FR6989"

6.1. Въведение

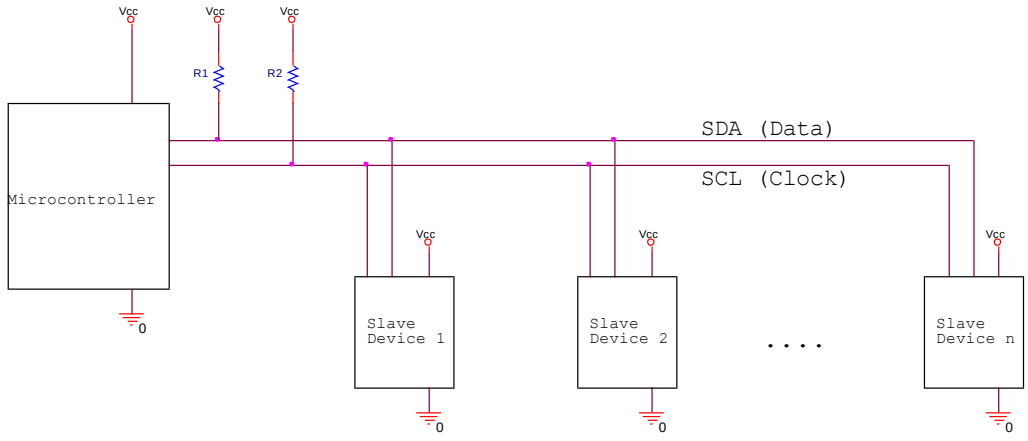
I²C е **синхронен сериен** интерфейс, който използва **два** (плюс още един маса) **проводника** за предаване на информация между два и повече чипа. Името му идва от IIC – Inter-Integrated Circuit и е създаден от Philips през 1982 г. От 2006 година използването на протокола за този интерфейс е безплатно. Платено е единствено запазването на уникален адрес за чип от даден вид, използващ I²C (това засяга само фирмите-производителки на чипа).

Връзката между два чипа се реализира на принципа master-slave, т.е. има една **главна** интегрална схема (master) и една **подчинена** (slave) такава. Обикновено master чипа е микроконтролерът в системата, а slave е периферен чип, с който ще се осъществява връзката. Съществува и режим multi-master, при който към I²C интерфейса са свързани два или повече главни (master) чипа.

На **фиг. 6.1** е показана блокова схема на система, използваща I²C интерфейса. Предаването на информацията се осъществява посредством сигналите:

- **SDA** – данните, предавани серийно по този проводник.
- **SCL** – тактовият сигнал, по който е синхронизирано изпращането бит по бит на данните.

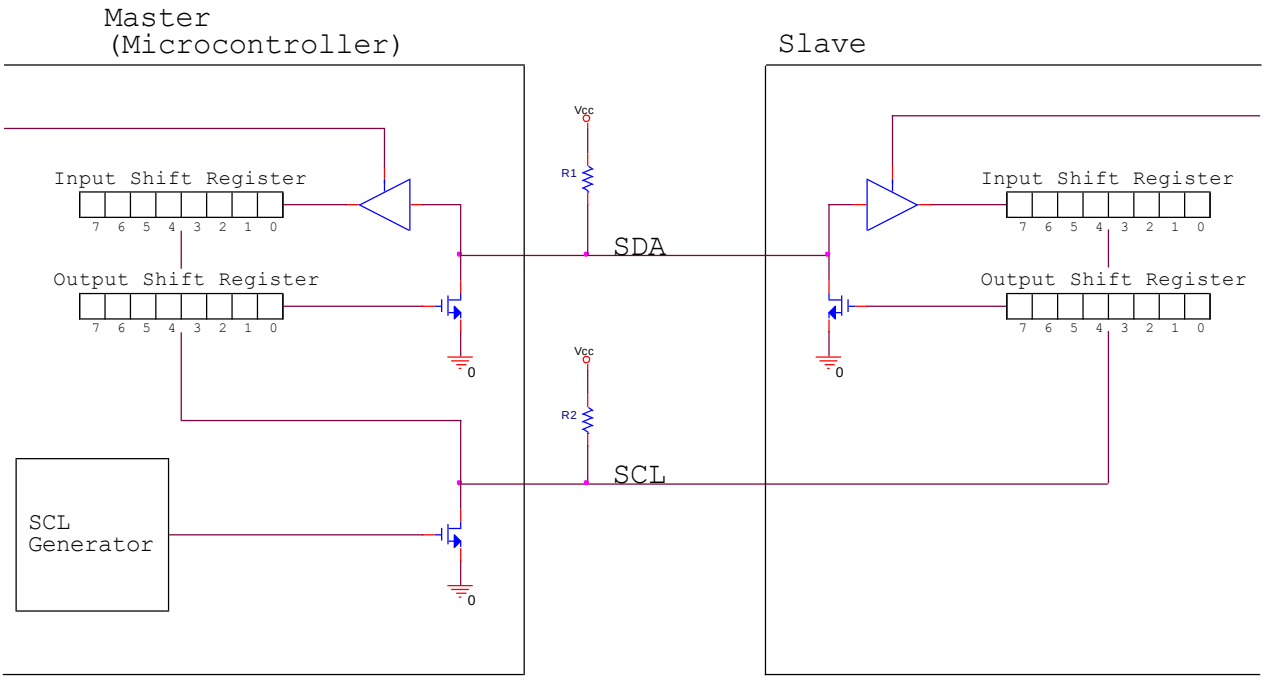
Към двата проводника са свързани изтеглящи резистори към захранването на системата V_{CC} (pull-up). Те са необходими, защото предаването на информация е **двупосочно** – от микроконтролера към подчинената схема и от подчинената схема към микроконтролера. Това означава, че микроконтролерът ще използва изводът си, свързан към SDA, **веднъж като вход и веднъж като изход**. Аналогично – подчиненото устройство ще използва SDA като вход, когато иска да приеме информация и след това като изход, когато иска да предаде информация (**фиг. 6.2**). Тази конфигурация води до една теоретична конфликтна ситуация, в която ако и двата чипа са установили изводите си SDA като изходи и единият е в логическа 1, а другият – в логическа 0, то ще се получи късо съединение между захранващия извод V_{CC} и масата GND. За да се избегне тази възможност се използват **изходи с отворен дрейн** (или колектор), които **изискват pull-up резистор**. **SCL линията също включва такъв резистор**, защото протокола I²C дефинира ситуация, в която някой от чиповете задържа тактовия сигнал в логическа 0.



Фиг. 6.1. Блокова схема на система, използваща I²C интерфейса

Аналогични на I²C са интерфейсите:

- SMBus – създаден от Intel през 1995. Дефинира по-строго изискванията за ниво и времеви параметри (timings) на сигнала.
- I²S – използва се за предаване на цифров звуков сигнал. Съдържа един допълнителен проводник (Word Select), указващ изпращаният байт на кой канал принадлежи (при стерео предаване – ляв или десен).

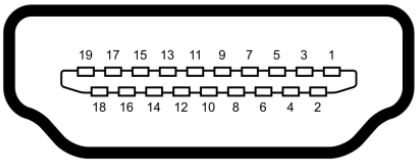


Фиг. 6.2. Хардуерна реализация на I²C интерфейса

I²C интерфейса се характеризира със следните скорости на предаване:

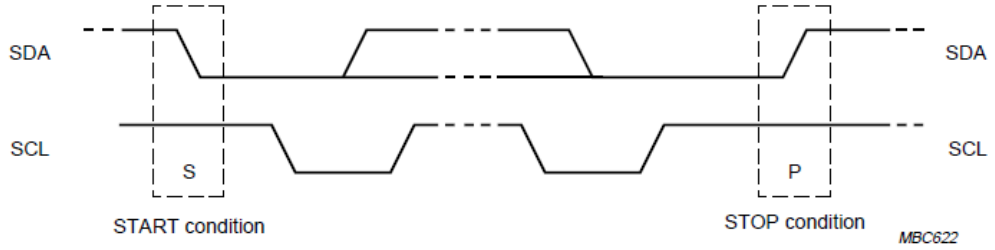
- Оригинална версия от 1982 – 100 kbit/s.
- Версия 1 (1992) – 400 kbit/s (Fast mode).
- Версия 2 (1998) – 3.4 Mbit/s (High-speed mode).
- Версия 3 (2007) – 1 Mbit/s (Fast mode plus).
- Версия 4 (2012) – 5 Mbit/s (Ultra Fast mode).

I²C интерфейса се включва в интегрални схеми с различни функции във вградените системи. АЦП, ЦАП, EEPROM, цифрови потенциометри, температурни датчици, хол датчици и т.н. са само малка част от приложението му. Въпреки че този интерфейс се използва главно за комуникация между ИС в рамките на една вградена система, то не липсват примери за приложения и в комуникацията между две отделни устройства. Например всеки персонален компютър използва I²C като част от HDMI интерфейса (**фиг. 6.3**). С негова помощ се „опознават" поддържаните видео формати от изобразяващото устройство (монитор, телевизор, прожектор и други).



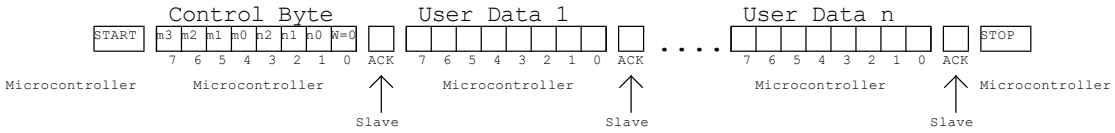
Фиг. 6.3. HDMI куплунг. I²C (извод 15 – SCL и извод 16 – SDA)

I²C комуникацията се осъществява по стандартизиран протокол. Най-общо казано, обменът на данни започва с условие **СТАРТ**, продължава с трансфер на данни и завършва с условие **СТОП**. Условията **старт и стоп са специални събития**, реализирани чрез **комбинации на логически нива и фронтове на SCL и SDA линиите**, които са **уникални** и се **различават от данните**. На **фиг. 6.4** са показани едно старт и стоп условие. **Старт условие** се генерира, когато **SDA линията премине в ниско ниво, докато SCL линията е във високо**. **Стоп условие** се генерира, когато **SDA линията премине във високо ниво, докато SCL линията е във високо**. Между старт и стоп условията се изпращат потребителските данни. Всеки бит от тях трябва да **запази своето състояние, докато SCL е във високо ниво** (в противен случай ще се генерира старт или стоп условие и трансферът ще се наруши), т.е. трансферът на данни се осъществява **по ниво**.



Фиг. 6.4. Състояние на линиите SDA и SCL при START и STOP условие

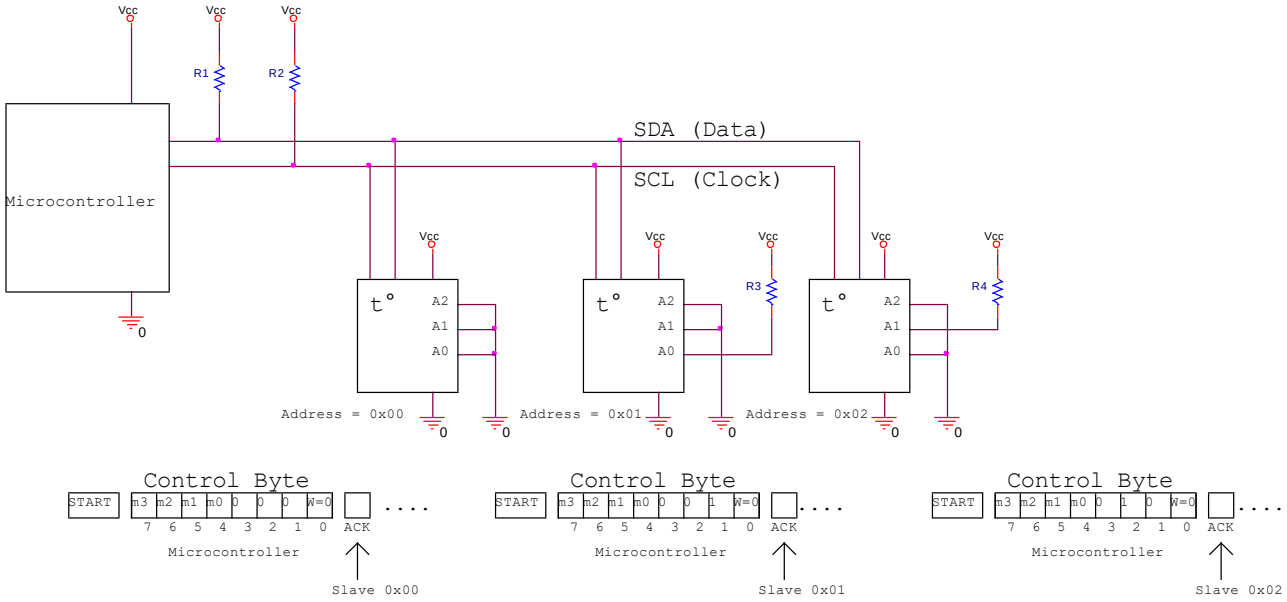
Форматът на данните при **запис** от микроконтролер в подчинена ИС е показан на **фиг. 6.5**.



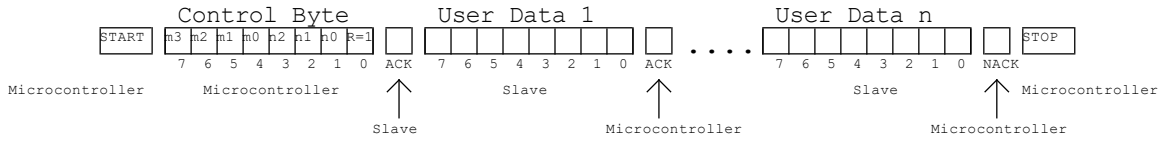
Фиг. 6.5. Форматът на данните при запис от микроконтролер в подчинена ИС

Микроконтролерът генерира СТАРТ условие, след което изпраща **контролен байт**. Този байт **достига до всички подчинени устройства**. Старшата тетрада (битове m0 ÷ m3) съдържа контролен код, част от **7-битов адрес**, който е различен за различните подчинени чипове – трябва да се провери datasheet-а им. Следват три бита, които указват младшата част на 7-битовия адрес на подчиненото устройство (битове n0 ÷ n2). Благодарение на тях към една I^2C шина могат да се свържат повече от един чипове от един и същи вид (на **фиг. 6.6** е показано свързване на три еднакви температурни датчика, чиито адреси са зададени хардуерно с изводи 1, 2 и 3). **Последният бит от контролния байт** е четене/запис (R/$\overline{W}$) и **при запис трябва да е 0**. Ако на I^2C шината има устройство със зададения адрес (битове m0 ÷ m3 и n0 ÷ n2), то трябва да отговори с изпращане на един бит, наречен **ACK** (acknowledge) или още - **потвърждение**. **Неговото ниво е логическа 0**. След това се изпращат потребителските данни User Data 1 ... User Data n и най-накрая се генерира **СТОП условие, с което трансферът приключва**.

Трансферът на данните от подчинена ИС към микроконтролера се определя като **четене** и форматът е показан на **фиг. 6.7**. Да се обърне внимание на предавателя и приемника под всеки байт на **фиг. 6.5** и **фиг. 6.7**! Ако микроконтролерът иска да спре приемането на данни, той трябва да генерира NACK бит (**No Acknowledge**, **логическа 1**) и **след това СТОП условие**.

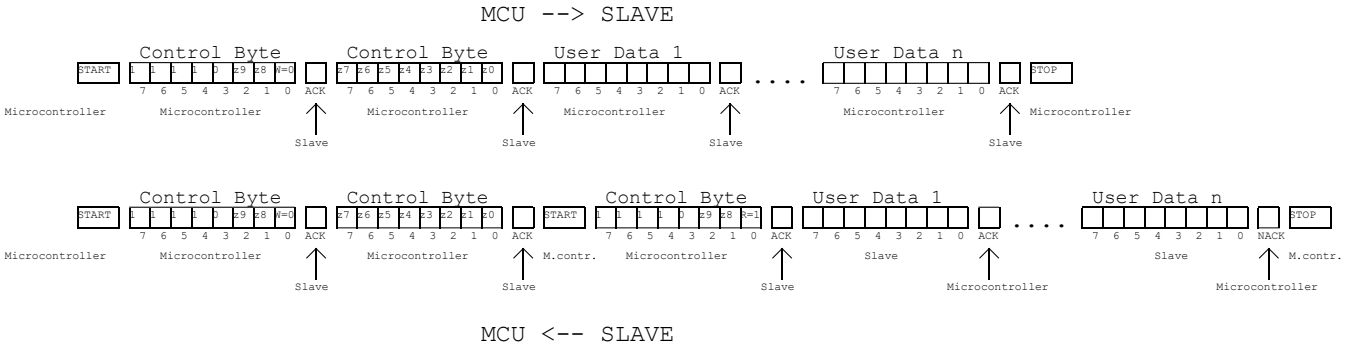


Фиг. 6.6. Свързване на три еднакви температурни датчика, чиито адреси са зададени хардуерно с изводи A0, A1 и A2



Фиг. 6.7. Форматът на данните при четене от подчинена ИС (трансфер от подчинената ИС към микроконтролера)

Отделените седем бита m0 ÷ m3 и n0 ÷ n2 позволяват на една I^{2}C шина да се свържат до 2^7 = 128 чипа (минус един броудкаст адрес и няколко за бъдещо ползване). Версия 1 дефинира някои леки промени в протокола, които позволяват адресиране на 2^{10} = 1024 (т.е. 10-битов адрес) чипа. Те са показани на **фиг. 6.8**. Новото тук е фиксираното число 11110 в контролния байт. То указва, че ще се използва 10-битово адресиране. Следват битове z9 ÷ z0, които са 10-битовия адрес на подчиненото устройство. Да се обърне внимание на двойното изпращане на условие СТАРТ и редуването на W=0 и R=1 при четене (MCU ← SLAVE).

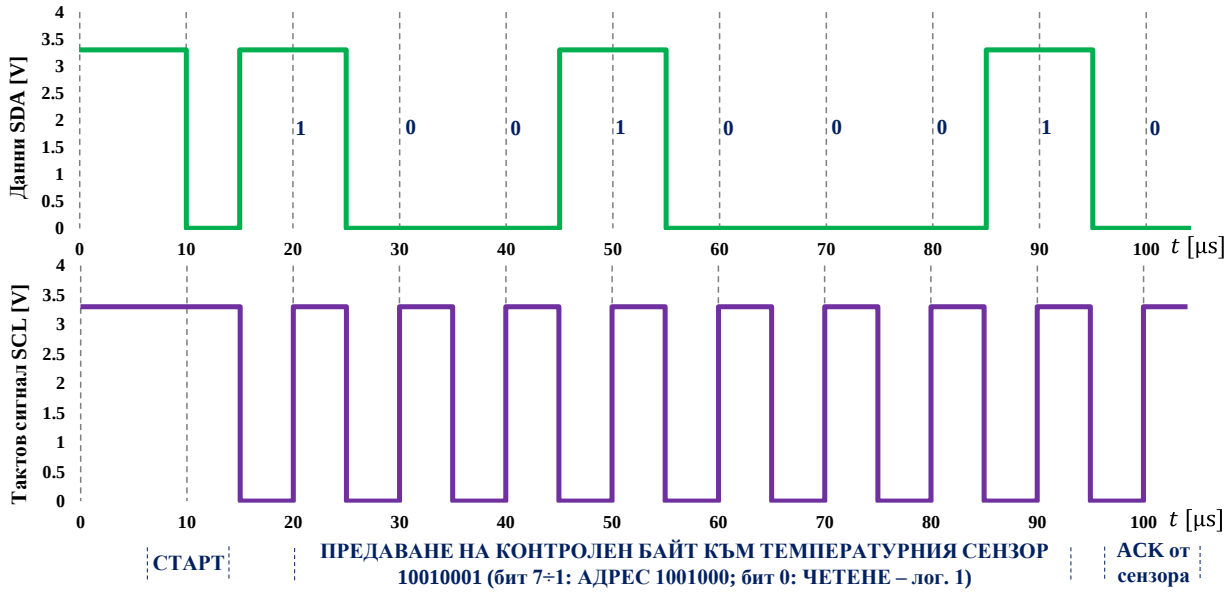


Фиг. 6.8. Протокол на четене и запис на данни при версия 1 на интерфейса

Пример за четене на стойност на температурата от сензор TMP102[1]

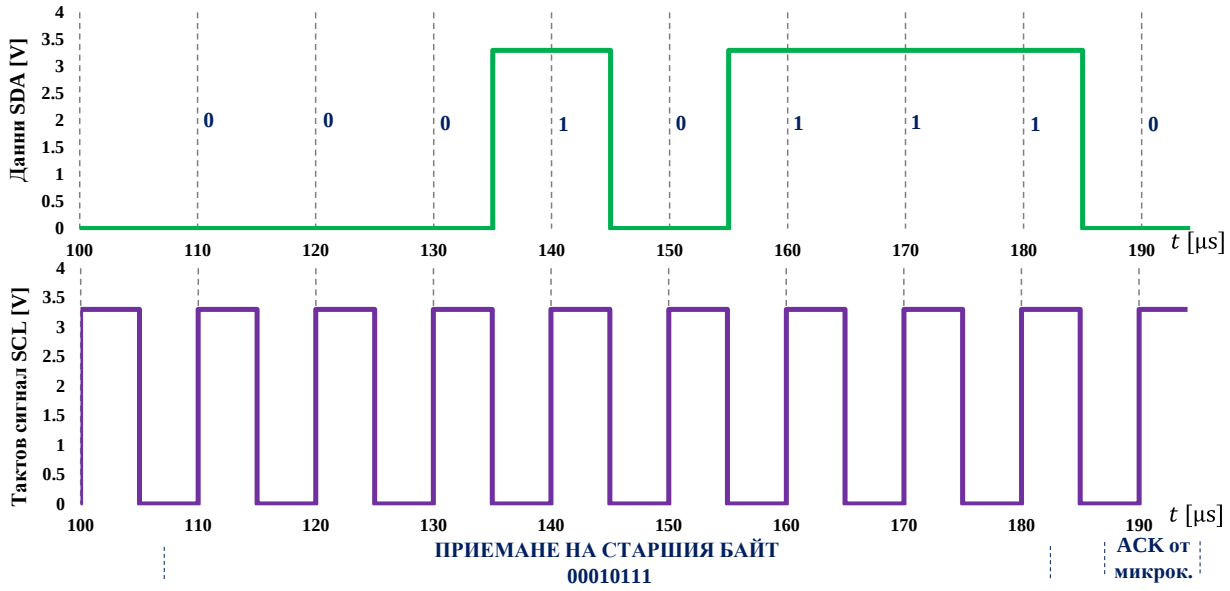
Нека приемем, че инициализацията на сензора е извършена вече. Ако искаме да четем стойността на температурата от сензора, трябва да се извърши следваната последователност:

1. Да се генерира СТАРТ условие от микроконтролера;
2. Да се изпрати от микроконтролера контролния байт, който съдържа 7-битовия адрес + бит за четене/запис. Адресът е 1001000, а когато искаме да четем битът за четене/запис е лог. 1. Тоест контролния байт ще бъде 10010001;
3. Да изчакаме сензорът да „даде потвърждение“ - да изпрати ACK, т.е. SDA линията да бъде лог. 0 при следващия такт;

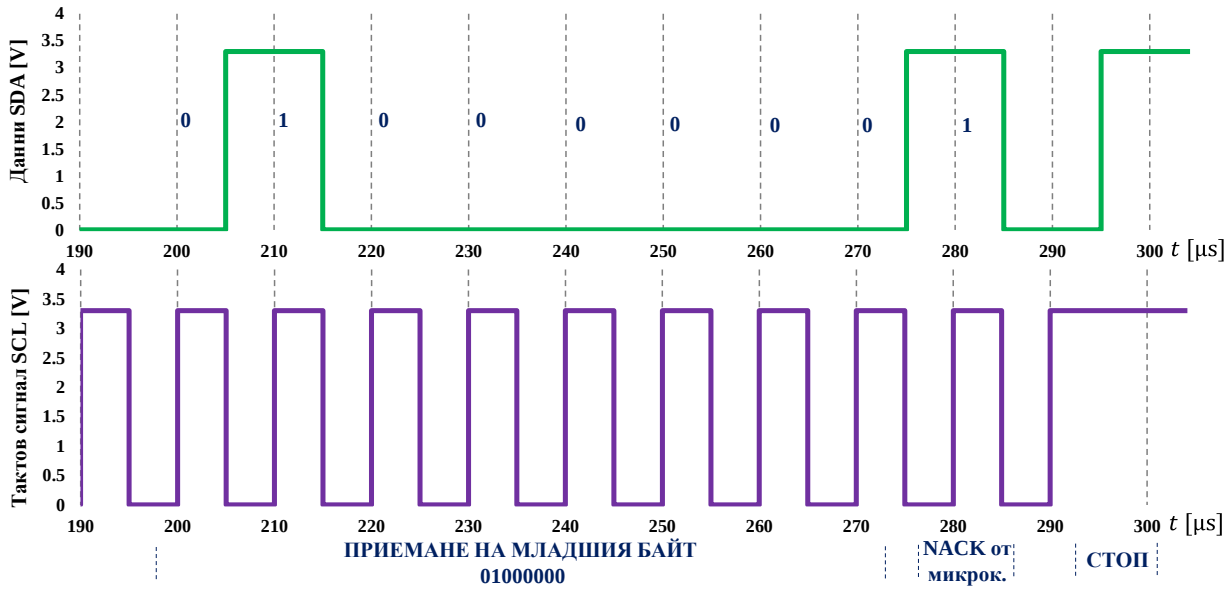


[1] https://www.ti.com/lit/gpn/tmp102

4. Сега сензорът започва да изпраща старшия байт от числото, съответстващо на температурата.
5. Трябва микроконтролера да приеме байта и да даде потвърждение (ACK), т.е. да задържи SDA линията в лог. 0 при следващия такт;



6. След като е приет старшия байт от микроконтролера, сензорът започва да изпраща младшия байт.
7. Трябва микроконтролера да приеме байта и да даде NACK за прекратяване на изпращането на данни от сензора, т.е. да задържи SDA линията в лог. 1 при следващия такт;
8. Да се генерира СТОП условие от микроконтролера;



За да се получи крайната стойност на числото е необходимо да се извършат някои операции:

- Старшият байт да се премести с 8 бита наляво - т.е. ще стане:
  0001011100000000
- Да се извърши операция лог. **ИЛИ** (оператор |) с младшия байт. Тогава резултатът е:
  0001011101000000
- Да се премести надясно с 4 бита полученото число:
  000101110100

Числото в десетичен код е 372. Съгласно документацията на сензора TMP102, температурата може да бъде изчислена от полученото число по следния начин:

$$T[℃] = \text{число} \times 0.0625 = 372 \times 0.0625 = 23.3\ ℃$$

ЗАДАЧИ ЗА ИЗПЪЛНЕНИЕ

1. Да се създаде нов проект с име **Lab_6_1** в папка **/Desktop/MSHT_GR_XX_N**, да се копира програмата на C, чрез която се четат данни от температурния датчик TMP102 посредством I^2C интерфейс. Да се наблюдават SCL и SDA линиите с осцилоскоп и да се свалят осцилограмите.

Datasheet на температурния сензор: https://www.ti.com/lit/gpn/tmp102

2. Да се създаде нов проект с име **Lab_6_2** в папка **/Desktop/MSHT_GR_XX_N** и да се въведе програмата на C, чрез която се реализира запис по I^2C интерфейса. За целта да се запише произволно число в един от регистрите T_{LOW} или T_{HIGH} на температурния датчик TMP102. Да се снемат осцилограмите. Да се сравнят с осцилограмите от предишната задача. Да се обяснят разликите.