**Лабораторная работа №4**

**Интерфейс I2C (ведущее устройство)**

**Цель:**

1. Изучить принципы обмена информацией по интерфейсу I2C;

2. Получить навыки использования интерфейса I2C для получения информации от датчиков.

**Задание:**

1. Реализовать ведущее устройство с интерфейсом I2C со скоростью, не превышающей 100 кбит / с при тактовой частоте МК 8 МГц;

2. Реализовать запись и чтение информации от датчика с адресом 1010000.

**Теория**

Шина I2C поддерживает любую технологию изготовления микросхем (НМОП, КМОП, биполярную). Две линии, данных (SDA) и синхронизации (SCL), служат для переноса информации. Каждое устройство распознается по уникальному адресу и может работать как передатчик или приёмник, в зависимости от назначения устройства. Кроме того, устройства могут быть классифицированы как ведущие и ведомые при передаче данных. Ведущий - это устройство, которое инициирует передачу данных и вырабатывает сигналы синхронизации. При этом любое адресуемое устройство считается ведомым по отношению к ведущему.

Шина I2C допускает несколько ведущих. Это означает, что более чем одно устройство, способное управлять шиной, может быть подключено к ней. Поскольку в качестве ведущих обычно выступают микроконтроллеры, давайте рассмотрим пример пересылки данных между двумя микроконтроллерами, подключенными к шине. Пример покажет взаимоотношения передатчик-приемник и ведущий-ведомый, существующие в шине I2C. Необходимо заметить, что эти отношения не постоянны, а зависят только от направления пересылки данных в данный момент времени.

**Код программы:**

Includes \*\*\*\*

.include "m16def.inc" ;mega16 ; change if an other device is used

;\*\*\*\* Global I2C Constants \*\*\*\*

;for any port and 2 pins in that port

.equ DDRI2C = DDRD ; I2C Port number ; direction register

.equ PORTI2C = PORTD ; PORTI2C output data port

.equ PINI2C = PIND ; input data port

.equ SCLP = 1 ; SCL Pin number

.equ SDAP = 0 ; SDA Pin number

.equ b\_dir = 0 ; transfer direction bit position in i2cadr (don`t change, standart i2c)

.equ i2crd = 1 ; direction bit value for read mode

.equ i2cwr = 0 ; direction bit value for write mode

;\*\*\*\* Global Register Variables \*\*\*\*

.def i2cdelay= r16 ; Delay loop variable

.def i2cdata = r17 ; I2C data transfer register

.def i2cadr = r18 ; I2C address and direction register

.def i2cstat = r19 ; I2C bus status register

;\*\*\*\* Interrupt Vectors \*\*\*\*

rjmp RESET ; Reset handle

; no interrupts usage

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

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

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

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* FUNCTION

;\* i2c\_hp\_delay

;\* i2c\_qp\_delay

;\*

;\* DESCRIPTION

;\* hp - half i2c clock period delay (normal: 5.0us / fast: 1.3us)

;\* qp - quarter i2c clock period delay (normal: 2.5us / fast: 0.6us)

;\*

;\* SEE DOCUMENTATION !!! for MHz

;\*

;\* USAGE

;\* no parameters

;\*

;\* RETURN

;\* none

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; 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

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* FUNCTION

;\* i2c\_rep\_start

;\*

;\* DESCRIPTION

;\* Assert repeated start condition and sends slave address.

;\*

;\* USAGE

;\* i2cadr - Contains the slave address and transfer direction.

;\*

;\* RETURN

;\* Carry flag - Cleared if a slave responds to the address.

;\*

;\* NOTE

;\* IMPORTANT! : This funtion must be directly followed by i2c\_start.

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i2c\_rep\_start:

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

cbi DDRI2C,SDAP ; release SDA ~ SDA = Z-state (but with pull-up resistor => 1)

rcall i2c\_hp\_delay ; half period delay

cbi DDRI2C,SCLP ; release SCL

rcall i2c\_qp\_delay ; quarter period delay

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* FUNCTION

;\* i2c\_start

;\*

;\* DESCRIPTION

;\* Generates start condition and sends slave address.

;\*

;\* USAGE

;\* i2cadr - Contains the slave address and transfer direction. (7 bit adress, 1 bit dir)

;\*

;\* RETURN

;\* Carry flag - Cleared if a slave responds to the address.

;\*

;\* NOTE

;\* IMPORTANT! : This funtion must be directly followed by i2c\_write.

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i2c\_start:

mov i2cdata,i2cadr ; copy address to transmitt register

sbi DDRI2C,SDAP ; force SDA low

rcall i2c\_qp\_delay ; quarter period delay

;start state accomplished

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* FUNCTION

;\* i2c\_write

;\*

;\* DESCRIPTION

;\* Writes data (one byte) to the I2C bus. Also used for sending

;\* the address.

;\*

;\* USAGE

;\* i2cdata - Contains data to be transmitted.

;\*

;\* RETURN

;\* Carry flag - Set if the slave respond transfer.

;\*

;\* NOTE

;\* IMPORTANT! : This funtion must be directly followed by i2c\_get\_ack.

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i2c\_write:

sec ; set carry flag, set 1 for understand data transfer boundary

rol i2cdata ; shift in carry and out bit one ;must set 1 for transfer all zeros

rjmp i2c\_write\_first

i2c\_write\_bit:

lsl i2cdata ; if transmit register empty

i2c\_write\_first:

breq i2c\_get\_ack ; goto get acknowledge (if transmit register empty)

sbi DDRI2C,SCLP ; force SCL low

; data on SDA line can be changed at this moment (while SCL = 0)

brcc i2c\_write\_low ; if bit high ; forking for writing different data value (0 or 1)

nop ; (equalize number of cycles)

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 ; (equalize number of cycles)

i2c\_write\_high:

rcall i2c\_hp\_delay ; half period delay

cbi DDRI2C,SCLP ; release SCL ; in that moment data would be readed from SDA line

rcall i2c\_hp\_delay ; half period delay

rjmp i2c\_write\_bit

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* FUNCTION

;\* i2c\_get\_ack

;\*

;\* DESCRIPTION

;\* Get slave acknowledge response.

;\*

;\* USAGE

;\* (used only by i2c\_write in this version)

;\*

;\* RETURN

;\* Carry flag - Cleared if a slave responds to a request.

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i2c\_get\_ack:

sbi DDRI2C,SCLP ; force SCL low ; 1 on base (open transistor) = 0 - output

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

;reciever get control on SDA and SCL lines

rcall i2c\_hp\_delay ; half period delay

cbi DDRI2C,SCLP ; release SCL

i2c\_get\_ack\_wait:

sbis PINI2C,SCLP ; wait SCL high

;(In case wait states are inserted) ; wait loop for SCL get value 1

rjmp i2c\_get\_ack\_wait

clc ; clear carry flag

sbic PINI2C,SDAP ; if SDA is high

sec ; if SDA 1 = set carry flag

; check acknowledge bit value oon SDA line

; carry flag = SDA

rcall i2c\_hp\_delay ; half period delay

ret

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* FUNCTION

;\* i2c\_do\_transfer

;\*

;\* DESCRIPTION

;\* Executes a transfer on bus. This is only a combination of i2c\_read

;\* and i2c\_write for convenience.

;\*

;\* USAGE

;\* i2cadr - Must have the same direction as when i2c\_start was called.

;\* see i2c\_read and i2c\_write for more information.

;\*

;\* RETURN

;\* (depends on type of transfer, read or write)

;\*

;\* NOTE

;\* IMPORTANT! : This funtion must be directly followed by i2c\_read.

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;data transfer in both directions

i2c\_do\_transfer:

sbrs i2cadr,b\_dir ; if direction 0 = goto write, if direction 1 = goto read

rjmp i2c\_write ; goto write data

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* FUNCTION

;\* i2c\_read

;\*

;\* DESCRIPTION

;\* Reads data (one byte) from the I2C bus.

;\*

;\* USAGE

;\* Carry flag - If set no acknowledge is given to the slave

;\* indicating last read operation before a STOP.

;\* If cleared acknowledge is given to the slave

;\* indicating more data.

;\*

;\* RETURN

;\* i2cdata - Contains received data.

;\*

;\* NOTE

;\* IMPORTANT! : This funtion must be directly followed by i2c\_put\_ack.

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i2c\_read:

rol i2cstat ; store acknowledge bit in i2cstat

; (used by i2c\_put\_ack)

ldi i2cdata,0x01 ; data = 0x01 as a data boundary

i2c\_read\_bit: ; do

sbi DDRI2C,SCLP ; force SCL low ; slave can put data on SDA line

rcall i2c\_hp\_delay ; half period delay

cbi DDRI2C,SCLP ; release SCL ; data is ready for read (current bit)

rcall i2c\_hp\_delay ; half period delay

clc ; clear carry flag

sbic PINI2C,SDAP ; if SDA is high

sec ; set carry flag

; carry = SDA

rol i2cdata ; store data bit ; store carry flag in data register (bit by bit)

brcc i2c\_read\_bit ; while receive register not full ; check 1 (boundary) from first loading

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* FUNCTION

;\* i2c\_put\_ack

;\*

;\* DESCRIPTION

;\* Put acknowledge.

;\*

;\* USAGE

;\* (used only by i2c\_read in this version)

;\*

;\* RETURN

;\* none

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i2c\_put\_ack:

sbi DDRI2C,SCLP ; force SCL low

; SDA can be changed

ror i2cstat ; get status bit (ack bit also)

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 ; half period delay

cbi DDRI2C,SCLP ; release SCL

; slave can read ack bit from SDA line

i2c\_put\_ack\_wait: ; wait loop for SCL = 1

sbis PINI2C,SCLP ; wait SCL high

rjmp i2c\_put\_ack\_wait

rcall i2c\_hp\_delay ; half period delay

ret

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* FUNCTION

;\* i2c\_stop

;\*

;\* DESCRIPTION

;\* Assert stop condition.

;\*

;\* USAGE

;\* No parameters.

;\*

;\* RETURN

;\* None.

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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 ; quarter period delay

cbi DDRI2C,SDAP ; release SDA

rcall i2c\_hp\_delay ; half period delay

ret

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* FUNCTION

;\* i2c\_init

;\*

;\* DESCRIPTION

;\* Initialization of the I2C bus interface.

;\*

;\* USAGE

;\* Call this function once to initialize the I2C bus. No parameters

;\* are required.

;\*

;\* RETURN

;\* None

;\*

;\* NOTE

;\* PORTI2C and DDRI2C pins not used by the I2C bus interface will be

;\* set to Hi-Z (!).

;\*

;\* COMMENT

;\* This function can be combined with other PORTI2C initializations.

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i2c\_init:

clr i2cstat ; clear I2C status register (used

; as a temporary register)

out PORTI2C,i2cstat ; set I2C pins to open colector

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

ret

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* PROGRAM

;\* main - Test of I2C master implementation

;\*

;\* DESCRIPTION

;\* Initializes I2C interface and shows an example of using it.

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

RESET:

main: rcall i2c\_init ; initialize I2C interface

;\*\*\*\* Write data => Adr(00) = 0x55 \*\*\*\*

;ldi i2cadr,$50

;lsl i2cadr

;ori i2cadr,i2cwr

;435-437 ~ 439 line

;ldi i2cadr,$A0+i2cwr ; Set device address and write ; $A0 - left-shifted address by 1 bit

;rcall i2c\_start ; Send start condition and address

;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

; it is necessary to check the carry bit and if it is not 0, then there is no slave device with that address

ldi i2cdata,$00 ; Write word address (0x00) ; send data

rcall i2c\_do\_transfer ; Execute transfer

ldi i2cdata,$55 ; Set write data to 01010101b

rcall i2c\_do\_transfer ; Execute transfer

rcall i2c\_stop ; Send stop condition

;\*\*\*\* Read data => i2cdata = Adr(00) \*\*\*\* ; combined format (not only read)

;ldi i2cadr,$A0+i2cwr ; Set device address and write

;rcall i2c\_start ; Send start condition and address

;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 ; Execute transfer

;ldi i2cadr,$A0+i2crd ; Set device address and read ; not interrupt transfer, but change direction

;rcall i2c\_rep\_start ; Send repeated start condition and address

;10bit address ; change direction and dont send full address

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

rcall i2c\_rep\_start

sec ; Set no acknowledge (if read is followed by a stop condition)

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

rcall i2c\_stop ; Send stop condition - releases bus

rjmp main ; Loop forewer

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

**Вывод.** В результате выполнения лабораторной работы было усвоено принципы обмена информацией по интерфейсу I2C и получены навыки использования интерфейса I2C для получения информации от датчиков.