## Лабораторні роботи 1-2. Швидке розроблення пристроїв IoT в середовищі віртуального моделювання Proteus

**Мета роботи:** ознайомлення з віртуальними моделями пристроїв IoT в програмному середовищі Proteus, отримання практичних навиків швидкого налагодження програм для AVR-мікроконтролерів в середовищі Proteus.

### Програма роботи

1. Навчитися аналізувати та проектувати пристрої, використовуючи віртуальні мікроконтролери Arduino, засоби налагодження програм.
2. Навчитися використовувати готові віртуальні пристрої (проекти) на основі плат Arduino в середовищі Proteus.
3. Навчитися виконувати конфігурацію, проводити моделювання розробленої схеми, використовуючи програми користувача.
4. Дослідити віртуальне середовище моделювання Proteus**.**
5. Підготувати звіт про виконану роботу.

### Теоретичні відомості

Розробником пакету Proteus є фірма Labcenter Electronics Великобританія. Сайт розробника - http://www.labcenter.co.uk/. Відмінність від аналогічних за призначенням пакетів програм, наприклад, Electronics Workbench, Multisim, MicroCap, Tina і т.п. в розвиненій системі симуляції (інтерактивного налагодження в режимі реального часу і покроково) для різних сімейств мікроконтролерів: 8051, PIC (Microchip), AVR (Atmel), і ін. Proteus має великі бібліотеки компонентів, в тому числі і периферійних пристроїв: світлодіодні і РК індикатори, температурні датчики, годинник реального часу - RTC, інтерактивних елементів введення-виведення: кнопок, перемикачів, віртуальних портів і віртуальних вимірювальних приладів, інтерактивних графіків, які не завжди присутні в інших подібних програмах.

Proteus VSM складається з двох самостійних програм: ISIS і ARES. ARES - це трасування друкованих плат. Основною програмою є ISIS, в ній передбачений гарячий зв'язок з ARES для розведення плати.

Спрощено, робота в середовищі моделювання Proteus ISIS складається з наступних пунктів:

1. Вибір електричної принципової схеми і задання їх параметрів функціонування з набору прикладів.
2. Розміщення віртуальних приладів там, де це необхідно, вибір режимів їх роботи.
3. Симуляція чи проведення спеціалізованого аналізу.
4. Виконання налагодження програм мікроконтролерів.

## Введення та дослідження навчальних прикладів засобами середовища моделювання Proteus

Для того що б отримати уявлення про те, на що здатний Proteus, як середовище моделювання, слід відкрити деякі файли прикладів проектів. У VSM Studio IDE була реалізована підтримка наборів інструментів Arduino AVR. Це дозволяє розробляти прототипи проектів Arduino безпосередньо всередині програми Proteus, нова функція проекту дуже корисна тут, так як проект можна легко створити для різних Arduino, але про це поговоримо пізніше.

**Приклад**: вимірювання температури за допомогою датчика DS18B20.

Подивитися на процеси, що відбуваються при вимірюванні температури можна, відкривши файл з набору прикладів: File-Open Sample Project - VSM for AVR - Arduino - Arduino DS18B20 OneWire (рисунку 1.1).

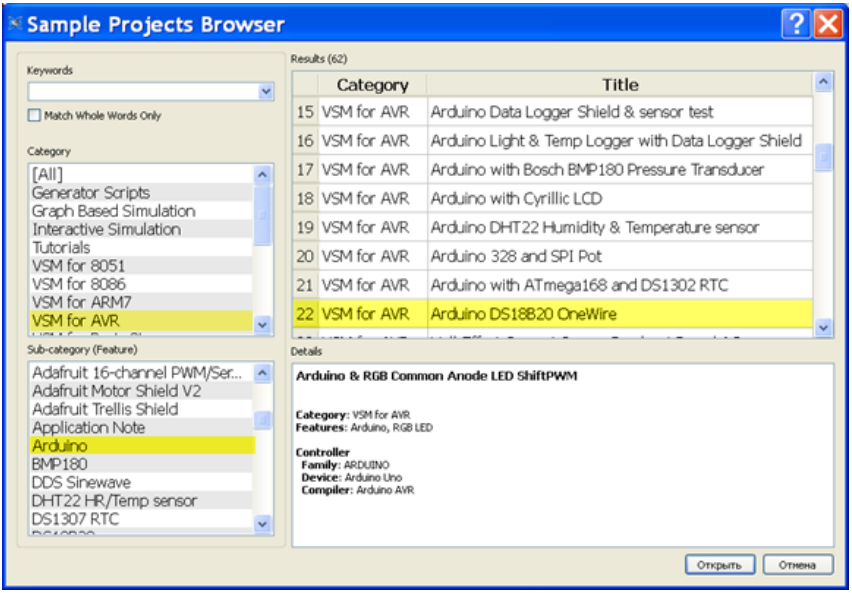


Рисунок 1.1 - Вибір проекту в середовищі моделювання Proteus

На рисунку 1.2. наведена схема моделювання цифрового термометра на мікроконтролері ATmega328 з цифровим датчиком температури DS18B20 фірми Dallas Semiconductor і виведенням інформації на семисегментний індикатор. Мікросхема DS18B20 забезпечує температурні вимірювання за шкалою Цельсія. Мікросхема DS18B20 підключається через 1-дротову шину, яка за визначенням вимагає тільки однієї лінії даних для взаємодії з мікроконтролером. Вона має робочий температурний діапазон від -55 ° C до + 125 ° C і точність ± 0.5 ° C в діапазоні від -10 ° C до + 85 ° C. Модель термометра DS18B20 дозволяє задавати температуру термодатчика.

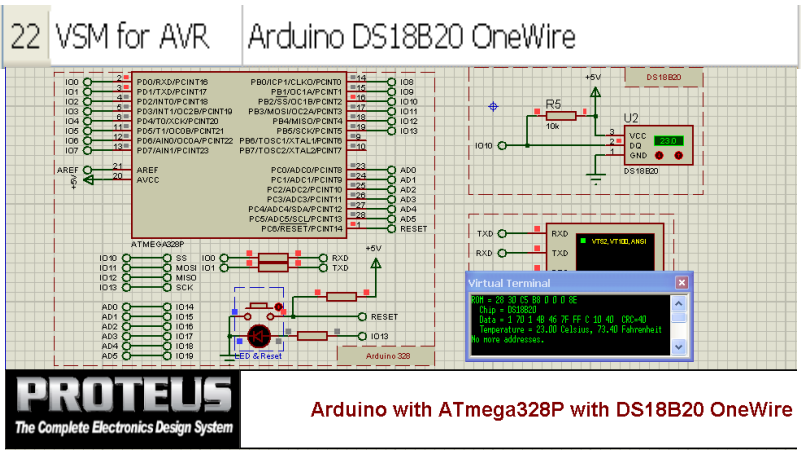


Рисунок 1.2 – Схема моделювання зміни температури з термодатчиком

За допомогою вбудованого в середовище моделювання Proteus осцилографа можна отримати осцилограму інформаційного обміну мікроконтролера з датчиком. Тимчасова діаграма сигналів (протокол 1-Wire) представлена на рисунку. 1.3.

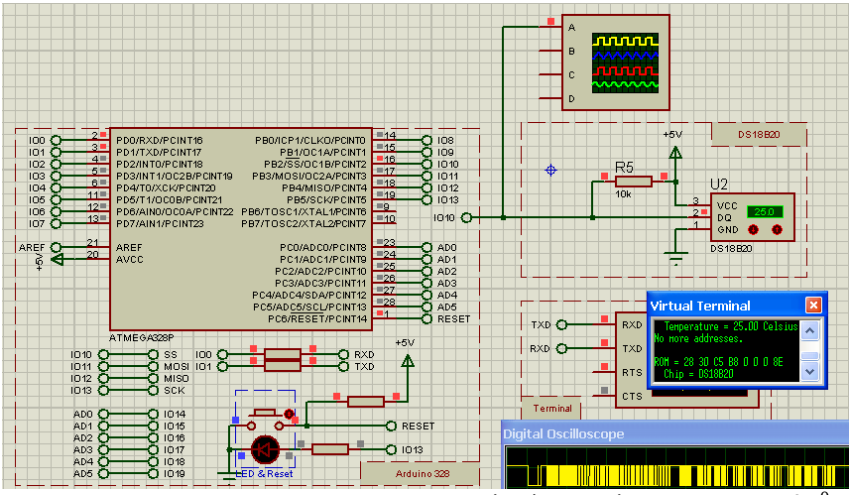


Рисунок 1.3 – Сигнал на однопровідній шині датчика при $25^0$C

Рівні напруги можна визначити за кольором міток на всіх виводах компонентів (лог.1 - червоний колір, лог.0 - синій, непідключений вивід - сірий колір).

### Налаштування мікроконтролера

Панель редагування властивостей мікроконтролера (рис. 1.4) відкривається подвійним клацанням по ньому лівою кнопкою миші або через праву кнопку і опцію Edit Properties (можна, виділивши його, натиснути на клавіатурі Ctrl + E).

В полі "Program File" потрібно вибрати файл з розширенням "elf" (для налагодження) або "hex", які створюються при компіляції програми в IDE Arduino, WinAVR і ін..

".hex" - файл "прошивки", яку завантажують і в реальний мікроконтролер. При виборі його можна налагоджувати пристрій без можливості перегляду виконуваного коду.

Щоб використовувати файл програми з розширенням ".hex" потрібно клацнути по мікроконтролеру правою кнопкою миші, вибрати пункт «правка властивостей», далі в цьому пункті вибрати «Program file» і ввести шлях до файлу, або просто клацнути по значку «відкрити папку», і вказати потрібний файл. Після цього ніяких додаткових дій робити не потрібно, крім натискання кнопки «ОК».

Clock Frequency - однозначно визначає будь-яку частоту тактування мікроконтролера при симуляції, яка відповідає частоті мікроконтролера в проекті. За умовчанням, встановлена частота синхронізації - 8 МГц. В Arduino Uno, Nano частота синхронізації - 16 МГц. У деяких Arduino Mini - 8 МГц.У середовищі програмування IDE Arduino за замовчуванням вибирається частота 16 МГц, але є можливість встановлення частоти 8 МГц. Для зміни тактової частоти в списку CKSEL Fuses потрібно вибрати Ext. Clock і в Advanced Properties замість (Default) вводимо потрібну частоту в герцах.

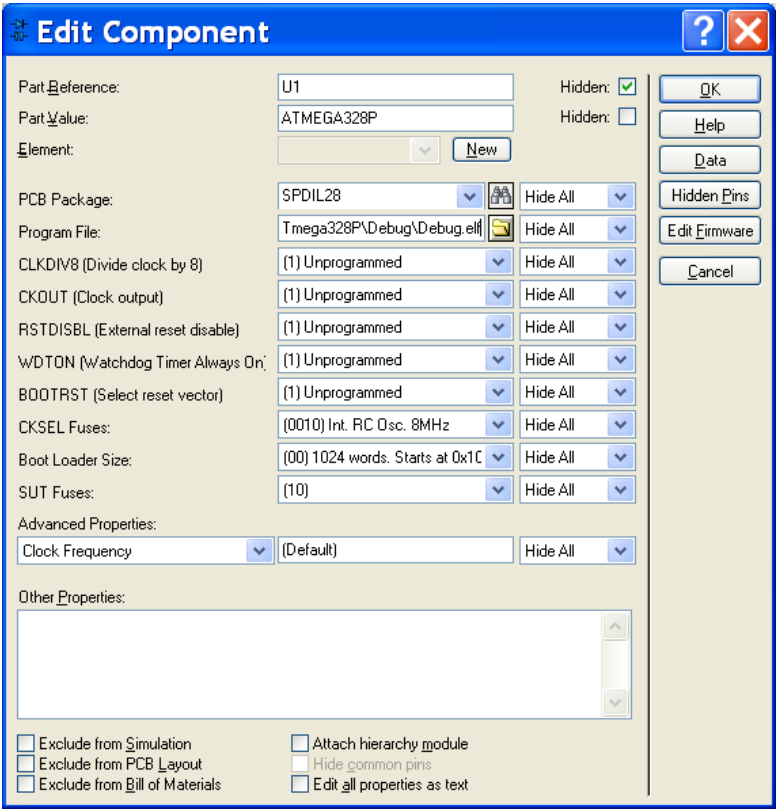


Рисунок 1.4 – Вікно налаштування властивостей мікроконтролера

Після цих установок проект із прикладів Proteus функціонує відповідно до заданої програми. Це дає можливість використовувати приклади проектів в Proteus demo для налагодження власних програм.

Proteus може використовувати компілятори, які встановлені в System - Compilers, зокрема:

- Arduino AVR - Yes - <Path IDE Arduino>;
- WinAVR - Yes - <Path WinAVR>.

Аналіз і редагування вихідного коду моделі мікроконтролера, використання віртуальних інструментів виконується відповідно до опису та додатків:

- Додаток 1. Вихідний текст програми і "hex" – файл.
- Додаток 2. Використання віртуального осцилографа для аналізу послідовностей сигналів.
- Додаток 3. Аналіз елементів моделі мікроконтролера.

**Задача 1 .**Ознайомитися з прикладами проектів працюючих схем в програмному середовищі Proteus для Arduino

- Arduino 4 Channel Relay;
- Arduino Cyrillic LCD;
- Arduino Motor Shield.

**Задача** 2. Описати процеси, які відбуваються при моделюванні схем– прикладів. .

**Зміст звіту**

1. Короткий опис змісту виконання роботи.
2. Опис прикладів симуляції в Proteus:
   - - принцип роботи;
   - - приклади осцилограм;
   - - дані обміну в терміналі;
   - - оцінку використуваних ресурсів (пам'яті програм і даних, виводів).
3. Висновки по роботі

**Контрольні питання**

1. Який порядок створення проекту в Proteus?
2. Як відкрити приклад для AVR МК?
3. Як визначити розмір програми в пам'яті МК?
4. Що таке .hex файл? Як його завантажити в МК?
5. Як вибрати віртуальний інструмент?
6. Для чого потрібний осцилограф?
7. Які основні налаштування віртуального осцилографа?
8. Які основні налаштування віртуального терміналу?
9. Як визначити часові інтервали параметрів сигналів в Proteus.
10. Як зняти значення з віртуального виходу порта?

## Література

1. Симулятор – отладчик PROTEUS VSM. ISIS.- Labcenter Electronics Co. - 26с.- – Proteus_ISIS_russian_manual.pdf. [Електронний ресурс]. – Режим доступу: http://labkit.ru/userfiles/file/education/Proteus/ Proteus_ISIS_russian_manual.pdf.

2. PROTEUS по-русски. Радио - ежегодник, выпуск 24. RYB_2013_24_Proteus.pdf. 2013. [Електронний ресурс]. – Режим доступу: https://www.rlocman.ru/book/book.html?di=148418 [Доступ 25.06.2019г ].

3. А.В. Пархоменко, О. М. Гладкова, Я. І. Залюбовський, А.В. Пархоменко, Інженерія вбудованих систем: Навчальний посібник. Запоріжжя: Дике Поле, 2017.

## Додаток 1. Вихідний текст програми і “hex”- файл

Приклади Proteus, як правило, містять вихідні тексти програм для мікроконтролерів в спеціальній закладці Source code, з якої вихідний код можна скопіювати і використовувати для вивчення і побудови власних програм. При конфігурації Proteus підключаються різні середовища розробки програм з відповідними компіляторами.

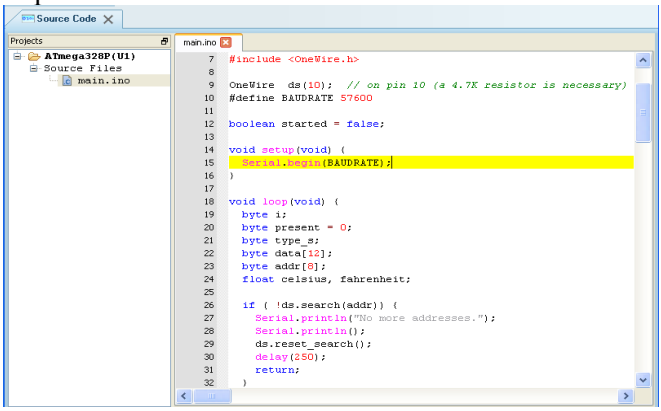


Рисунок 1.5. - Вигляд вікна з вихідним текстом програми

Результатом компіляції тексту програми є “hex” - файл. Наведений нижче фрагмент файлу містить коди команд, що розміщуються за адресами $ 0000 ... $ 00AF Flash - пам'яті програм мікроконтролера (порівняти з вмістом вікна налаштувань вікна пам'яті програм).

:10000000**0C94D8000C9400010C9400010C94000**
**1**95
:10001000**0C9400010C9400010C9400010C94000**
**1**5C
:10002000**0C9400010C9400010C9400010C94000**
**1**4C
:10003000**0C9400010C9400010C9400010C94000**
**1**3C
:10004000**0C94CA040C9400010C9498040C94720**
**4**4F
:10005000**0C9400010C9400010C9400010C94000**
**1**1C
:10006000**0C9400010C940001005EBCE2613FDD**
**83**52
:10007000**C29C7E20A3FD1F419DC3217FFCA2**
**401E**88
:100080005**F01E3BD3E6082DC237D9FC1421CF**
**EA0**78
:10009000**E1BF5D0380DE3C62BEE0025CDF816**
**33D**68
:1000A0007**C22C09E1D43A1FF4618FAA427799**
**BC5**58

Після налагодження програми “hex” - файл програми за допомогою програматора завантажується в пам'ять програм мікроконтролера пристрою

**Додаток 2. Використання віртуального осцилографа для аналізу послідовності сигналів**

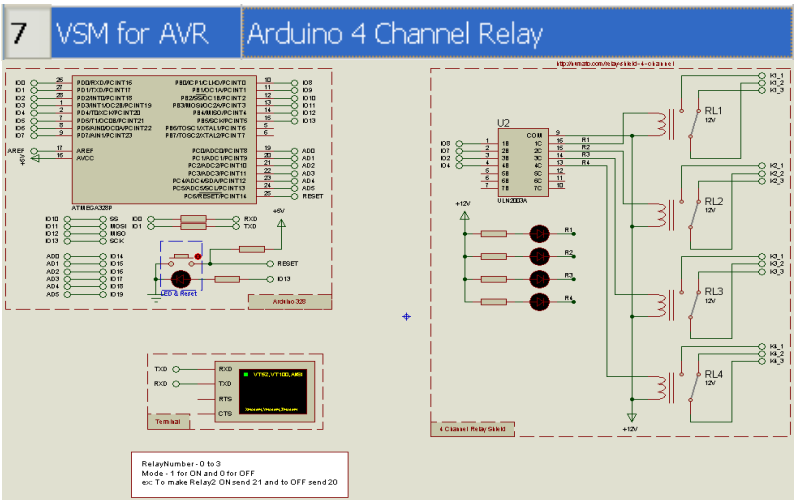


Рисунок 1.6 – Arduino 4 Channel Relay4.

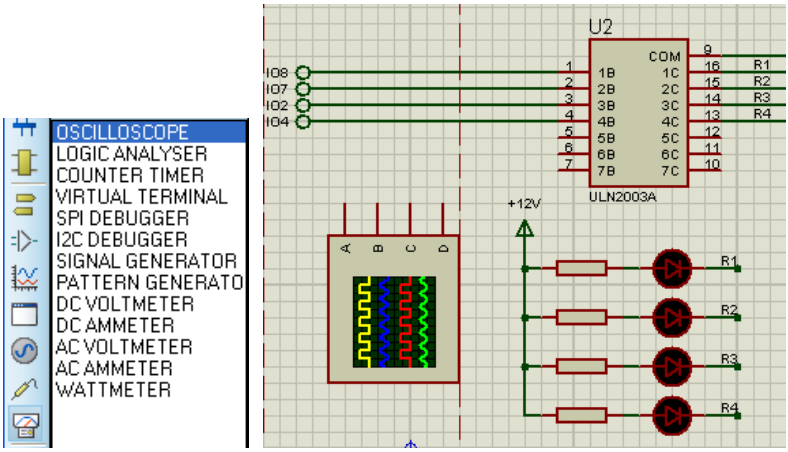


Рисунок 1.7 – Винесення терміналів сигналів IO8,7,2,4 і додавання 4- канального віртуального осцилографа

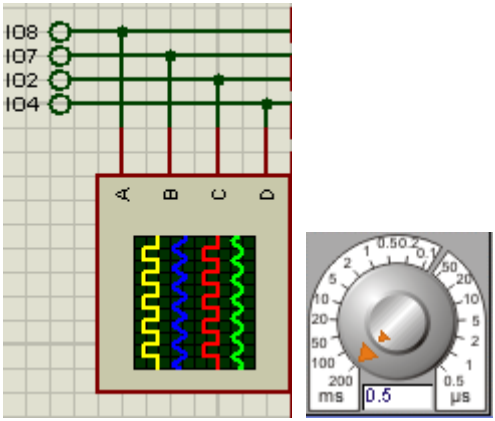


Рисунок 1.8 – Встановлення терміналів і підключення до каналів A,B,C,D із часовою шкалою 0.5 мс

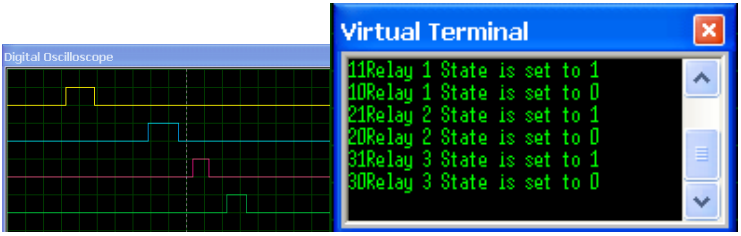


Рисунок 1.9 – Спостереження реакції МК (IO8,7,2,4) на команди, що поступають по послідовному каналу

**Додаток 3. Аналіз елементів моделі МК**

При налагоджені програм Proteus надає доступ до внутрішніх елементів моделі мікроконтролера при виборі Debug - AVR (рисунок 1.10):

- CPU Registers (регістри загального призначення, рисунок 1.11);
- Data Memory (оперативна пам'ять даних - ОЗУ (SRAM), рисунок 1.12) ;
- EPROM Memory (енергонезалежна пам'ять даних, рисунок 1.13);
- Program memory (Flash пам'яті програм, рисунок 1.14);
- IO Registers (регістри введення-виведення, рисунок 1.15).

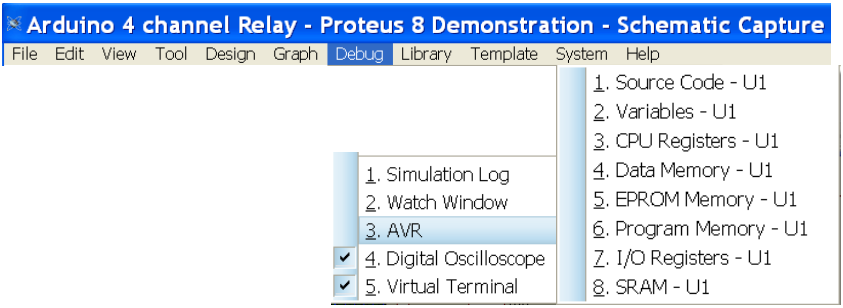


Рисунок 1.10 – Вибір елементів моделі МК

Робочі регістри (регістри загального призначення) R0…31 утворюють область надоперативної пам'яті. Регістри беруть безпосередню участь при виконанні машинних команд. Використовуються при програмуванні на мові низького рівня - Асемблер.

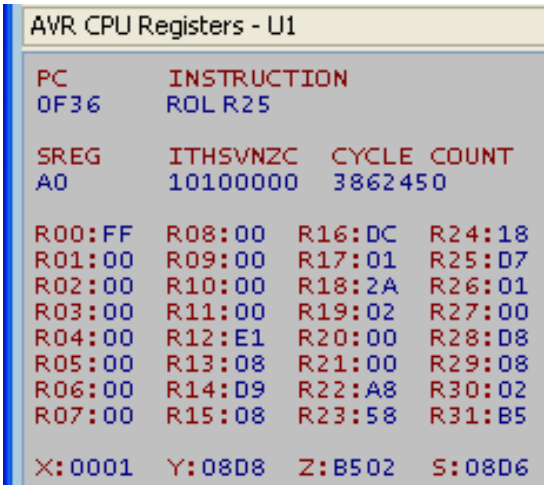


Рисунок 1.11 – Стан 32 робочих регістрів ядра МК (CPU Registers)

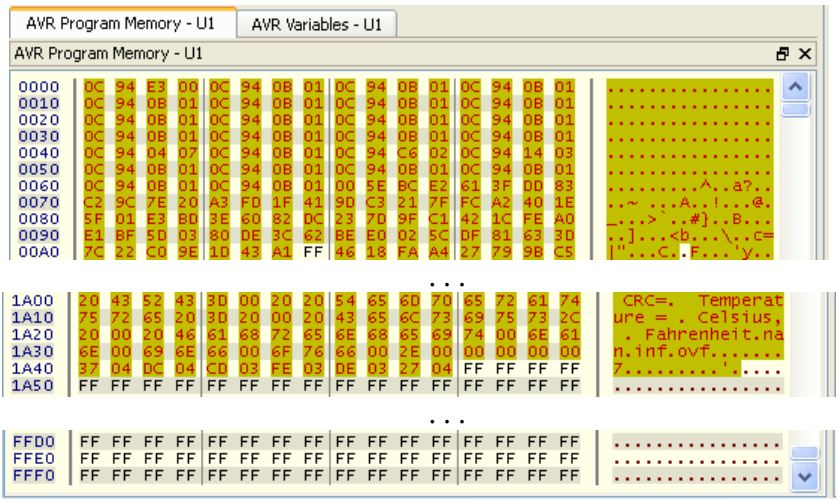


Рисунок 1.12 - Вміст 64К=65536 байт ( 32768 слів) Flash пам’яті програм.

Коди команд програми займають 6732 байт в діапазоні адрес $0000 … $1A4B (0 .. 6731).

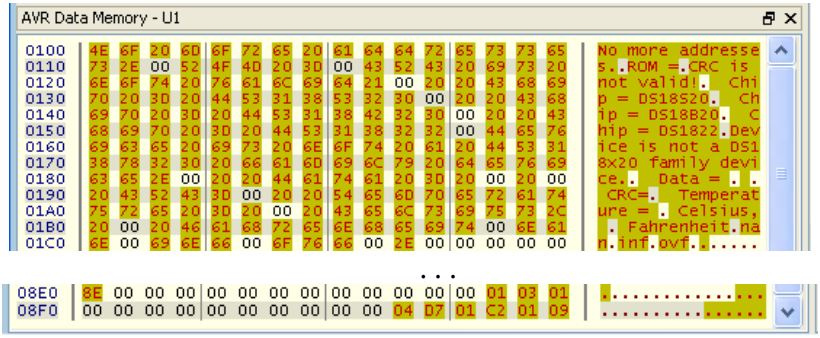


Рисунок 1.13 – Масив 2048 байт (2 Кб) комірок ОЗП (SRAM)