|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное автономное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**Отчет**

**по лабораторной работе № \_5\_**

**Название:** Обмен данными по интерфейсу SPI

**Дисциплина:** Микропроцессорные системы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-63Б |  |  | В.К. Залыгин Р.В. Дорохов |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Е.Ю. Гаврилова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2025

**Цель работы:**

- изучение структуры модуля SPI в микроконтроллере AVR,

- программирование передачи и приема данных по SPI.

**Задание 1. Передача данных в симплексном режиме в Proteus**

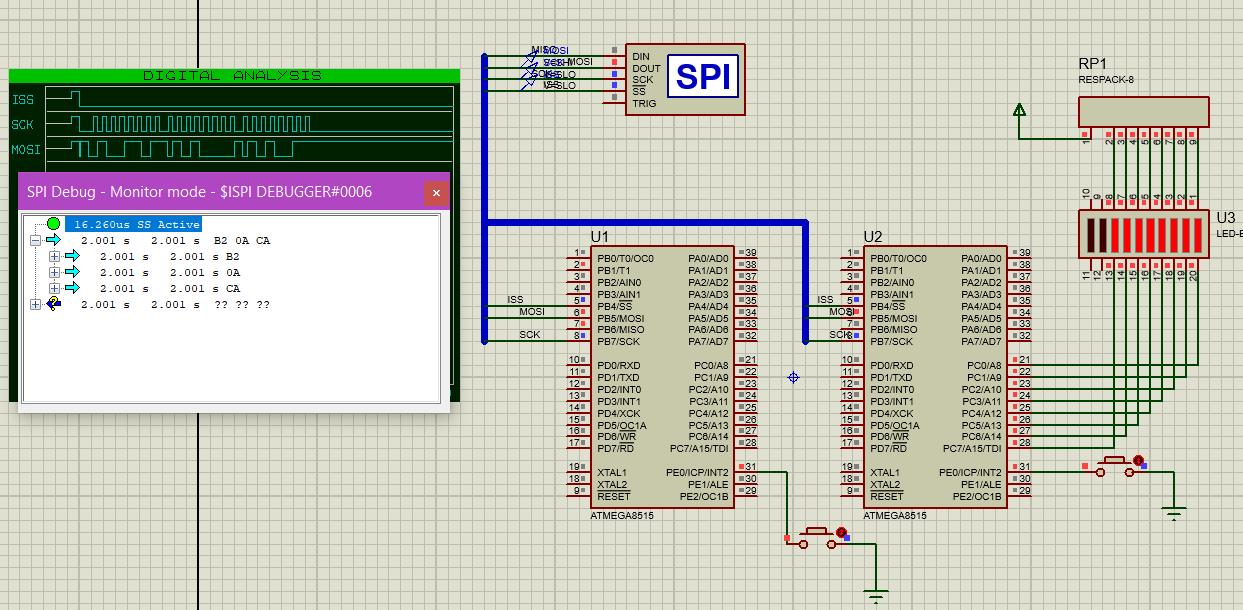


Рисунок 1 - Схема с открытым окном SPI Debugger

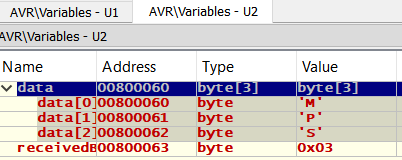




Рисунок 2 - Содержимое памяти микроконтроллера-приёмника с выделенными

байтами, которые были получены по UART

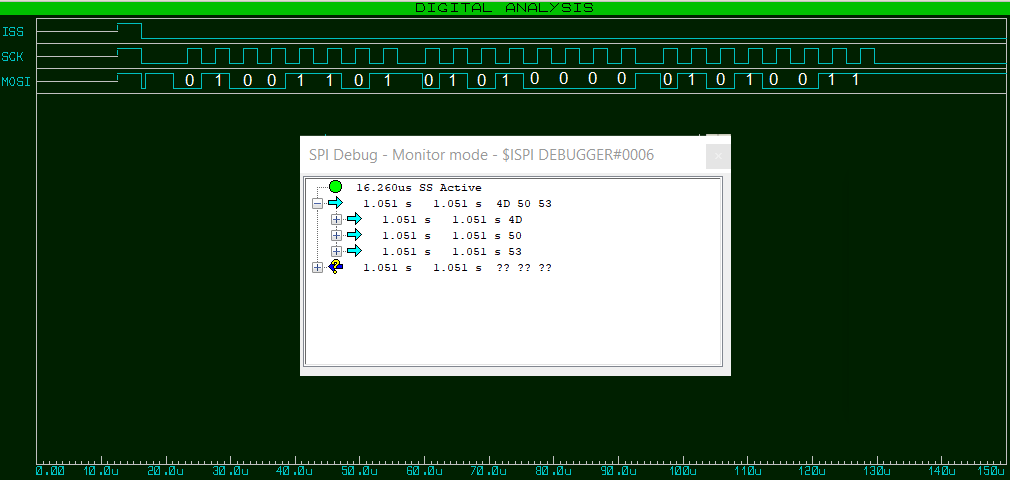


Рисунок 3 - Временная диаграмма с расшифровкой

**Задание 2. Обработка прерывания SPI**

Листинг 1 - Программа ведущего

|  |
| --- |
| #include <avr/io.h>  /\* Кнопка START - PE0 \*/  #define BUTTON\_START 0  #include <avr/interrupt.h>  /\* Выводы SPI - PB \*/  #define PIN\_SS 4  #define PIN\_SS1 1  #define PIN\_MOSI 5  #define PIN\_SCK 7  /\* Передаваемые данные \*/  #define DATA\_LENGTH 3  const unsigned char data[DATA\_LENGTH] = {'M', 'P', 'S'};  uint8\_t i = 0;  uint8\_t sending = 0;  ISR(SPI\_STC\_vect) {  PORTB |= (1<<PIN\_SS1);  if (i < DATA\_LENGTH) {  PORTB &= ~(1<<PIN\_SS1);  SPDR = data[i++];  }  if (i == DATA\_LENGTH) {  sending = 0;  }  } |

Листинг 1 - Программа ведущего

|  |
| --- |
| int main() {  /\* Инициализация SPI \*/  /\* Настройка выводов MOSI, SCK, SS на вывод \*/  DDRB = (1<<PIN\_MOSI)|(1<<PIN\_SCK)|(1<<PIN\_SS);  /\* Включение SPI в режиме ведущего, частота передачи f\_clk/16 \*/  SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0)|(1<<SPIE);  /\* Инициализация портов ввода-вывода \*/  /\* Настройка PIN\_SS1 на вывод \*/  DDRB |= (1<<PIN\_SS1);  PORTB |= (1<<PIN\_SS1);  /\* Настройка PE0 на ввод с подтягивающим резистором \*/  PORTE = (1<<BUTTON\_START);  sei();  /\* Бесконечный цикл \*/  while(1) {  /\* Проверка нажатия кнопки \*/  if (!(PINE & (1<<BUTTON\_START))) {  /\* Ожидание отпускания кнопки \*/  while (!(PINE & (1<<BUTTON\_START)))  ;  while (sending);  i = 0;  PORTB &= ~(1<<PIN\_SS1);  SPDR = data[i];  i++;  }  }  return 0;  } |

**Задание 3. Передача данных в симплексном режиме на макете**

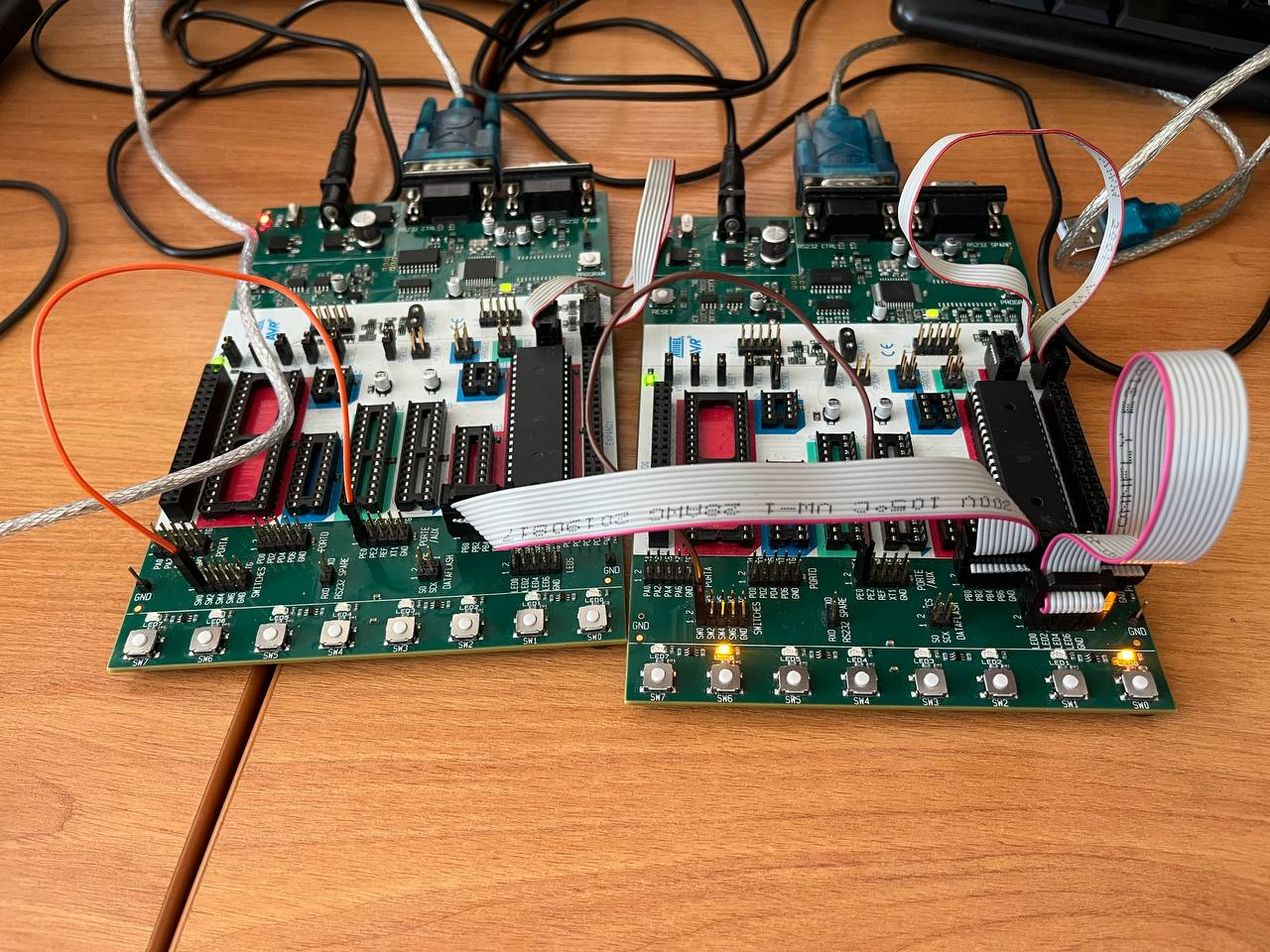


Рисунок 4 - Фотография макета с принятым байтом на светодиодах

**Задание 4. Передача данных в дуплексном режиме в Proteus**

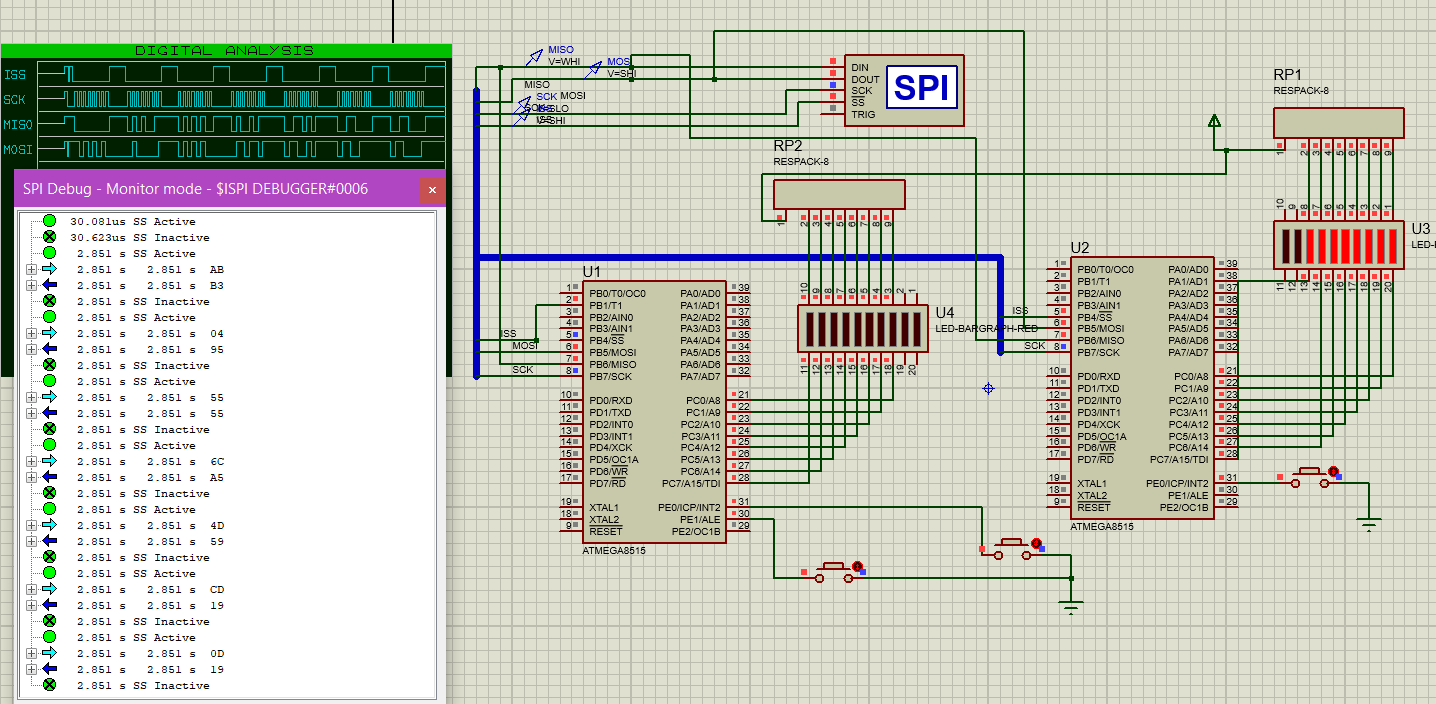
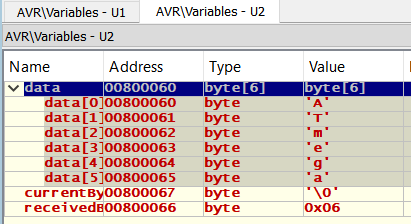
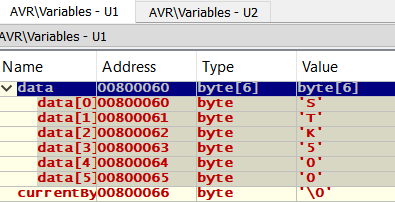
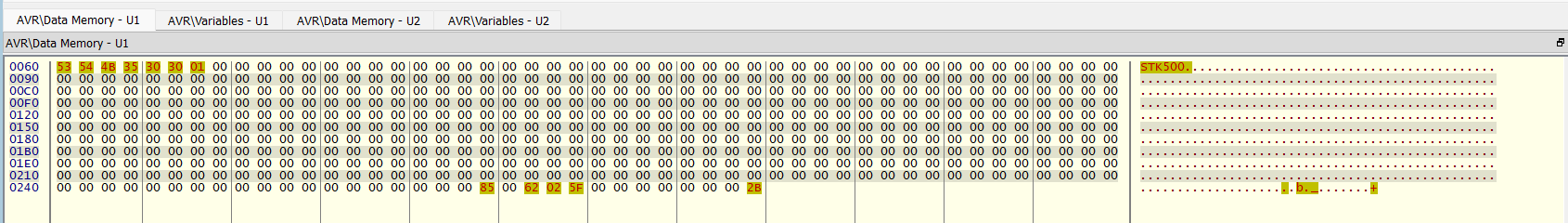


Рисунок 5 - Схема с открытым окном SPI Debugger





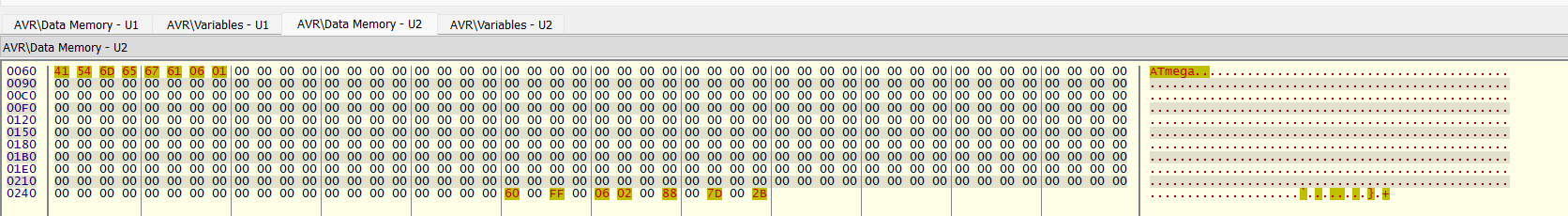


Рисунок 6 - Содержимое памяти обоих МК с выделенными байтами, которые

были получены по SPI

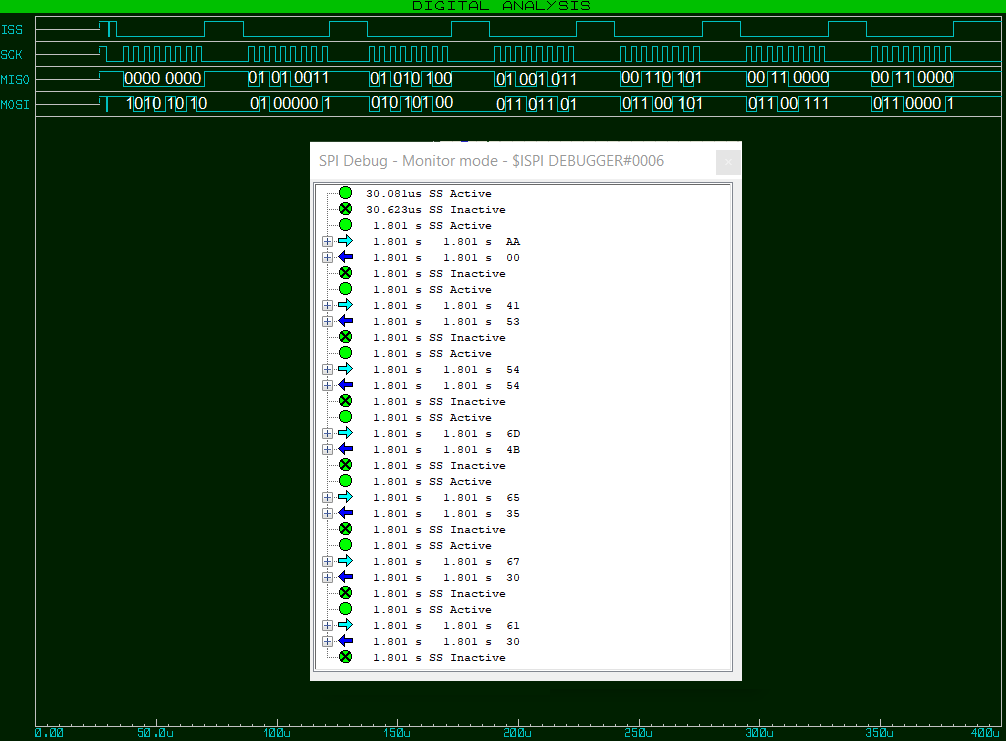


Рисунок 7 - Временная диаграмма с расшифровкой

**Задание 5. Передача данных в дуплексном режиме на макете**

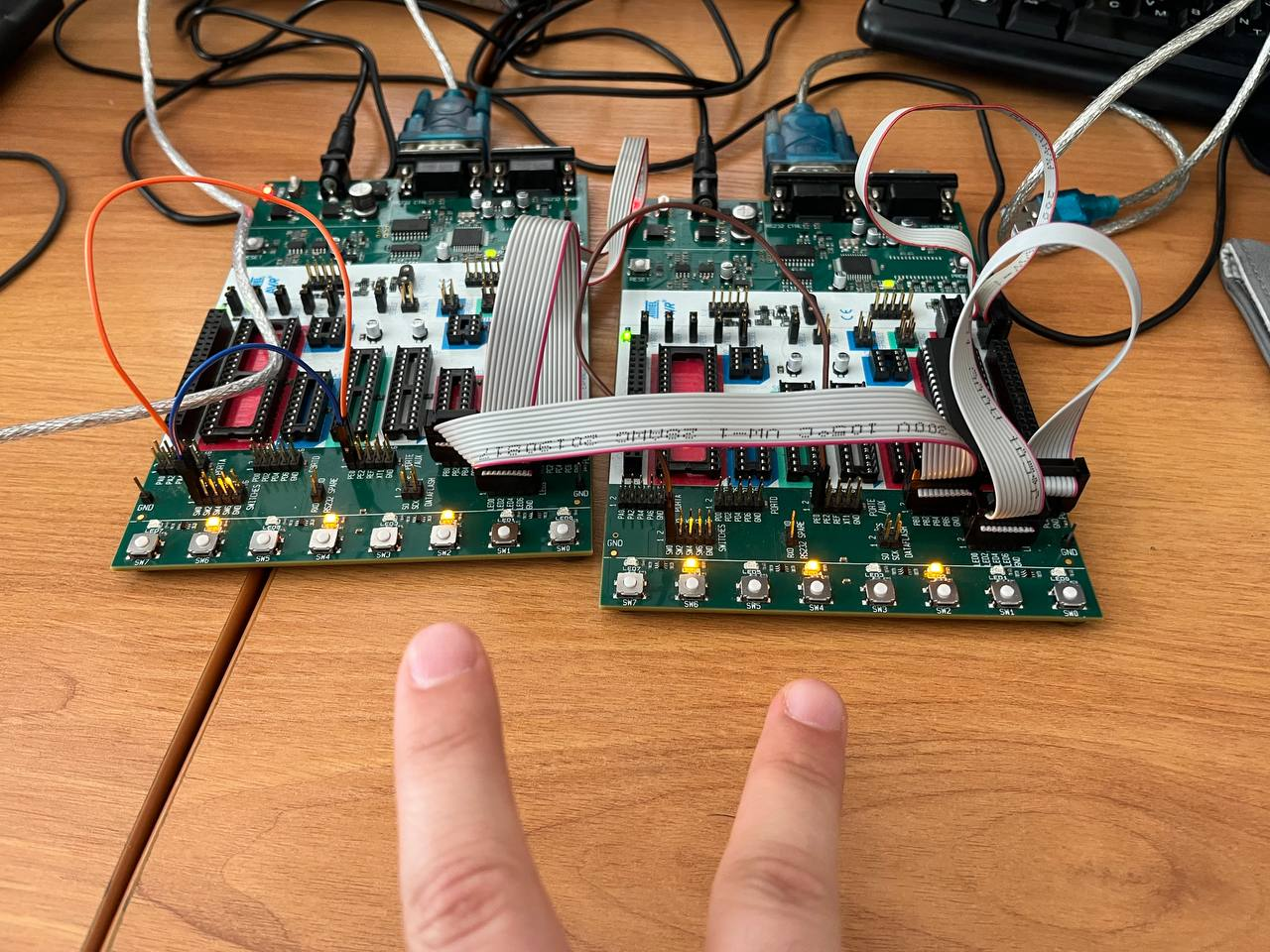


Рисунок 8 - Фотография макета с принятыми байтами на светодиодах

**Задание 6. Передача произвольного сообщения**

Листинг 2 - Программа ведущего

|  |
| --- |
| #include <avr/io.h>  #define F\_CPU 3690000  #include <util/delay.h>  /\* Кнопки управления - PE \*/  #define BUTTON\_START 0  #define BUTTON\_SHOW 1  #define PIN\_SS 4  #define PIN\_SS1 1  #define PIN\_MOSI 5  #define PIN\_SCK 7  /\* Массив данных \*/  #define DATA\_LENGTH 5  unsigned char data[DATA\_LENGTH] = {'H', 'e', 'l', 'l', 'o'};  uint8\_t currentByte = 0; /\* Выводимый байт \*/  /\* Байт для подачи сигнала о начале обмена \*/  #define START\_SYM 0xAA  char SpiSendReceive(char c) {  /\* Выбор ведомого \*/  PORTB &= ~(1<<PIN\_SS1);  /\* Отправка байта \*/  SPDR = c;  /\* Ожидание освобождения буфера \*/  while( !(SPSR & (1<<SPIF)) )  ;  /\* Завершение обмена с ведомым \*/  PORTB |= (1<<PIN\_SS1);  /\* Чтение байта \*/  return SPDR;  }  int main() {  /\* Инициализация SPI \*/  /\* Настройка выводов MOSI, SCK, SS на вывод \*/  DDRB = (1<<PIN\_MOSI)|(1<<PIN\_SCK)|(1<<PIN\_SS);  /\* Включение SPI в режиме ведущего, частота передачи f\_clk/16 \*/  SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0)|(1<<CPHA)|(1<<DORD);  /\* Инициализация портов ввода-вывода \*/  /\* Настройка PIN\_SS1 на вывод \*/  DDRB |= (1<<PIN\_SS1);  PORTB |= (1<<PIN\_SS1);  /\* Настройка PE0, PE1 на ввод с подтягивающим резистором \*/  PORTE = (1<<BUTTON\_START)|(1<<BUTTON\_SHOW);  /\* Настройка PC на вывод \*/  DDRC = 0xFF;  /\* Погасить светодиоды, подключенные к PC \*/  PORTC = 0xFF;  /\* Бесконечный цикл \*/  while(1) {  /\* Проверка нажатия кнопки START \*/  if (!(PINE & (1<<BUTTON\_START))) {  /\* Ожидание отпускания кнопки \*/  while (!(PINE & (1<<BUTTON\_START)));  /\* Отправка сигнала о начале обмена \*/  SpiSendReceive(START\_SYM);  /\* Задержка 10 мкс \*/  \_delay\_us(10);  /\* Цикл передачи данных \*/  uint8\_t i;  for (i = 0; i < DATA\_LENGTH; i++) {  /\* Отправка и прием i-го байта \*/  data[i] = SpiSendReceive(data[i]);  /\* Задержка 10 мкс \*/  \_delay\_us(10);  }  }  /\* Проверка нажатия кнопки SHOW \*/  if (!(PINE & (1<<BUTTON\_SHOW))) {  /\* Ожидание отпускания кнопки \*/  while (!(PINE & (1<<BUTTON\_SHOW)))  ;  /\* Вывод данных с инверсией битов для светодиодов \*/  PORTC = ~data[currentByte];  currentByte = (currentByte + 1) % DATA\_LENGTH;  }  }  return 0;  } |

Листинг 3 - Программа ведомого

|  |
| --- |
| #include <avr/io.h>  #include <avr/interrupt.h>  /\* Кнопка SHOW - PE0 \*/  #define BUTTON\_SHOW 0  /\* SPI MISO - PB6 \*/  #define PIN\_MISO 6  /\* Массив данных \*/  #define DATA\_LENGTH 5  unsigned char data[DATA\_LENGTH] = {'w', 'o', 'r', 'l', 'd'};  uint8\_t receivedBytes = 0; /\* Счетчик принятых байтов \*/  uint8\_t currentByte = 0; /\* Выводимый байт \*/  /\* Байт для подачи сигнала о начале обмена \*/  #define START\_SYM 0xAA  /\* Обработчик прерывания SPI\_STC \*/  ISR(SPI\_STC\_vect) {  unsigned char incomingByte = SPDR;  if (incomingByte == START\_SYM) {  /\* Загрузка 0-го байта для дальнейшей отправки \*/  SPDR = data[0];  }  else if (receivedBytes < DATA\_LENGTH) {  SPDR = data[receivedBytes + 1];  data[receivedBytes] = incomingByte;  receivedBytes++;  /\* Включить светодиоды по завершении приема \*/  if (receivedBytes == DATA\_LENGTH) {  PORTC = 0x00;  }  }  }  int main() {  /\* Инициализация SPI \*/  /\* Включение SPI в режиме ведомого \*/  SPCR = (1<<SPE)|(1<<SPIE)|(1<<CPHA)|(1<<DORD);  /\* Инициализация портов ввода-вывода \*/  /\* Настройка MISO на вывод \*/  DDRB = (1<<PIN\_MISO);  /\* Настройка PE0 на ввод с подтягивающим резистором \*/  PORTE = (1<<BUTTON\_SHOW);  /\* Настройка PC на вывод \*/  DDRC = 0xFF;  /\* Погасить светодиоды, подключенные к PC \*/  PORTC = 0xFF;  /\* Глобальное разрешение прерываний \*/  sei();  /\* Вывод полученных данных на светодиоды, байт за байтом \*/  /\* Бесконечный цикл \*/  while(1) {  /\* Проверка нажатия кнопки \*/  if (!(PINE & (1<<BUTTON\_SHOW))) {  /\* Ожидание отпускания кнопки \*/  while (!(PINE & (1<<BUTTON\_SHOW)))  ;  /\* Вывод данных с инверсией битов для светодиодов \*/  PORTC = ~data[currentByte];  currentByte = (currentByte + 1) % DATA\_LENGTH;  }  }  return 0;  } |

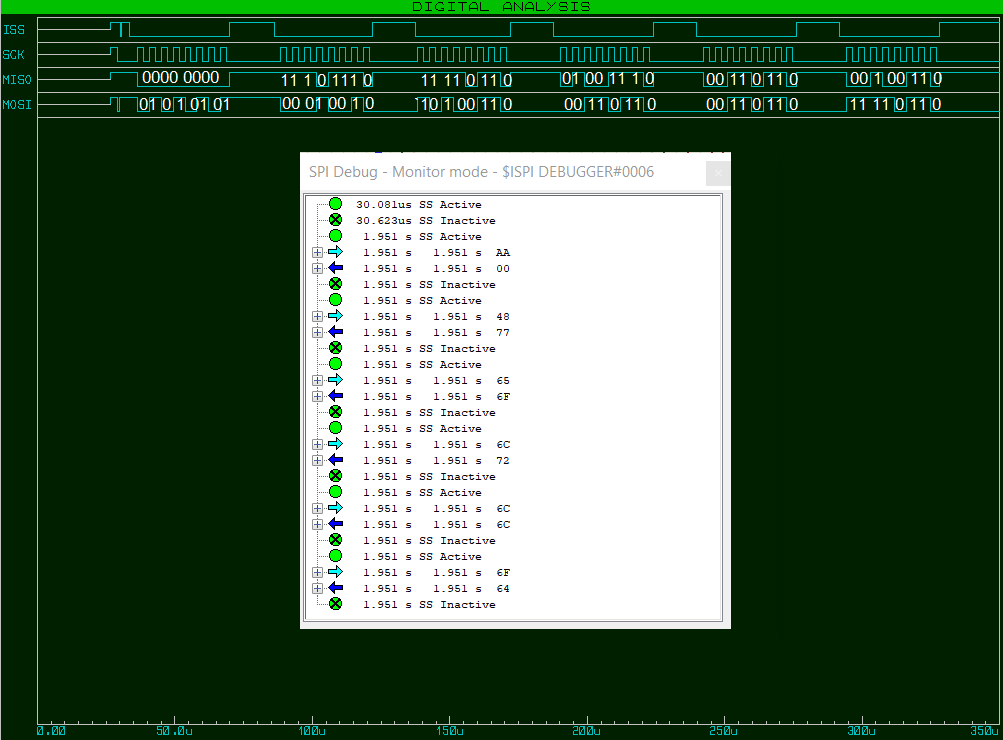


Рисунок 9 - Временная диаграмма с расшифровкой

**Вывод**

Изучил структуры модуля SPI в микроконтроллере AVR, а так же программировал передачу и прием данных по протоколу SPI.