

Страницы / [ММК] НИОКРы / Архитектурное решение

# Чтение данных из контроллера

Создатель Бяков Сергей, отредактировано мар 04, 2020

- 1 [Обзор](#)
- 2 [Описания стенда для разработки](#)
- 3 [Java библиотеки для работы с контроллерами:](#)
  - 3.1 [s7connector](#)
  - 3.2 [snap7](#)
- 4 [Утилита проверки работы с контроллерами](#)
  - 4.1 [Требования к утилите](#)

## Обзор

На предприятии используются контроллеры:

Siemens S7 400 (<https://mall.industry.siemens.com/mall/ru/ru/Catalog/Products/10036892>)

Siemens S7 300 (<https://mall.industry.siemens.com/mall/ru/ru/Catalog/Products/5000014?tree=CatalogTree>)

В нашем решении по вибродиагностике, мы планируем использовать:

Siemens S7 1200 (<https://new.siemens.com/ru/ru/produkty/avtomatizacia/sistemy-avtomatizacii/promyshlennye-sistemy-simatic/kontroller-simatic/s7-1200.html>)

Данные контроллеры обмениваются данными по протоколам Profibus (<https://ru.wikipedia.org/wiki/Profibus>) и Profinet (<https://en.wikipedia.org/wiki/PROFINET>)

## Описания стенда для разработки

Для тестирования и разработки установлен контроллер Siemens S7 1200

Сетевые настройки контроллера:

IP 172.30.143.30

mask: 255.255.255.0

Gw: 172.30.143.1

Данные хранятся по адресу:

Rack 0

Slot 0

Datablock 3

1ый и 2ой байты

В нем организована следующая логика работы:

В блоке данных с адресом DB3, в ячейку памяти с адресом 0, записывается последовательность от 0 до 10 с периодичностью в одну секунду, что эмулирует поступление данных с датчика №1

В блоке данных с адресом DB3, в ячейку памяти с адресом 1 (или 0 со сдвигом 1 байт) записывается последовательность от 0 до 10 с периодичностью в одну секунду, что эмулирует поступление данных с датчика №2

Таким образом мы эмулируем поступление данных с двух датчиков.

При проведении работ по разработке, запрещено перепрограммировать контроллер, в виду отсутствия у нас компетенции по программированию контроллеров.

## Java библиотеки для работы с контроллерами:

В результате анализа вариантов реализации, найдено две библиотеки позволяющие работать с контроллером на прямую, для гибкости решения и отсутствия зависимости от инфраструктуры заказчика. Обе библиотеки протестированы для опроса тестового контроллера.

### s7connector

<https://github.com/s7connector/s7connector>

Тестовый пример:

```
package ru.datana.siemensopc;
import com.github.s7connector.api.DaveArea;
import com.github.s7connector.api.S7Connector;
import com.github.s7connector.api.factory.S7ConnectorFactory;
import com.github.s7connector.exception.S7Exception;
import java.util.Arrays;
/**
 * Echo connector for testing
 *
 * returns the same buffer in read() as given in write() regardless of the byte-range
 *
 * @author Thomas Rudin
 */
public class Test0PC {

    public static void main(String[] args) {
        try {
            S7Connector connector =
                S7ConnectorFactory
                    .buildTCPConnector()
                    .withHost("172.30.143.30")
                    .withRack(0) //optional
                    .withSlot(0) //optional
                    .build();
            //Read from DB100 10 bytes
            byte[] bs = connector.read(DaveArea.DB, 3, 1, 0);

            System.out.println(Arrays.toString(bs));
        } catch (S7Exception e) {
            System.out.println(e.getCause());
        }
    }
}
```

### snap7

Это набор библиотек для разных языков программирования. На первый взгляд более продвинутая.

Есть реализация для Java (Moka7)

<http://snap7.sourceforge.net/>

Можно скачать скомпилированные под разные платформы примеры, например:

System Info | Data read/write | Multi read/write | Directory | Block - Up/Download | Block - DB Get/Fill | Read SZL | Date/Time | Control | Security

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0000	\$09	\$09	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0010	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0020	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0030	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0040	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0050	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0060	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0070	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0080	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0090	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
00A0	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
00B0	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
00C0	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
00D0	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
00E0	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
00F0	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0100	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0110	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0120	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0130	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0140	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0150	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0160	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0170	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0180	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
0190	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00
01A0	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00	\$00

Read Data | 8 ms | OK

В двух ячейках памяти мы видим как раз наши тестовые данные.

## Утилита проверки работы с контроллерами

Необходимо написать утилиту на java, которая будет проверять возможность работы с контроллерами расположенными в разных сегментах сети заказчика и контроллерами разных типов. Перед проектированием нашей системы, с помощью нее так же будем проверять тип данных на контроллере и возможность чтения интересующих нас данных.

### Требования к утилите

Необходимо релизовать утилиту на java в которой на вход будут передаваться следующие параметры:

- 1 ip адрес контроллера
- 2 rack
- 3 slot
- 4 адрес дата блока

5 тип данных которые мы хотим получить (byte, bit, word, dword, real)

6 сдвиг относительно начала блока данных (видимо надо тип сдвига тоже указать в тех же типах что данные)

7 частота чтения с контроллера в миллисекундах (1000 - 1 сек)

8 количество чтений, сколько раз нужно обратиться к контроллеру прежде чем завершить программу, при этом если были ошибки при передаче данных по сети, чтения с контроллера или иные, то это обращение не учитывается. т.е. учитываются только успешные чтения данных.

утилита должна обрабатывать ошибки контроллера, сети и прочие исключения и выводить полученные данные, с указанием времени в ms между запросом и получением ответа  
формат вывода

[ значение полученное с контроллера ] - кол-во миллисекунд на запрос

минимальная частота обращения к контроллеру 0.5сек

---