

Лекция 1 Введение. Язык программирования С

План курса «Встраиваемые микропроцессорные системы»:

Лекция 1: Введение. Язык программирования С

Лекция 2: Язык программирования С, применение для встраиваемых систем

Лекция 3: Стандартная библиотека языка С

Лекция 4: Ядро ARM Cortex-M3. Микроконтроллер Миландр К1986ВЕ92QI

Лекция 5: Этапы разработки микропроцессорных систем

Лекция 6: Разработка и отладка программ для встраиваемых систем

Лекция 7: Архитектура программного обеспечения

Лекция 8: Периферийные модули: Timer, DMA, ADC, DAC

Лекция 9: Периферийные модули: CAN, USB, Ethernet, SDIO

- Язык программирования С и его применение для встраиваемых микропроцессорных систем;
- Микроконтроллер **Миландр К1986BE92QI (MDR32F9Q2I)**;
 - Ядро ARM Cortex-M3 – 32-х разрядное RISC ядро;
 - Тактовая частота до 80 МГц;
 - ПЗУ 128 Кбайт;
 - ОЗУ 32 Кбайт;
 - Периферия (АЦП, Таймеры, SPI, UART, ЦАП, USB, Ethernet и т.д.).
- Современные методики разработки и отладки встраиваемых микропроцессорных систем.

Литература

Основная литература:

- Б. Керниган, Д. Ритчи Язык программирования С (2-е издание).
- Микроконтроллер Миландр 1986BE9xx:
 - Благодаров А.В., Л.Л. Владимиров Программирование микроконтроллеров;
 - Спецификация микросхем серии 1986BE9xx (Datasheet).

Дополнительная литература:

- Elicia White Making Embedded Systems: Design Patterns for Great Software;
- Joseph Yiu The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors;
- Б. Керниган, Р. Пайк Практика программирования.

Электронные ресурсы

- Материалы курса «Встраиваемые микропроцессорные системы»
<https://github.com/smirnovalnik/embedded-systems-course>
- Материалы курса «Микропроцессорные устройства»
<https://github.com/smirnovalnik/microprocessor-units-course>
- Стандартная библиотека периферии 1986x (Milandr MCU 1986x Standard Peripherals Library) <https://github.com/eldarkg/emdr1986x-std-per-lib>
- Документация и примеры для стандартной библиотеки периферии 1986x (Documentation to Milandr MCU 1986x Standard Peripherals Library)
<https://github.com/eldarkg/emdr1986x-std-per-lib-doc>
- Спецификация на серию 1986BE9x
<https://ic.milandr.ru/upload/iblock/170/vn8t3glqnoucqcncrc4p1y3g79zpiviae/K1986BE92QI.pdf>
- Интегрированная среда разработки MDK-Lite Edition. Версия для обучения.
<http://www2.keil.com/mdk5/editions/lite>
- Ответы на все вопросы по языку C
<https://stackoverflow.com/>

Язык программирования С

Основные вехи развития:

- 1972 – изобретен Д. Ритчи в лаборатории AT&T;
- 1978 – опубликована книга «Язык программирования С» Б. Керниган, Д. Ритчи (K&R C);
- 1989 – C89/ANSI, стандартизация языка;
- 1990 – C90, стандартизация языка ISO;
- 1999 – C99, наиболее используемый стандарт, VLA, inline;
- 2011 – C11, изменения в стандартной библиотеке, generics;
- 2017 – C17, устранение неточностей в C11;
- 2023 – C2x в настоящее время в разработке.

В этом курсе используется C99.

Язык С применяется в:

- системном программировании (95 % ядра Linux на С, драйверы);
- встраиваемых системах (от холодильников до смартфонов, от автомобилей до самолетов).

Язык программирования C: преимущества и недостатки

- ✓ Быстрый код;
- ✓ Минимальная среда исполнения (runtime);
- ✓ Практически полный контроль над аппаратным обеспечением;
- ✓ Широко распространен (библиотеки, промышленный стандарт);
- ✓ Существуют компиляторы практически для всех архитектур;
- ✗ Практически полный контроль над аппаратным обеспечением;
- ✗ небезопасная работа с памятью (нет проверки диапазона переменных, нет проверки типа данных);
- ✗ Не поддерживает современные парадигмы программирования.

Расширением языка являются: C++, Objective C.

Язык повлиял на Java, C#, JavaScript, TypeScript, Solidity, Python, Rust.

Свойства языка программирования С

- Компактность (небольшое количество ключевых слов);
- Доступ к памяти через указатели (адреса);
- Статическая типизация данных (слабая);
- Простые структуры данных – массивы, структуры, объединения;
- Отсутствие сложных встроенных структур данных – списки, деревья, хэш-таблицы;
- Подключаемая стандартная библиотека;
- Препроцессор макросов.

Компилятор С для персонального компьютера

gcc – один из наиболее распространенных компиляторов.

Операционная система	Компилятор	Установка
Linux	gcc	Ubuntu: <code>sudo apt install build-essential</code>
Windows	MinGW	Скачать и установить MingW-W64-builds с сайта https://mingw-w64.org/doku.php/download или через пакетный менеджер Chocolatey. В командной строке запустить: <code>choco install gcc</code>
MacOS	gcc	Скачать и установить менеджер пакетов Homebrew. В командной строке запустить: <code>brew install gcc</code>

Для запуска компилятора из любой директории путь к директории с компилятором должен быть прописан в переменной PATH в ОС Windows.

Результат компиляции будет работать только на системе для которой предназначен компилятор.

Компиляция для микроконтроллера требует другого компилятора (кросс компилятора).

Первая программа на C по K&R

```
/*
    Текст первой программы на языке C
*/

/* Директива препроцессора для добавления файла
из стандартной библиотеки C */
#include <stdio.h>

/* Функция main. Точка входа в программу */
int main(void)
{
    printf("Hello, world!\n"); /* Напечатать строку в терминал */
    return 0; /* Возвратить системе значение 0 - все прошло хорошо */
}
```

Текст программы следует набрать в текстовом редакторе (Блокнот, Notepad++, Visual Studio Code или другие подобные, но не Word!) и сохранить с расширением *.c, например, example.c

Первая программа на С по К&Р: компиляция и запуск

Командную строку следует открыть в директории с исходным файлом или перейти к ней из другой директории.

Компиляция `example.c` из командной строки в исполняемый файл с именем `hello`:

```
gcc example.c -Wall -o hello
```

(ключ `-o` имя исполняемого файл, `-Wall` – отобразить все предупреждения)

Запуск из командной строки (Windows) и результат работы программы:

```
.\hello.exe  
Hello, world!
```

Запуск из командной строки (Linux, MacOS) и результат работы программы:

```
./hello  
Hello, world!
```

Имена объектов. Комментарии

```
/* Каждое утверждение(statement) должно заканчиваться символом «;» */
te+st/ = 1; /* Ошибка: имена должны содержать только буквы, цифры и
символ _ */

1test = 3; /* Ошибка: имена не должны начинаться с цифры */
test = 4; /* ОК */

/* Test и test разные объекты - регистр имеет значение */
Test = 5;
test = 6;

TestLab1 = 7; /* ОК CamelCase */
lesson_number_1 = 8; /* ОК snake_case */
MPEI_ER_02_13 = 2; /* ОК SCREAMING_SNAKE_CASE */
is_valid_parameter(first_param); /* ОК */

/* Многострочный комментарий по
стандарту C. Комментарии полностью игнорируются компилятором. */
a = a + b; // Однострочный комментарий в стиле C++
```

Объявление переменных. Типы данных

```
/* Объявление целочисленной
переменной со знаком
(дополнительный код со
знаком) */
int i;
i = 13;
/* Объявление беззнаковой
целочисленной переменной */
unsigned int j;
j = 0;
/* Объявление переменной
с плавающей запятой */
float a, b;
a = 10.0;
b = 20.0;
/* Объявление и инициализация
символьной переменной */
char c = 'a';
```

Тип данных	Размер, бит	Диапазон
char	8	-128 – 127 или 0 – 255
short, signed short	16	-32768 – 32767
int, signed int	32	$-2^{31} - (2^{31} - 1)$
long long, signed long long	64	$-2^{63} - (2^{63} - 1)$
unsigned char	8	0 – 255
unsigned short	16	0 – 65535
unsigned int	32	0 – $2^{32} - 1$
unsigned long long	64	0 – $2^{64} - 1$
float	32	1,175494351e-38 – 3,402823466e+38
double	64	2,2250738585072014e-308 – 1,7976931348623158e+308

Перед использованием переменные должны быть объявлены.
Размеры типов данных могут зависеть от архитектуры
(центрального процессора и шины данных) и от компилятора.

Оператор присваивания

```
/* Оператор присваивания – скопировать содержимое из одной ячейки памяти  
в другую */
```

```
/* Объявление и инициализация */
```

```
int a = 1;
```

```
int b = 2;
```

```
/* Обмен значений a и b через промежуточную переменную */
```

```
int tmp;
```

```
tmp = a;
```

```
a = b;
```

```
b = tmp; /* a = 2, b = 1 */
```

```
int i, j, k;
```

```
/* Множественное присваивание: k = 0, j = k -> j = 0, i = j -> i = 0 */
```

```
i = j = k = 0;
```

Арифметические операции

```
int a = 0;
int b = 2;
int c;
/* Сложение */
c = a + b; /* c = 2 */
c = c + 2; /* c = 4 */
/* Вычитание */
c = b - a; /* c = 2 */
/* Умножение */
c = 4 * b; /* c = 8 */
/* Деление */
a = 10;
c = a / 2; /* a = 5 */
c = a / 100; /* a = 0 */
/* Остаток от деления */
a = 13;
c = a % 10; /* c = 3 */
```

Оператор	Описание
+	сложение
-	вычитание
/	целочисленное деление для char, short, int, long, деление для float, double
*	умножение
%	взятие остатка от деления (для целочисленных переменных)

Арифметические операции: деление

```
int a;
```

```
/* Целочисленное деление */
```

```
a = 1 / 2; /* a = 0 */
```

```
a = 10 / 100; /* a = 0 */
```

```
a = 3 / 2; /* a = 1 */
```

```
a = 11 / 2; /* a = 5 */
```

```
float c;
```

```
/* Деление чисел с плавающей запятой */
```

```
c = 1.0 / 2.0; /* c = 0.5 */
```

```
c = 1 / 2.0; /* c = 0.5 */
```

```
c = 1.0 / 2; /* c = 0.5 */
```

```
c = 1 / 2; /* c = 0.0 */
```

В простых микропроцессорных системах, как правило, отсутствуют аппаратные блоки работы с числами с плавающей запятой (floating point). Компилятор производит программную эмуляцию данных операций, что занимает много циклов.

Поэтому в таких системах часто используют числа с фиксированной запятой (fixed point).

float по стандарту IEEE 754

	Знак S	Показатель степени E	Мантисса M
Бит	31	30:23	22:0

$$\text{Float} = (-1)^S \times M \times 2^E$$

Арифметические операции: сокращенная форма

```
int i = 0;
```

```
i++; /* i = 1 */
```

```
++i; /* i = 2 */
```

```
/* Разница между префиксной и  
постфиксной формой */
```

```
i = 0;
```

```
a = i++; /* a = 0, i = 1 */
```

```
a = ++i; /* a = 2, i = 2 */
```

```
i = 0;
```

```
i += 10; /* i = 10 */
```

```
a = 2;
```

```
i /= a + 3; /* i = 2 */
```

Оператор	Описание	Действие
<code>i++;</code>	Инкремент, постфиксная форма	<code>i = i + 1;</code>
<code>++i;</code>	Инкремент, префиксная форма	<code>i = i + 1;</code>
<code>i--;</code>	Декремент, постфиксная форма	<code>i = i - 1;</code>
<code>--i;</code>	Декремент, префиксная форма	<code>i = i - 1;</code>
<code>i += 1;</code> <code>i += 2;</code> <code>i -= 3;</code> <code>i *= 2;</code> <code>i /= 2;</code> <code>i %= 2;</code>	Сокращенная форма	<code>i = i + 1;</code> <code>i = i + 2;</code> <code>i = i - 3;</code> <code>i = i * 2;</code> <code>i = i / 2;</code> <code>i = i % 2;</code>

Битовые операции

/* Запись констант:

0x0A - шестнадцатеричная форма,

012 - восьмеричная форма,

10 - десятичная форма */

```
int a = 0x88;
```

```
a = a & 0xFE; /* Сброс бита a = 0x80 */
```

```
a = a | 0x02; /* Установка бита a = 0x82 */
```

```
a = a ^ 0x01; /* Инверсия бита a = 0x83 */
```

```
a = a ^ 0x01; /* Инверсия бита a = 0x82 */
```

```
a = ~a; /* Инверсия a = 0x7D */
```

```
a = 0x01;
```

```
a = a << 2; /* Сдвиг влево a = 0x04 */
```

```
a = a >> 1; /* Сдвиг вправо a = 0x02 */
```

Оператор	Описание
&	Побитовое И
	Побитовое ИЛИ
^	Побитовое исключающее ИЛИ
~	Побитовая инверсия
>>	Сдвиг вправо
<<	Сдвиг влево
<pre>i &= 0xFE; i = 0x02; i ^= 1; i <<= 2; i >>= 1;</pre>	Сокращенная форма

Оператор ветвления

/ Простая форма без {} */*

```
if (выражение)
    оператор;
```

/ Простая форма с {} */*

```
if (выражение) {
    оператор1;
    оператор2;
}
```

/ Проверка на четность */*

```
if (a % 2 == 0)
    cnt++; // Одно утверждение
```

/ Простая форма с {} */*

```
if (is_ready) {
    timeout = 100;
    status = OK;
}
```

Обобщенная форма

Пример

Оператор ветвления

/ Полная форма */*

```
if (выражение) {  
    оператор1;  
}
```

```
else {  
    оператор2;  
}
```

```
if (выражение1) {  
    оператор1;  
}
```

```
else if (выражение2) {  
    оператор2;  
}
```

```
else {  
    оператор3;  
}
```

Обобщенная форма

```
if (voltage < 100) {  
    status = OK;  
}
```

```
else {  
    status = FAIL;  
}
```

```
if (cmd == RUN) {  
    run();  
}
```

```
else if (cmd == STOP) {  
    stop();  
}
```

```
else {  
    idle();  
}
```

Пример

Операция сравнения и логические операции

```
if (a < 0) {  
    ...  
}  
  
if ((status & 0x01) == 0) {  
    ...  
}  
  
if ((a > 0) && (a != 10)) {  
    ...  
}  
  
if (!is_stopped) {  
    ...  
}
```

Оператор	Описание
> < >= <=	Больше Меньше Больше или равно Меньше или равно
== !=	Равно Не равно
&&	Логическое И
	Логическое ИЛИ
!	Логическое НЕ

Не следует путать оператор присваивания = и оператор сравнения ==. Компилятор может не указать об ошибке.

Оператор switch

```
switch (выражение) {  
    case константа1:  
        оператор1;  
        оператор2;  
        break;  
    case константа2:  
        оператор3;  
        break;  
    case константа3:  
    case константа4:  
        оператор4;  
        break;  
    default: // необязательно  
        оператор5;  
}
```

Обобщенная форма

```
switch (cmd) {  
    case CMD_RUN:  
        set_pwm(100);  
        run();  
        break;  
    case CMD_STOP:  
        stop();  
        break;  
    case CMD_IDLE:  
    case CMD_RESET:  
        idle();  
        break;  
    default:  
        error();  
}
```

Пример

Массивы

```
/* Объявление массива без инициализации */
```

```
int arr[10];
```

```
arr[0] = 1; /* Первый элемент */
```

```
arr[1] = 4; /* Второй элемент */
```

```
arr[9] = 99; /* Последний элемент */
```

```
/* Компилятор не проверяет  
выход за пределы */
```

```
arr[-1] = 100; /* Ошибка при исполнении */
```

```
arr[10] = 100; /* Ошибка при исполнении */
```

```
/* Объявление массива с инициализацией */
```

```
char letters[] = {'x', 'y', 'z'};
```

Индекс	Адрес	Данные
...
		Другие данные
[0]	arr + 0	1
[1]	arr + 4	4
[2]	arr + 8	13
...
[9]	arr + 36	99
		Другие данные
...

Мл. адрес

Ст. адрес

Многомерные массивы

```
int marr[10][10];
```

```
marr[0][0] = 13; /* Первый элемент */
```

```
marr[0][1] = 27;
```

```
marr[0][9] = 43;
```

```
marr[1][0] = 69;
```

```
marr[9][8] = 87;
```

```
marr[9][9] = 42; /* Последний элемент */
```

Мл. адрес	Индекс	Адрес	Данные

Ст. адрес			Другие данные
	[0][0]	marr+0	13
	[0][1]	marr+4	27

	[0][9]	marr+36	43
	[1][0]	marr+40	69

	[9][8]	marr+392	87
	[9][9]	marr+396	42
			Другие данные

Операторы цикла

`/* Цикл while */`

```
while (выражение) {  
    оператор;  
}
```

`/* Цикл do-while */`

```
do {  
    оператор;  
} while (выражение);
```

`/* Цикл for */`

```
for (выраж1; выраж2; выраж3) {  
    оператор;  
}
```

Обобщенная форма

```
while (i < 10) {  
    a[i] = 0;  
    i++;  
}
```

```
spi_send_byte(cmd);  
do {  
    status = spi_get_status();  
} while (status != SPI_OK);
```

```
/* "Зануление" массива */  
for (i = 0; i < 10; i++) {  
    a[i] = 0;  
}
```

```
/* Бесконечный цикл */  
for(;;);  
while(1);
```

Пример

Операторы цикла

```
/* Оператор break */
while (выражение1) {
    ...
    if (выражение2)
        break;
    ...
}
```

```
/* Оператор continue */
while (выражение1) {
    ...
    if (выражение2)
        continue;
    ...
}
```

Обобщенная форма

```
/* Проверка наличия элемента в массиве */
int is_found = 0, i = 0;
while (i < n) {
    if (a[i] == target) {
        is_found = 1;
        break;
    }
    i++;
}
/* Обработка только положительных элементов */
int i;
for (i = 0; i < n; i++) {
    if (a[i] < 0)
        continue;
    ...
}
```

Пример

- Язык программирования С:
 - Основные сферы применения: системное программирование, встраиваемые системы;
 - Основные типы данных: целочисленные (char, int, long), с плавающей запятой (float, double), символьные (char);
 - Арифметические, побитовые, логические операции и операции сравнения;
 - Операторы ветвления (if, switch);
 - Операторы цикла (do while, while, for).
- Рекомендуется начать самостоятельное изучение языка по книге Б. Керниган, Д. Ритчи Язык программирования С