**Лабораторна робота №4**

**Тема:** ІнтерфейсI2C

**Мета:** Засвоїти принципи обміну інформацією за інтерфейсом I2C; отримати навики використання інтерфейсу I2C для отримання інформації від датчиків.

**Теоретичні відомості**

Послідовний протокол обміну даними IIC (також названий I2C - Inter-Integrated Circuits, межмікросхемное з'єднання) використовує для передачі даних дві двонаправлені лінії зв'язку, які називаються шина послідовних даних SDA (Serial Data) і шина тактування SCL (Serial Clock). Також є дві лінії для живлення. Шини SDA і SCL підтягуються до шини живлення через резистори.

У мережі є хоча б один **ведучий пристрій** (Master), який ініціалізує передачу даних і генерує сигнали синхронізації. У мережі також є **ведені пристрої** (Slave), які передають дані по запиту ведучого. У кожного віденого пристрою є унікальна адреа, за якою ведучий і звертається до нього. Адреса пристрою вказується в паспорті (datasheet). До однієї шині I2C може бути підключено до 127 пристроїв, в тому числі кілька ведучих. До шини можна підключати пристрої в процесі роботи, тобто вона підтримує «гаряче підключення».

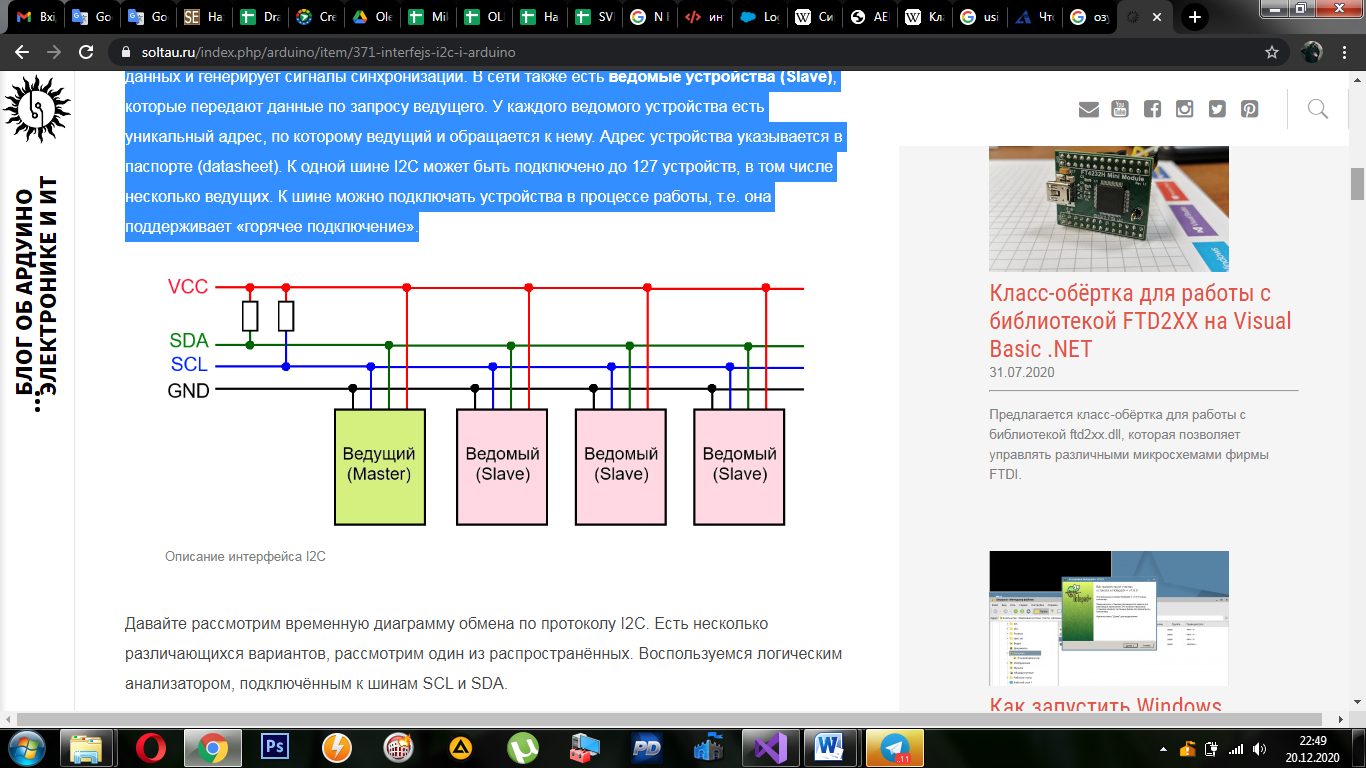


Рис. 1. Описання інтерфейсу I2C

Майстер (ведучий) ініціює обмін. Для цього він починає генерувати тактові імпульси і посилає їх по лінії SCL пачкою з 9-ти штук. Одночасно на лінії даних SDA він виставляє адресу пристрою, з якою необхідно встановити зв'язок, які тактуються першими 7-ми тактовими імпульсами (звідси обмеження на діапазон адреси: 27 = 128 мінус нульова адреса). Наступний біт посилки - це код операції (читання або запис) і ще один біт - біт підтвердження (ACK), що ведений пристрій прийняв запит. Якщо біт підтвердження не прийшов, на цьому обмін закінчується. Або майстер продовжує посилати повторні запити.

**Завдання**

1. Реалізувати ведучий пристрій з інтерфейсом *I*2*C* зі швидкістю, яка не перевищує 100 кбіт/с при тактовій частоті МК 8 МГц.
2. Реалізувати запис та читання інформації від датчика з адресою 1010000.

**Код програми**

**;\*\*\*\* Includes \*\*\*\***

.include "m16def.inc" ;mega16

**;\*\*\*\* Глобальные константы I2C \*\*\*\***

; **для любого порта, 2 контакта в этом порту**

.equ DDRI2C = DDRD ; **Номер порта I2C; регистр направления**

.equ PORTI2C = PORTD ; **Порт выходных данных PORTI2C**

.equ PINI2C = PIND ; **порт входных данных**

.equ SCLP = 1 ; **SCL номер контакта**

.equ SDAP = 0 **; SDA номер контакта**

.equ b\_dir = 0 ; **положение бита направления передачи в i2cadr (не менять, стандарт i2c)**

.equ i2crd = 1 ; **значение бита направления для режима чтения**

.equ i2cwr = 0 ; **значение бита направления для режима записи**

**;\*\*\*\* Глобальные переменные регистра \*\*\*\***

.def i2cdelay= r16 ; **Переменная цикла задержки**

.def i2cdata = r17 ; **Регистр передачи данных I2C**

.def i2cadr = r18 ; **Регистр адреса и направления I2C**

.def i2cstat = r19 ; **Регистр состояния шины I2C**

**;\*\*\*\* Векторы прерывания \*\*\*\***

rjmp RESET ; Reset handle сброс

; no interrupts usage

; ( rjmp EXT\_INT0 ) ; ( IRQ0 handle )

; ( rjmp TIM0\_OVF ) ; ( Timer 0 overflow handle )

; ( rjmp ANA\_COMP ) ; ( Analog comparator handle )

**; \* i2c\_hp\_delay**

**; \* i2c\_qp\_delay**

**; \* hp - половина задержки периода i2c (нормальная: 5,0 мкс / быстрая: 1,3 мкс)**

**; \* qp - четверть задержки периода тактового сигнала i2c (нормальная: 2,5 мкс / быстрая: 0,6 мкс);\***

; For what MHz = = = = 8 MHz

; N=1+((n-1)\*3+1\*2)+4 {n=i2cdelay passages, N=tacts}

; N=4+3\*n

; n=12 N=40 {40 tacts = 5 us with 8 Mhz}

i2c\_hp\_delay:

ldi i2cdelay,12 ; **ldi 1 tact**

i2c\_hp\_delay\_loop:

dec i2cdelay ; **dec 1 tact**

brne i2c\_hp\_delay\_loop ; **brne 1/2 tact/s**

ret ; **ret 4 tact**

i2c\_qp\_delay:

ldi i2cdelay,6 ; **half of half**

i2c\_qp\_delay\_loop:

dec i2cdelay

brne i2c\_qp\_delay\_loop

ret

**;\* i2c\_rep\_start**

**; \* Подтверждает повторяющееся условие запуска и отправляет адрес ведомого.**

**; \* i2cadr - Содержит адрес подчиненного устройства и направление передачи.**

**; \* Флаг переноса - сбрасывается, если ведомое устройство отвечает на адрес.**

**; \* Эта функция должна сопровождаться i2c\_start.;\***

i2c\_rep\_start:

sbi DDRI2C,SCLP ; **force SCL low, SCL = 0**

cbi DDRI2C,SDAP ; **release SDA ~ SDA = Z-state (но с подтягивающим резистором=> 1)**

rcall i2c\_hp\_delay ; **задержка на половину периода**

cbi DDRI2C,SCLP ; release SCL

rcall i2c\_qp\_delay ; **четверть периода задержки**

**;\* i2c\_start**

**;\* Формирует условие запуска и отправляет подчиненный адрес.**

**;\* i2cadr - Содержит адрес подчиненного устройства и направление передачи. (7-битный адрес, 1-битный каталог)**

**; \* Флаг переноса - сбрасывается, если ведомое устройство отвечает на адрес.**

**Эта функция должна сопровождаться i2c\_write.;\***

i2c\_start:

mov i2cdata,i2cadr ; **копировать адрес для передачи register**

sbi DDRI2C,SDAP ; **force SDA low**

rcall i2c\_qp\_delay ; **четверть периода задержки**

; **начальное состояние выполнено**

**;\* i2c\_write**

**;\* Записывает данные (один байт) на шину I2C. Также используется для отправки адрес.**

**; \* i2cdata - Содержит данные для передачи.**

**; \* Флаг переноса - Устанавливается, если ведомое отвечает на передачу.**

**; \* Эта функция должна сопровождаться i2c\_get\_ack.;\***

i2c\_write:

sec ; **установить флаг переноса, установить 1 для понимания границы передачи данных**

rol i2cdata ; **сдвиг в битах переноса и вывода один; необходимо установить 1 для передачи всех нулей**

rjmp i2c\_write\_first

i2c\_write\_bit:

lsl i2cdata ; **если регистр передачи пуст**

i2c\_write\_first:

breq i2c\_get\_ack ; **перейти к получению подтверждения (если регистр передачи пуст)**

sbi DDRI2C,SCLP ; force SCL low

; **данные на линии SDA могут быть изменены в этот момент (пока SCL = 0)**

brcc i2c\_write\_low ; **если бит высокий; разветвление для записи другого значения данных (0 или 1)**

nop ;(**уравнять количество циклов)**

cbi DDRI2C,SDAP ;**release SDA ; if not write\_low then set SDA =1**

rjmp i2c\_write\_high

i2c\_write\_low: **; else**

sbi DDRI2C,SDAP **;force SDA low**

rjmp i2c\_write\_high ;**уравнять количество циклов)**

i2c\_write\_high:

rcall i2c\_hp\_delay ; **задержка на половину периода**

cbi DDRI2C,SCLP **; release SCL ; в этот момент данные будут удалены из строки SDA**

rcall i2c\_hp\_delay ; **задержка на половину периода**

rjmp i2c\_write\_bit

**; \* i2c\_get\_ack**

**; \* Получить ответ о подтверждении ведомого.**

**; \* (в этой версии используется только i2c\_write)**

**; \* Флаг переноса - сбрасывается, если ведомое устройство отвечает на запрос.;\***

i2c\_get\_ack:

sbi DDRI2C,SCLP ; **force SCL low ; 1 на базе (открытый транзистор) = 0 - выход**

cbi DDRI2C,SDAP ; **release SDA ; 0 on base = 1 by pull-up resistors**

**; приемник получает контроль на линиях SDA и SCL**

rcall i2c\_hp\_delay ; **задержка на половину периода**

cbi DDRI2C,SCLP **; release SCL**

i2c\_get\_ack\_wait:

sbis PINI2C,SCLP ; **wait SCL high**

; **(В случае вставки состояний ожидания); цикл ожидания для SCL получить значение 1**

rjmp i2c\_get\_ack\_wait

clc ; **очистить флаг переноса**

sbic PINI2C,SDAP ; **if SDA is high**

sec ; **если SDA 1 = установить флаг переноса**

**; проверить значение бита подтверждения на линии SDA**

; **carry flag = SDA**

rcall i2c\_hp\_delay ; **задержка на половину периода**

ret

**; \* i2c\_do\_transfer**

**; \* Выполняет передачу по шине. Это всего лишь комбинация i2c\_read**

**; \* и i2c\_write для удобства.**

**; \* i2cadr - Должен иметь то же направление, что и при вызове i2c\_start.**

**; \* см. i2c\_read и i2c\_write для получения дополнительной информации.**

**; \* (зависит от типа передачи, чтение или запись)**

**;\*Эта функция должна сопровождаться i2c\_read.;\***

;data transfer in both directions

i2c\_do\_transfer:

sbrs i2cadr,b\_dir ; **если направление 0 = перейти к записи, если направление 1 = перейти к чтению**

rjmp i2c\_write ;**перейти к записи данных**

;

**; \* i2c\_read**

**; \* Считывает данные (один байт) с шины I2C.**

**; \* Флаг переноса - если установлен, ведомому устройству не отправляется подтверждение с указанием последней операции чтения перед СТОПом.**

**; \* Если сброшено, подчиненному отправляется подтверждение с указанием дополнительных данных.**

**; \* i2cdata - Содержит полученные данные.**

**;\*Эта функция должна сопровождаться i2c\_put\_ack.;\***

i2c\_read:

rol i2cstat ; **сохранить бит подтверждения в i2cstat**  ; **(используется i2c\_put\_ack)**

ldi i2cdata,0x01 ; **data = 0x01 как граница данных**

i2c\_read\_bit: ; **do**

sbi DDRI2C,SCLP ;**force SCL low ; ведомое устройство может помещать данные в линию SDA**

rcall i2c\_hp\_delay **; задержка на половину периода**

cbi DDRI2C,SCLP ;**release SCL ; данные готовы к чтению (текущий бит)**

rcall i2c\_hp\_delay ; **задержка на половину периода**

clc ; **clear carry flag**

sbic PINI2C,SDAP ; **if SDA is high**

sec ; **установить флаг переноса**

; carry = SDA

rol i2cdata ; **храение бита данных; сохранить флаг переноса в регистре данных (побитно)**

brcc i2c\_read\_bit ; **пока не заполнен регистр приема; отметка 1 (граница) от первой загрузки**

**; \* i2c\_put\_ack**

**; \* (в этой версии используется только i2c\_read)**

**;\* ВОЗВРАЩЕНИЕ**

**;\* никто;\***

i2c\_put\_ack:

sbi DDRI2C,SCLP ; **force SCL low**

; SDA can be changed

ror i2cstat ; **получить бит состояния (также бит состояния)**

brcc i2c\_put\_ack\_low ; **if ack bit low goto assert low**

cbi DDRI2C,SDAP ; **if ack bit not low => release SDA**

rjmp i2c\_put\_ack\_high

i2c\_put\_ack\_low: ; **else**

sbi DDRI2C,SDAP ; **if ack bit low => force SDA low**

i2c\_put\_ack\_high:

rcall i2c\_hp\_delay ; **задержка на половину периода**

cbi DDRI2C,SCLP **; release SCL**

; **ведомое устройство может читать бит подтверждения из линии SDA**

i2c\_put\_ack\_wait: ; **цикл ожидания для SCL = 1**

sbis PINI2C,SCLP ; **wait SCL high**

rjmp i2c\_put\_ack\_wait

rcall i2c\_hp\_delay ; **задержка на половину периода**

ret

**; \* i2c\_stop**

**; \* Утвердить условие остановки.;\***

i2c\_stop:

sbi DDRI2C,SCLP ; **force SCL low**

sbi DDRI2C,SDAP ; **force SDA low**

rcall i2c\_hp\_delay ; **half period delay**

cbi DDRI2C,SCLP ; **release SCL**

rcall i2c\_qp\_delay ; **четверть периода задержки**

cbi DDRI2C,SDAP ; **release SDA**

rcall i2c\_hp\_delay ; **half period delay**

ret

**; \* i2c\_init**

**; \* Инициализация интерфейса шины I2C.**

**; \* Вызовите эту функцию один раз для инициализации шины I2C.**

**; \* Выводы PORTI2C и DDRI2C, не используемые интерфейсом шины I2C, будут**

**; \* устанавливать Hi-Z (!).**

**; \* Эту функцию можно комбинировать с другими инициализациями PORTI2C.**

i2c\_init:

clr i2cstat ; **очистить регистр состояния I2C (используется как временный регистр)**

out PORTI2C,i2cstat **;установить контакты I2C на открытый коллектор**

out DDRI2C,i2cstat ; **direction (0 - input, Z-state/ 1 - output, output with 0)**

ret

**; \* main - Тест реализации мастера I2C**

**; \* Инициализирует интерфейс I2C и показывает пример его использования.;\***

RESET:

main: rcall i2c\_init ; **инициализировать интерфейс I2C**

**;\*\*\*\* Запись данных => Adr(00) = 0x55 \*\*\*\***

;ldi i2cadr,$50

;lsl i2cadr

;ori i2cadr,i2cwr

;435-437 ~ 439 line

;ldi i2cadr,$A0+i2cwr ; **Установка адреса устройства; $ A0 - адрес со смещением влево на 1 бит**

;rcall i2c\_start ; **Отправить условие начала и адрес**

**;10bit address**

ldi i2cadr,0b11110xx0+i2cwr ; **XX - старшие биты адреса**

rcall i2c\_start

ldi i2cdata,0bxxxxxxxx ; **XXXXXXXX - LOW bits of address**

rcall i2c\_do\_transfer

**; необходимо проверить бит переноса, и если он не равен 0, значит, нет ведомого устройства с этим адресом**

ldi i2cdata,$00 ; **Написать адрес слова (0x00); отправить данные** rcall i2c\_do\_transfer ; Execute transfer

ldi i2cdata,$55 **; Установить данные записи на 01010101b**

rcall i2c\_do\_transfer ; **Выполнить перевод**

rcall i2c\_stop ; **Отправить условие остановки**

**;\*\*\*\* Read data => i2cdata = Adr(00) \*\*\*\* ;** **комбинированный формат (не только чтение)**

;ldi i2cadr,$A0+i2cwr ; **Установика адреса устройства и запись**

;rcall i2c\_start ; **Отправить условие начала и адрес**

;10bit address

ldi i2cadr,0b11110xx0+i2cwr ; **XX - HIGH bits of address**

rcall i2c\_start

ldi i2cdata,0bxxxxxxxx ; **XXXXXXXX - LOW bits of address**

rcall i2c\_do\_transfer

ldi i2cdata,$00 ; **Write word address**

rcall i2c\_do\_transfer ; **Выполнить перевод**

;ldi i2cadr,$A0+i2crd ; **Установить адрес устройства и прочитать; не прерывать передачу, а менять направление**

;rcall i2c\_rep\_start ; **Отправить повторное условие запуска и адрес**

; **10-битный адрес; изменить направление и не отправлять полный адрес**

ldi i2cadr,0b11110xx0+i2crd ; **XX - HIGH bits of address**

rcall i2c\_rep\_start

sec ; **Установить отсутствие подтверждения (если за чтением следует условие остановки)**

rcall i2c\_do\_transfer ; **Execute transfer (read)**

rcall i2c\_stop ; **Отправить условие остановки - releases bus**

rjmp main ; **Loop forewer**

**;\*\*\*\* End of File \*\*\*\***

**Висновок:** під час виконання лабораторної роботи ми засвоїли принципи обміну інформацією за інтерфейсом I2C; отримали навики використання інтерфейсу I2C для отримання інформації від датчиків. Реалізували ведучий пристрій з інтерфейсом I2C і запис та читання інформації від датчика з адресою згідно завдання.