**Лабораторная работа № 5**

**Интерфейс 1-Wire**

**Цель:**

1. Изучить принципы обмена информацией по интерфейсу 1 Wire;

2. Получить навыки использования интерфейса 1 Wire для получения информации от датчиков.

**Задание:** Реализация ведущего устройства на шине 1-Wire. Передача и приём битов, байтов и пакета состоящего из импульса сброса, присутствия, команды, данных.

**Теория:**

Интерфейс 1 Wire является последовательным асинхронным полудуплексным интерфейсом передачи данных, который часто используется для обеспечения сообщения МК и периферийных устройств. Интерфейс 1 Wire использует одну двунаправленную линию данных, которую подтянуто к напряжению питания. Также по этой линии может передаваться паразитную питания для низкопотребляющие датчиков. Линия данных имеет выход типа "открытый коллектор" или "открытый сток". Используется топология "общая шина".

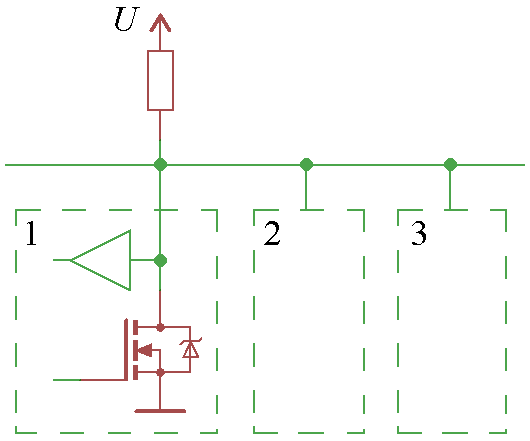


Рис. 1. Физический уровень интерфейса 1 Wire

**Код программы:**

.include "m16def.inc" // Подключаем заголовочный файл

.def temp = r16 // Присваиваем регистрам символические имена

.def razr1 = r17

.def razr2 = r18

.def razr3 = r19

.equ FREQ = 8000000 // Частота процессора

.equ W1\_DDR = DDRA // Регистр порта Датчика

.equ W1\_PORT = PORTA // Регистр порта Датчика

.equ W1\_PIN = PINA // Регистр порта Датчика

.equ W1\_BIT = 0 // Бит порта на котором датчик

.dseg

Trm: // Ячейки ОЗУ под температуру

.byte 2

.cseg // Программный сегмент

.org 0 // Вектор прерывания Reset

jmp Reset // Переходим на Reset

Reset:

ldi temp, high(RAMEND) // Инициализируем стек

out sph, temp

ldi temp, low(RAMEND)

out spl, temp

rcall W1\_Sbros // Сбрасываем шину и проверяем есть ли датчик

cpi r17, 0 // После сброса, если датчик есть в r17 будет 1

breq Proga\_not // Если нету, уходим в бесконечность лени

rcall W1\_Init\_12bit // Перестраиваем конфигурационный байт на 12 битную схему работы

Proga:

rcall W1\_Sbros // Сбрасываем шину и проверяем есть ли датчик

cpi r17, 0 // После сброса, если датчик есть в r17 будет 1

breq Proga\_not // Если нету, уходим в бесконечность лени

rcall W1\_ConvTemp // Говорим датчику конвертировать температуру

rcall Delay // Подождем пару секунд, пока датчик закончит конвертировать

rcall W1\_Sbros // Сбрасываем шину и проверяем есть ли датчик

cpi r17, 0

breq Proga\_not // Если нету, уходим в бесконечность лени

rcall W1\_ReadMem // Читаем в ОЗУ текущую температуру

rjmp Proga // Зацикливаем

Proga\_not: // Сюда приходим если датчика нету

rjmp Proga\_not // и ничего больше не делаем

//////////////////////////// Начало подпрограмм 1-Wire

W1\_Sbros: // Сброс шины и проверка датчик на месте ли

//lds r16, W1\_BIT // Записываем в r16 ножку где датчик(не полезная)

sbi W1\_DDR, W1\_BIT // Ногу на выход

cbi W1\_PORT, W1\_BIT // Опрокидываем вывод на землю

rcall W1\_DelayH // Задержка 480 мкс, для сброса

cbi W1\_DDR, W1\_BIT // Ногу на вход

rcall W1\_DelayI // Ждем тайм слот 70 мкс

sbis W1\_PIN, W1\_BIT // Пропускаем следующую строку, если бит порта в 1

ldi r17, 1 // И установим сигнальный регистр в 1

sbic W1\_PIN, W1\_BIT // Пропускаем следующую строку, если бит порта в 0

ldi r17, 0 // И установим сигнальный регистр в 0

rcall W1\_DelayJ //Ждем тайм слот 410 мкс

ret // Если датчик на месте, в r17 по выходу отсюда будет 1, в противном случае 0

W1\_ReadMem: // Чтение памяти регистров температуры

ldi r16, 0xCC // Пошлем команду 0xCC, это пропустить уникальный номер датчика

rcall ds\_byte\_wr // Так как он у нас один на проводе

ldi r16, 0xBE // Говорим датчику, что мы сейчас будем читать

rcall ds\_byte\_wr // Запуливаем байт

rcall ds\_byte\_rd // А тут уже начинаем читать, прочитали первый

sts Trm, r16 // И запулили его в память, по метке Trm

rcall ds\_byte\_rd // Читаем второй

sts Trm+1, r16 // И запулили его в память, по метке Trm+1

ret

W1\_ConvTemp: // Подпрограмма конвертирования температуры

ldi r16, 0xCC // Пропускаем уникальный номер датчика

rcall ds\_byte\_wr

ldi r16, 0x44 // Говорим что надо бы сконвертировать температуру, этот процесс занимает 750

rcall ds\_byte\_wr // миллисекунд, поэтому идем что-то делать, или ленится

ret

W1\_Init\_12bit: // Подпрограмма перестройки на 12 бит температуры

ldi r16, 0xCC // Пропускаем уникальный номер датчика(осуществляется пропуск адреса,CC-команда пзу)

rcall ds\_byte\_wr // Спуливаем в датчик

ldi r16, 0x4E // Говорим что сейчас будем писать в RAM регистры датчика(4Е запись данных в регистры устройства)

rcall ds\_byte\_wr // Спуливаем в датчик

ldi r16, 0xFF // 0xFF записываем в первые 2 регистра, это регистры температуры, он нам не

rcall ds\_byte\_wr // нужен, поэтому их оставляем в стандартном состоянии

ldi r16, 0xFF // 0xFF второй байт температуры

rcall ds\_byte\_wr // Спуливаем на порт

ldi r16, 0x7F // А вот тут говорим что 12 бит - 7F, или 1F - 9бит, 3F - 10 бит, 5F - 11 бит

rcall ds\_byte\_wr // Спуливаем на порт

ret

ds\_byte\_rd: // Подпрограмма чтения данных в регистр r16 с 1 Wire

ldi r17, 8 // Пишем в r17 - 8, т.к. у нас в бит в регистре

clr r16 // Чистим r16, сюда будем читать данные

ds\_byte\_rd\_0:

sbi W1\_DDR, W1\_BIT // Вывод на выход

cbi W1\_PORT, W1\_BIT // Опрокидываем вывод на землю

rcall W1\_DelayA // Ждем 6 микросекунд

cbi W1\_DDR, W1\_BIT // Вывод на вход

rcall W1\_DelayE // Ждем 9 микросекунд

sbis W1\_PIN, W1\_BIT

clc // Очищаем бит C = 0

sbic W1\_PIN, W1\_BIT

sec // Очищаем бит C = 1

ror r16 // Производим циклический сдвиг вправо через С

rcall W1\_DelayF // Ждем 55 микросекунд

dec r17 // Понижаем на 1 регистр r17

brne ds\_byte\_rd\_0 // если не равен 0 вращаемся в цикле

ret

// Побитное выставление команды/данных на шину 1wire

ds\_byte\_wr: // Подпрограмма записи данных из регистра r16 в датчик

ldi r17, 8 // Пишем в r17 - 8, т.к. у нас 8 бит в регистре

ds\_byte\_wr0:

sbi W1\_DDR, W1\_BIT // Вывод на выход

cbi W1\_PORT, W1\_BIT // Опрокидываем вывод на землю

sbrc r16, 0 // Проверим, в r16 бит 0 очищен или установлен

rjmp ds\_byte\_write\_1 // Если установлен перейдем по этой метке

rjmp ds\_byte\_write\_0 // Если очищен перейдем по этой метке

ds\_byte\_wr1:

lsr r16 // Логический сдвиг вправо

dec r17 // Понижаем r17 на 1

brne ds\_byte\_wr0 // Если не равен 0, вращаемся в цикле

ret // Выход из подпрограммы

ds\_byte\_write\_0: // Запись 0

rcall W1\_DelayC // Ждем 60 микросекунд

cbi W1\_DDR, W1\_BIT // Вывод на вход

rcall W1\_DelayD // Ждем 10 микросекунд

rjmp ds\_byte\_wr1

ds\_byte\_write\_1: // Запись 1

rcall W1\_DelayA // Ждем 6 микросекунд

cbi W1\_DDR, W1\_BIT // Вывод на вход

rcall W1\_DelayB // Ждем 64 микросекунд

rjmp ds\_byte\_wr1

W1\_DelayA: // Задержка 6 mcs

ldi XH, high(FREQ/1000000)

ldi XL, low(FREQ/1000000)

rcall W1\_Delay

ret

W1\_DelayB: // Задержка 64 mcs

ldi XH, high(FREQ/65000)

ldi XL, low(FREQ/65000)

rcall W1\_Delay

ret

W1\_DelayC: // Задержка 60 mcs

ldi XH, high(FREQ/68000)

ldi XL, low(FREQ/68000)

rcall W1\_Delay

ret

W1\_DelayD: // Задержка 10 mcs

ldi XH, high(FREQ/500000)

ldi XL, low(FREQ/500000)

rcall W1\_Delay

ret

W1\_DelayE: // Задержка 9 mcs

ldi XH, high(FREQ/600000)

ldi XL, low(FREQ/600000)

rcall W1\_Delay

ret

W1\_DelayF: // Задержка 55 mcs

ldi XH, high(FREQ/75000)

ldi XL, low(FREQ/75000)

rcall W1\_Delay

ret

W1\_DelayH: // Задержка 480 mcs

ldi XH, high(FREQ/8332)

ldi XL, low(FREQ/8332)

rcall W1\_Delay

ret

W1\_DelayI: // Задержка 70 mcs

ldi XH, high(FREQ/58000)

ldi XL, low(FREQ/58000)

rcall W1\_Delay

ret

W1\_DelayJ: // Задержка 410 mcs

ldi XH, high(FREQ/9756)

ldi XL, low(FREQ/9756)

rcall W1\_Delay

ret

W1\_Delay: // Подпрограмма воспроизведения задержки

sbiw XH:XL, 1 // Вычитаем единицу из регистровой пары

brne W1\_Delay // Если не равно 0 крутимся в цикле

ret // Выход из подпрограммы

//////////////////////////// Конец подпрограмм 1-Wire

Delay: // Стандартная задержка

ldi razr1, 255

ldi razr2, 255

ldi razr3, 10

Pdelay:

dec razr1

brne Pdelay

dec razr2

brne Pdelay

dec razr3

brne Pdelay

ret

**Выводы.** В результате выполнения лабораторной работы было усвоено принципы обмена информацией по интерфейсу 1 Wire и получены навыки использования интерфейса 1 Wire для получения информации от датчиков.