AVR®

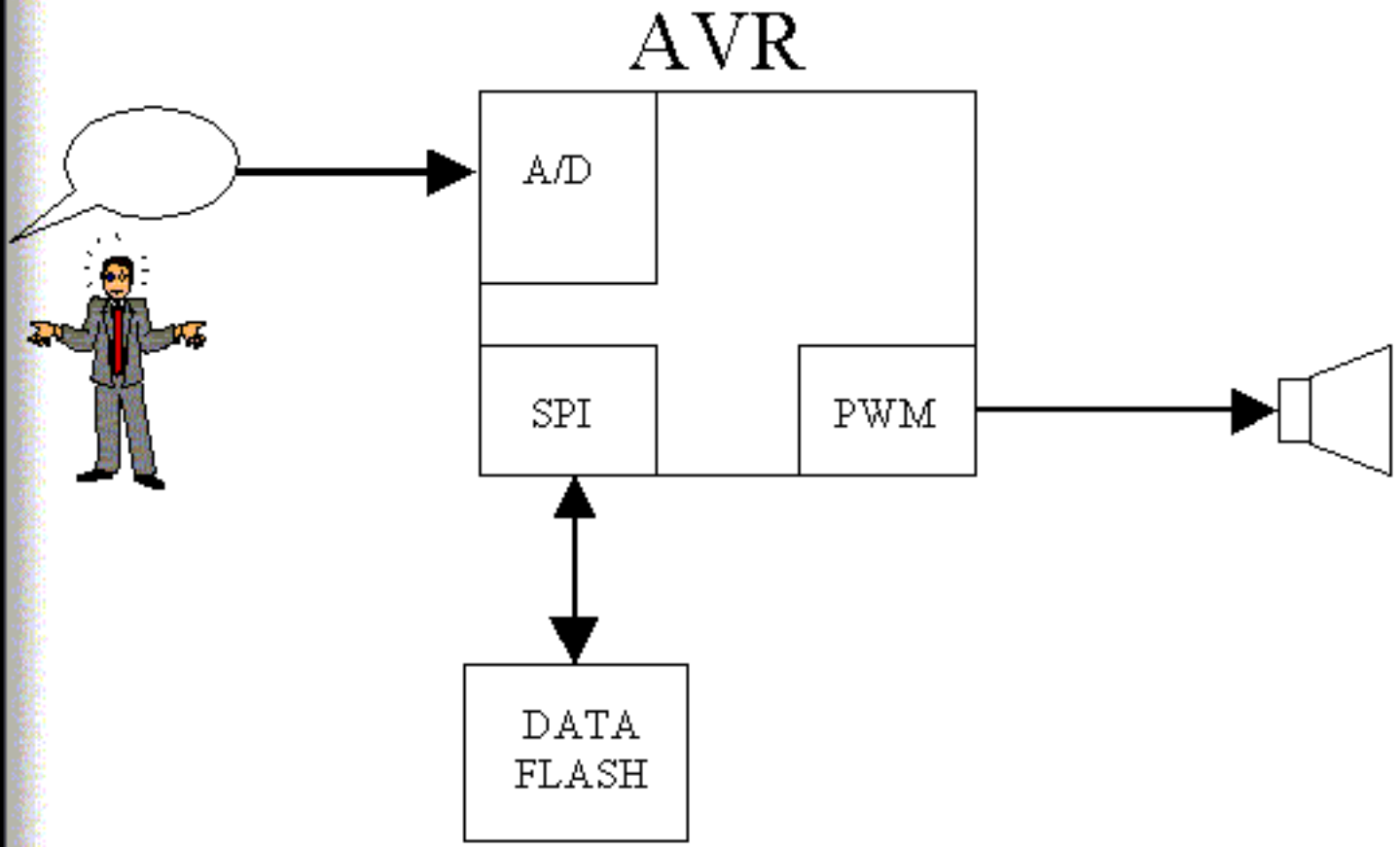
Европейский технический семинар

Марокко, май, 2000 г.

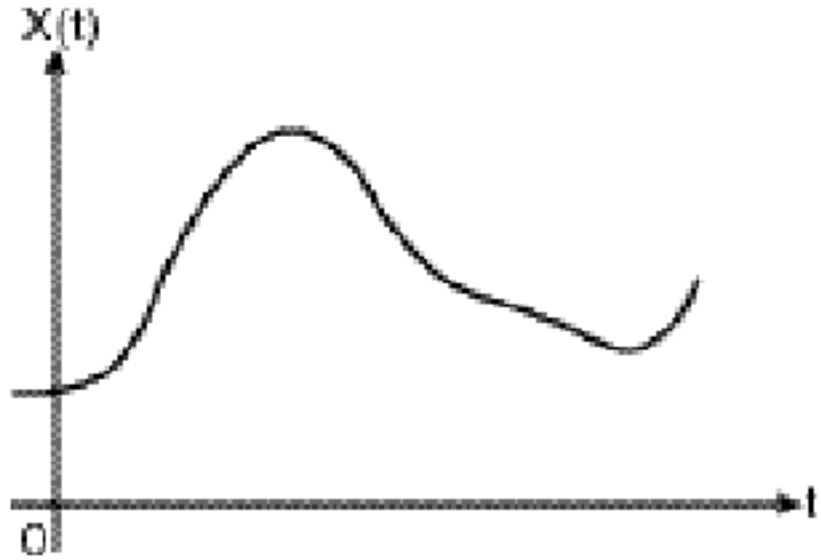
Часть 2

Пример использования AVR

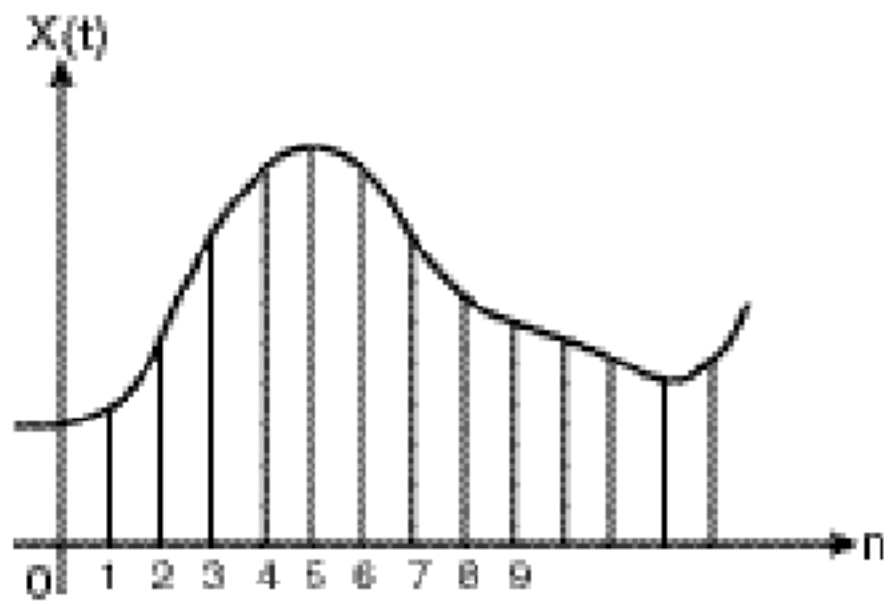
Цифровой диктофон



Оцифровка входного сигнала

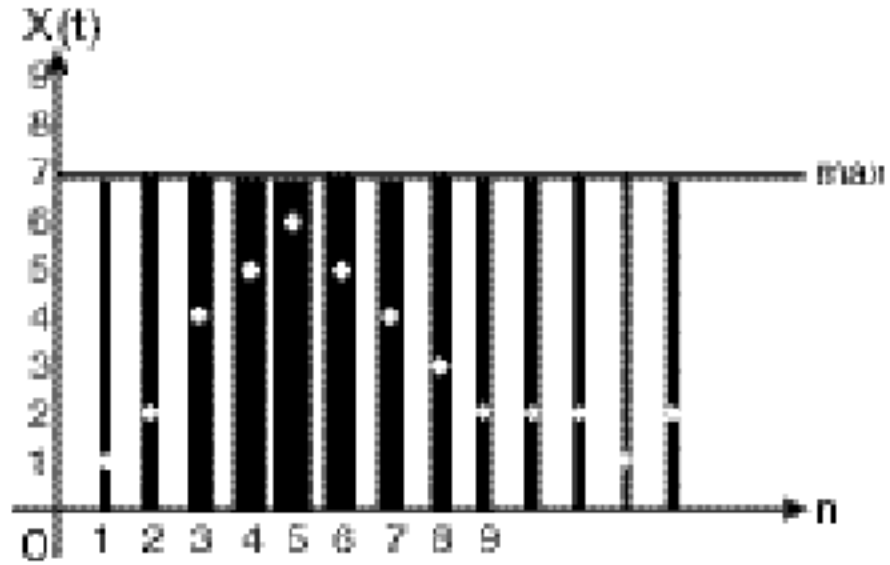


Аналоговый сигнал

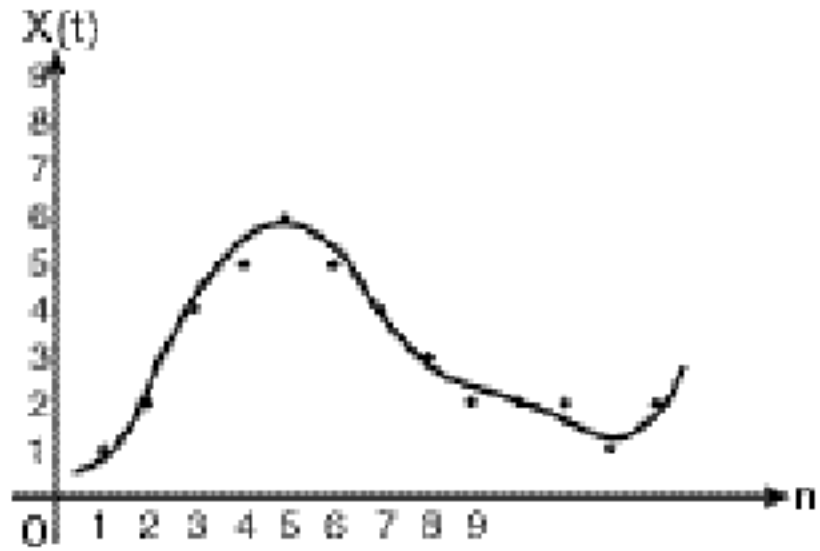


Выборки АЦП

Использование ШИМ в качестве ЦАП



Выход ШИМ



Аналоговый сигнал

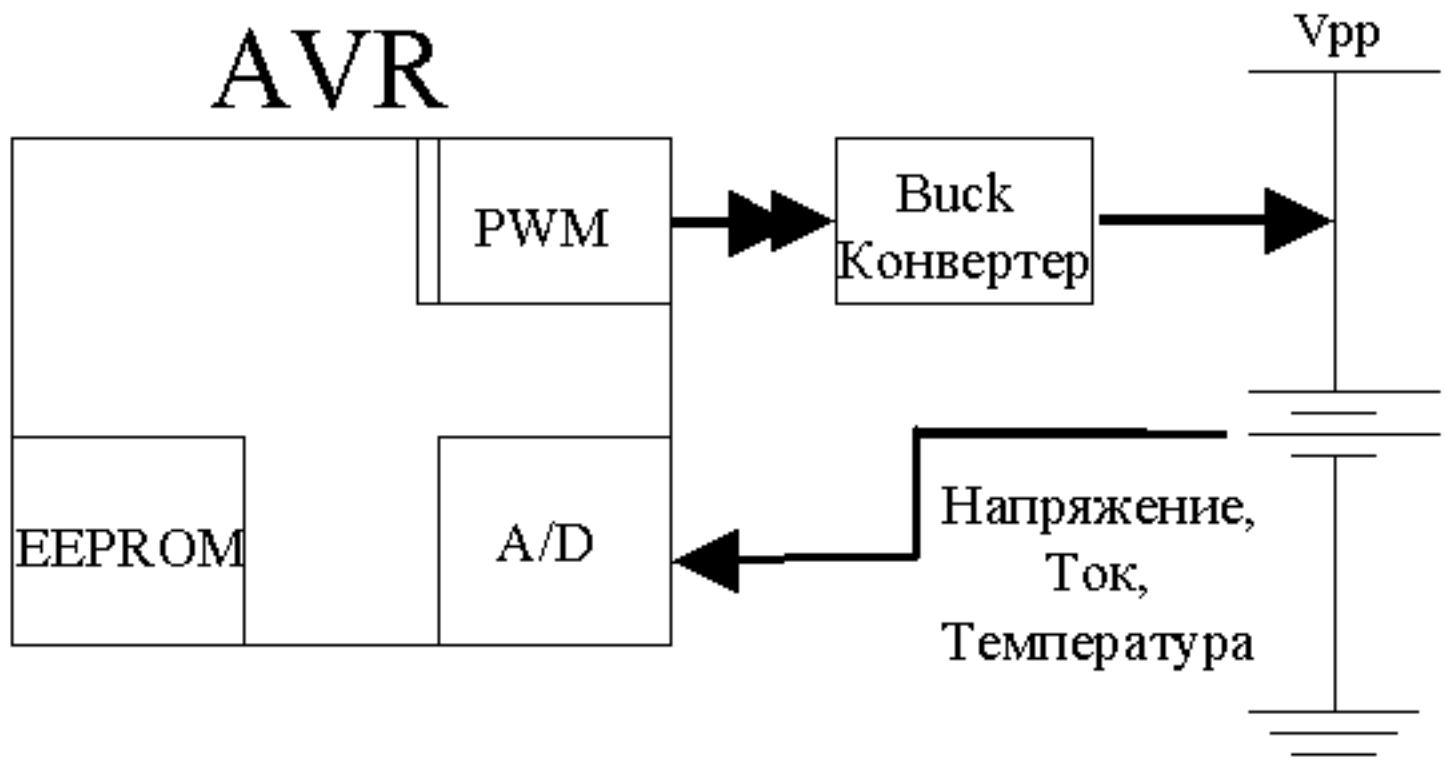
после фильтрации

Цифровой диктофон

- Исходный код на языке Си
- Размер загружаемого кода < 500 байт
- Другие приложения данной схемы:
  - Регистраторы температуры
  - Системы определения местоположения - GPS
  - Датчики
  - Телефонные автоответчики
  - MP3 плееры

Зарядное устройство

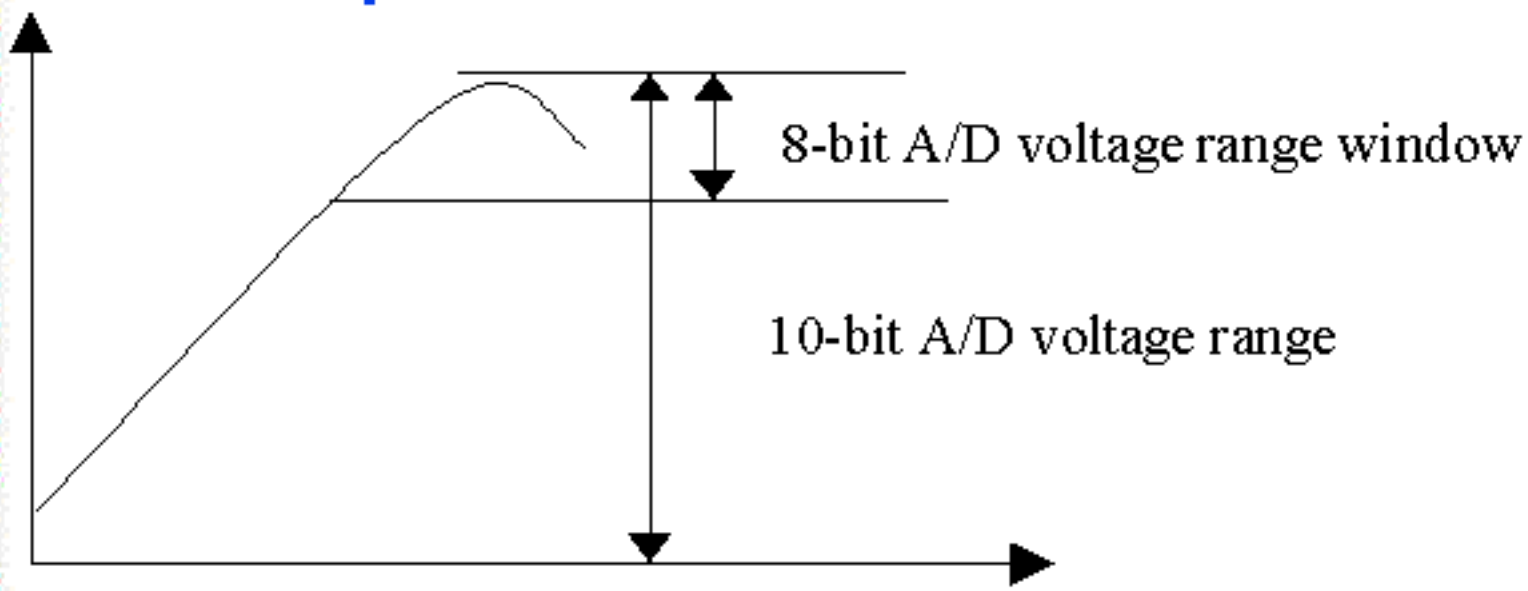
- Исходный код на Си и ассемблере
- Размер загружаемого кода < 350 Байт
- SLA, NiCd, NiMh и Li-Ion аккумуляторы



Зарядное устройство

в зарядном устройстве АЦП измеряет зарядное напряжение, ток и температуру

- Преимущества 10-битного АЦП
  - Большая точность
  - Уменьшенное число внешних компонентов
  - 8-битные АЦП требуют использования внешних операционных усилителей для реализации измерения в “окне”



Зарядное устройство

- ФЛЭШ ПЗУ программ
  - Уменьшает время “доводки” прибора
- ЭСПЗУ
  - Протоколирование параметров каждого конкретного аккумулятора
- ШИМ
  - Используется для управления Buck-конвертером для заряда постоянным током или постоянным напряжением

AVR Flash Microcontrollers

Эффективное программирование на Си для AVR

Рекомендации по уменьшению размера кода

Сравнение ассемблера и Си

Ассемблер	Си
• Полное управление ресурсами	• Ограниченное управление ресурсами
• Компактный и “быстрый” код в малых приложениях	• Большой и “медленный” код в малых приложениях
• Малоэффективный код в больших приложениях	• Эффективный код в больших приложениях
• “Малочитабельный” код	• Структурный код
• Трудоемкая последующая модификация	• Простая последующая модификация
• Непереносимый код	• Переносимый на другие платформы код

Адресация портов ввода/вывода

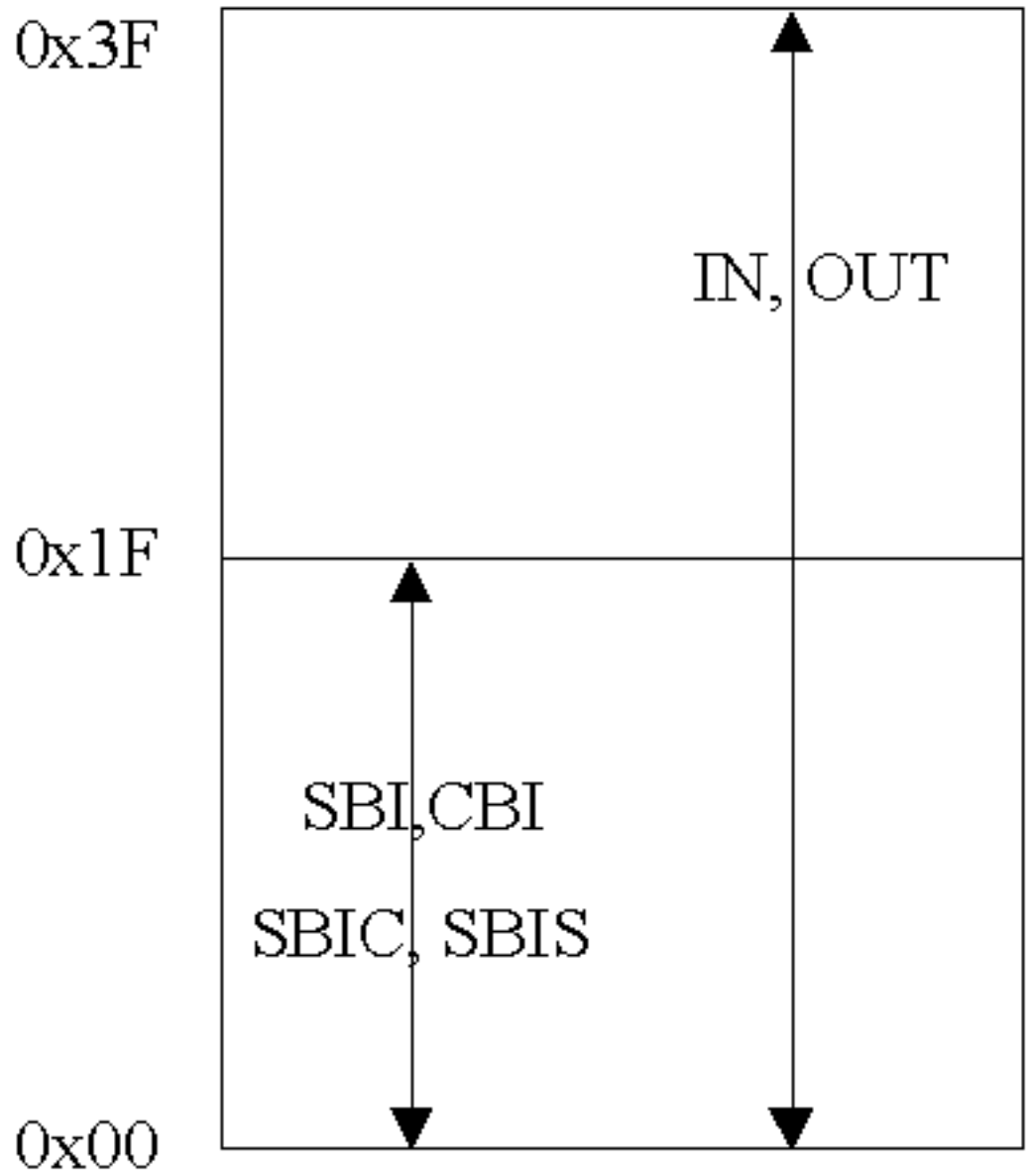
- **Чтение из порта:**

```
temp = PIND;
   IN     R16,LOW(16)
```

- **Запись в порт:**

```
TCCR0 = 0x4F;
   LDI    R16,79
   OUT    LOW(51),R16
```

Карта портов в/в AVR



Установка и очистка бита порта

- **Адресное пространство в/в ниже адреса 0x1F:**

```
PORTB |= (1<<PIND2);
  SBI       LOW(24),LOW(2)

ADCSR &= ~(1<<ADEN);
  CBI       LOW(6),LOW(7)
```

- **Адресное пространство в/в выше адреса 0x1F:**

```
TCCR0 &= ~(0x80);
  IN        R16,LOW(51)
  ANDI      R16,LOW(127)
  OUT       LOW(51),R16
```

Проверка бита

- **Ожидание пока бит=0 для адреса < 0x1F**

```
while(PIND & (1<<PIND6));
   SBIC    LOW(16),LOW(6)
   RJMP    ?0002
```

- **Ожидание пока бит=1 для адреса > 0x1F**

```
while(!(TIFR & (1<<TOV0)));
   IN      R16,LOW(56)
   SBRS    R16,LOW(0)
   RJMP    ?0004
```

16-битные переменные

- **Всегда используйте наименьший возможный тип данных**

8-bit counter:

```
char count8 = 5;
do
{
}while(--count8);
```

```
LDI     R16,5
DEC     R16
BRNE    ?0004
```

Всего - 6 байт

16-bit counter:

```
int count16 = 5;
do
{
}while(--count16);
```

```
LDI     R24,LOW(5)
LDI     R25,0
SBIW    R24,LWRD(1)
BRNE    ?0004
```

Всего - 8 байт

Глобальные и локальные переменные

- **Глобальные переменные**
  - Инициализируются в начале программы
  - Хранятся в ОЗУ
  - Для использования должны быть загружены в регистровый файл
- **Локальные переменные**
  - Инициализируются внутри функции
  - Хранятся в регистре
  - Находятся в регистре до выхода из функции

Глобальные и локальные переменные

Не “увлекайтесь” глобальными переменными

- Local variable
  - void main(void)
  - {
  - char local;
  - local = local - 34;
  - }

- SUBI R17,LOW(34)

- *Total 2 bytes*

- Global Variable:
  - char global;
  - void main(void)
  - {
  - global = global - 45;
  - }

- LDS R16,LWRD(global)
- SUBI R16,LOW(45)
- STS LWRD(global),R16

- *Total 10 Bytes*

Эффективное использование глобальных переменных

- Соберите все глобальные переменные в структуре:

```
typedef struct
{
   intt_count;
   char        sec;          // global seconds
   char        min;          // global minutes
}t;
t time;

Void main(void)
{
   t *temp = &time;
   temp->sec++;
   temp->min++;
   temp->t_count++;
}
```

Вызов функции с параметрами

- Использование параметров для передачи данных между функциями

```
char add(char number1, char number2)
{
    return number1+number2;
}
```

Параметры между функциями передаются в регистрах R16 - R23

Циклы

- Бесконечный цикл:

```
for(; ;)
{
}
```

Обычный цикл:

```
char counter = 100;
do
{
} while(--counter) ;
```

Предекремент переменной дает лучший код

Для оптимизации кода следует:

- Компилировать с оптимизацией по коду
- По возможности использовать локальные переменные
- Использовать минимально возможные типы данных
- Собирать глобальные переменные в структуры
- Использовать for(;;) для бесконечных циклов
- Использовать do{ } while; с предекрементом