Лекция 2 Язык программирования С, применение для встраиваемых систем

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 программ: компилятор, сборщик, отладчик, интегрированная среда разработки

Лекция 7: Внутрисхемная отладка, загрузка программы, трассировка

Лекция 8: Архитектура программного обеспечения

Лекция 9: Периферийные модули: Timer, DMA, ADC, DAC

Лекция 10: Периферийные модули: CAN, USB, Ethernet, SDIO

Объявление и вызов функций

```
/* Объявление функций */
тип-возвр-значения имя-функции(аргументы)
{
    объявления;
    операторы;
}

/* Вызов функций */
переменная = имя-функции(аргументы);
```

Обобщенная форма

```
int add_two(int n)
{
    int tmp = 0;

    tmp = n + 2;
    return tmp;
}
a = add_two(2);
```

Пример

Объявление и вызов функций

```
/* Объявление функций */
/* Абсолютное значение числа */
int abs(int n)
{
    if (n < 0)
        return -n;
    else
        return n;
}

/* Поиск максимального элемента в массиве */
int arr_max(int a[], int n)
{
    int max = a[0];
    for (int i = 1; i < n; i++)
    if (a[i] > max)
        max = a[i];
    return max;
}
```

```
/* Включить светодиод */
void led_on()
{
    PORTC |= 0x01;
}

/* Вызов функций */
void main()
{
    int c;
    c = abs(-10); /* c = 10 */

    int arr[5] = {4, 3, 1, 7, 5};

    c = arr_max(arr, 5); /* c = 7 */
    led_on();
}
```

Объявление и вызов функций

```c
/* Объявление функций */
/* Обратить порядок элементов в массиве */
void reverse(char a[], int n)
{
    int i, j, tmp;
    for (i = 0, j = n - 1; i < j; i++, j--){
        tmp = a[i];
        a[i] = a[j];
        a[j] = tmp;
    }
    return; /* Необязательно */
}
```

```c
/* Вызов функций */
void main()
{
    char arr[5] = {4, 3, 1, 7, 5};
    char len = 5;

    reverse(arr, len);
    /* arr = {5, 7, 1, 3, 4} */
    int i;
    for (i = 0; i < len; i++){
        printf("%d ", arr[i]);
    }
}
```

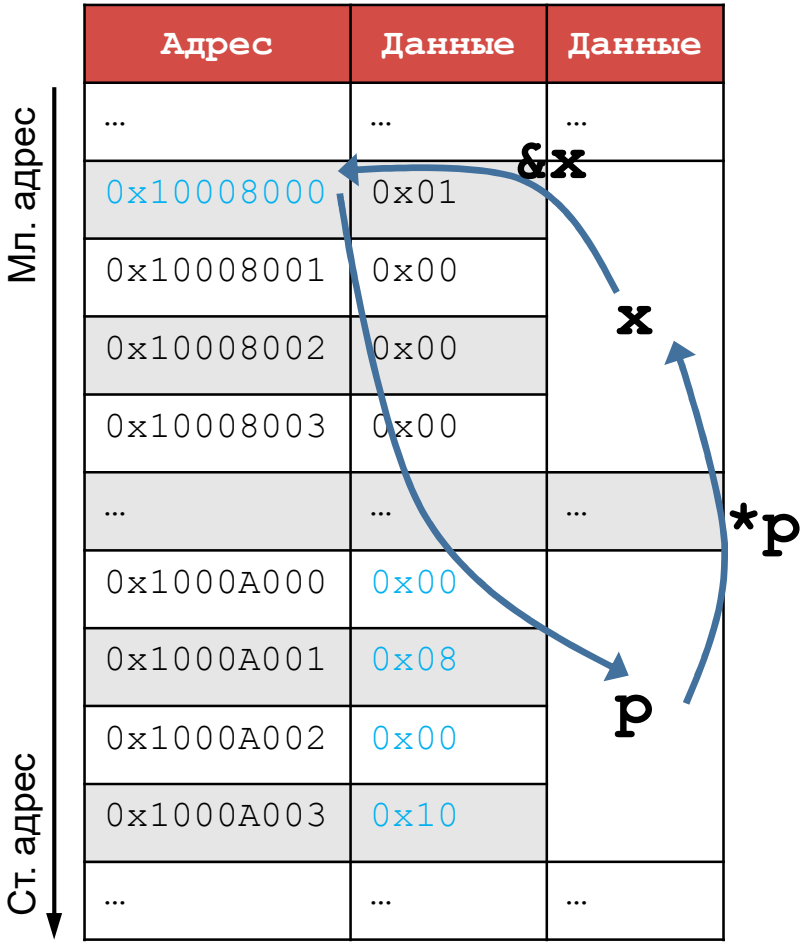
Указатели

```
/* Объявление переменных и массива */
int x = 1, y = 2, z[10];
/* Объявление указателя */
int *p;

p = &x;/* p - адрес переменной x
или p - указывает на x */

/* Разыменование *p */
y = *p; /* *p = x = 1 -> y = 1 */
*p = 0; /* x = 0 */

p = &z[0];/*p - указывает на z[0]*/
*p = 100;/* z[0] = 100 */
```



Массивы и указатели

```
/* Объявление массива */
int a[10];
/* Объявление указателя */
int *pa;

pa = &a[0]; /* Указатель на a[0] */
pa = a; /* Эквивалентно pa = &a[0] */

/* a[i] эквивалентно *(pa + i) */
*pa = 0; /* a[0] = 0 */
*(pa + 2) = 2; /* a[2] = 2 */
```

Мл. адрес

Ст. адрес

Индекс	Указатель	Адрес	Данные
…	…		…
			Другие данные
a[0]	*pa	pa	1
a[1]	*(pa+1)	pa+4	4
a[2]	*(pa+2)	pa+8	13
…	…		…
a[9]	*(pa+9)	pa+36	99
			Другие данные
…	…		…

МЭИ

Строки и символы

```
/* Объявление и инициализация
   строки */
char msg[] = "Hello\r\n";
/* Объявление и инициализация
   символа */
char ch = 'a';
```

Мл. адрес

Ст. адрес

Индекс	Адрес	Данные (HEX)	Данные (ASCII)
…	…		…
		Другие данные	
msg[0]	msg	0x48	H
msg[1]	msg+1	0x65	e
msg[2]	msg+2	0x6C	l
msg[3]	msg+3	0x6C	l
msg[4]	msg+4	0x6F	o
msg[5]	msg+5	0x**0D**	\r
msg[6]	msg+6	0x0A	\n
msg[7]	msg+7	0x00	\0
		Другие данные	
…	…	…	…

МЭИ

Структуры

```
/* Объявление структуры */
struct point {
    int x;
    int y;
};

/* Объявление переменной pt1 типа point */
struct point pt1;

/* Инициализация полей структуры */
pt1.x = 22;
pt1.y = 7;

/* Инициализация структуры */
struct point pt2 = {-10, 0};
```

Объединения

```
/* Объявление объединения */
union {
     int word;
     short hword[2];
     char byte[4];
} u;

u.word = 0x12345678;

short lo_hword = u.hword[0];/* lo_hword = 0x5678 */
char hi_byte = u.byte[3];/* hi_byte = 0x12 */
```

Адрес	0	1	2	3
Память	0x78	0x56	0x34	0x12
word	0x12345678			
hword	0x5678		0x1234	
byte	0x78	0x56	0x34	0x12

Константы

```
/* Макроопределения */
#define MAXLEN 100
char msg[MAXLEN + 1];

#define TRUE 1
#define FALSE 0

if (res == TRUE)
    ...

/* Перечисления */
enum month {JAN = 1, FEB, MAR, APR,
MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC};
/* FEB = 2, MAR = 3, ...*/

enum boolean {FALSE = 0, TRUE};/* TRUE = 1 */
```

Битовые операции

```
/* Битовые поля */
struct {
    unsigned int enable : 1;
    unsigned int test: 1;
    unsigned int err_code: 3;
} state_flags;

state_flags.enable = 1;
state_flags.test = 0;
state_flags.err_code = 4;
```

```
/* Пример без битовых полей */
#define ENABLE_FLAG 1
#define TEST_FLAG 2
#define ERR_MASK 7
#define ERR_SHIFT 2

unsigned int flags;

flags |= ENABLE_FLAG;/* Установить бит 0*/
flags = flags | ENABLE_FLAG;
flags &= ~TEST_FLAG;/* Сбросить бит 1 */


/* Сбросить код ошибки */
flags &= ~(ERR_MASK << ERR_SHIFT);
/* Установить новый код ошибки */
flags |= (4 & ERR_MASK) << ERR_SHIFT;
```

Препроцессор

```
/* Макроопределения */
#define ARR_SIZE 10
#define TRUE 1
#define FALSE 0
#define STEP 100

/* Подключение файлов */
#include "adc.h"
#include <stdio.h>


//#define DEBUG

/* Условная компиляция */
#ifdef DEBUG
...
#endif
```

Применение языка С для встраиваемых систем

- Обращение к регистрам специальных функций периферийных модулей;
- Обработка прерываний;
- Ассемблерные вставки.

Уровни абстракции:
Обращение к регистрам специальных функций

1. Язык С + документация

```
#define PORTC *((volatile unsigned int *)(0x400B8000))
...
PORTC = PORTC | 1; /* Установить 1 в PC0 */
```

2. Язык С + заголовочные файл + документация

```
#include "MDR32Fx.h"
...
MDR_PORTC->RXTX = MDR_PORTC->RXTX | 1; /* Установить 1 в PC0 */
```

3. Язык С + библиотека (например, Standard Peripheral Library для К1986ВЕ92QI)

```
#include <MDR32F9Qx_port.h>
...
PORT_SetBits(MDR_PORTC, PORT_Pin_0); /* Установить 1 в PC0 */
```

Первая программа на С для микроконтроллера. Версия 1

```
#define RST_CLK_PER  *((volatile unsigned int *)(0x4002001C))
#define PORTC_RXTX   *((volatile unsigned int *)(0x400B8000))
#define PORTC_OE     *((volatile unsigned int *)(0x400B8004))
#define PORTC_ANALOG *((volatile unsigned int *)(0x400B800C))
#define PORTC_PWR    *((volatile unsigned int *)(0x400B8018))

/* Функция main. Точка входа в программу */
int main(void)
{
    RST_CLK_PER = RST_CLK_PER | (1 << 23); /* Включаем тактирование порта C */

    PORTC_OE = PORTC_OE | 0x01;            /* Настраиваем ножку 0 порта C на вывод */
    PORTC_ANALOG = PORTC_ANALOG | 0x01;    /* Включаем цифровой режим работы ножки 0 */
    PORTC_PWR = PORTC_PWR | 0x02;          /* Настраиваем мощность выходного буфера ножки 0 */

    for (;;)
    {
        PORTC_RXTX = PORTC_RXTX ^ 0x01;   /* Инвертирование бита в регистре PORTC */
        for (int i = 0; i < 100000; i++); /* Программная задержка */
    }
}
```

Первая программа на С для микроконтроллера. Версия 2

```c
#include <MDR32Fx.h>

/* Функция main. Точка входа в программу */
int main(void)
{
    /* Включаем тактирование порта С */
    MDR_RST_CLK->PER_CLOCK = MDR_RST_CLK->PER_CLOCK | (1 << 23);

    MDR_PORTC->OE = MDR_PORTC->OE | 0x01;          /* Настраиваем ножку 0 порта С на вывод */
    MDR_PORTC->ANALOG = MDR_PORTC->ANALOG | 0x01; /* Включаем цифровой режим работы ножки 0 */
    MDR_PORTC->PWR = MDR_PORTC->PWR | 0x02; /* Настраиваем мощность выходного буфера ножки 0 */

    for (;;)
    {
        MDR_PORTC->RXTX = MDR_PORTC->RXTX ^ 0x01; /* Инвертирование бита в регистре PORTC */
        for (int i = 0; i < 100000; i++); /* Программная задержка */
    }
}
```

Первая программа на С для микроконтроллера. Версия 3

```
#include <MDR32Fx.h>
#include <MDR32F9Qx_config.h>
#include <MDR32F9Qx_rst_clk.h>
#include <MDR32F9Qx_port.h>
int main()
{
    RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTC, ENABLE);
    PORT_InitTypeDef Port_InitStructure;
    PORT_StructInit(&Port_InitStructure);
    Port_InitStructure.PORT_Pin = PORT_Pin_0;
    Port_InitStructure.PORT_OE = PORT_OE_OUT;
    Port_InitStructure.PORT_FUNC = PORT_FUNC_PORT;
    Port_InitStructure.PORT_SPEED = PORT_SPEED_FAST;
    Port_InitStructure.PORT_MODE = PORT_MODE_DIGITAL;
    PORT_Init(MDR_PORTC, &Port_InitStructure);
    for(;;) {
        for (int i = 0; i < 100000; i++);
        PORT_SetBits(MDR_PORTC, PORT_Pin_0);
        for (int i = 0; i < 100000; i++);
        PORT_ResetBits(MDR_PORTC, PORT_Pin_0);
    }
}
```

Обработка прерываний в Cortex-M3

Файл startup_MDR32F9Qx.s:

```
...
DCD Timer1_IRQHandler ; IRQ14
DCD ADC_IRQHandler ; IRQ17
...
```

Файлы разработчика, например, main.c:

```
/* Обработка прерывания по Timer1 */
void Timer1_IRQHandler(void)
{
...
}
/* Обработка прерывания по ADC */
void ADC_IRQHandler(void)
{
...
}
```

Ассемблерные вставки

Использование ассемблера:

- Для оптимизации по скорости выполнения и размеру программы;
- Для прямого манипулирования регистрами;
- Для использования старого ассемблерного кода в новых проектах;
- Для специальных инструкций (WFI, BKP, SVC);
- Для учебных целей.

Ассемблерные вставки

```
/* Для IDE Keil uVision */
__asm void add(int x1, int x2, int x3)
{
    ADDS R0, R0, R1
    ADDS R0, R0, R2
    BX LR
}

int swap32(int i)
{
    int res;
    __asm {
        REVSH res, i
    }
    return res;
}

__asm("WFI"); /* Выполнение одной команды */
```

Результат компиляции: HCS08

Код на языке C

```
 1. void func(void)
 2. {
 3.   return;
 4. }
 5.
 6. void main(void)
 7. {
 8.    char i;
 9.
10.    i = 5;
11.    while (i > 0)
12.    {
13.        func();
14.        i--;
15.    }
16.    for (;;);
17. }
```

Результат компиляции для HCS08

```
...
 8:      char i;
 9:
10:      i = 5;
         LDA   #5
         TSX
         STA   ,X
         L5:
11:      while (i > 0)
12:      {
13:          func();
         BSR   func
14:          i--;
         TSX
         DEC   ,X
         TST   ,X
         BNE   L5
         LC:
15:      }
16:      for (;;);
         BRA   LC
17:  }
```

Результат компиляции: ARM Cortex-M3

Код на языке C

```
 1. void func(void)
 2. {
 3.   return;
 4. }
 5.
 6. void main(void)
 7. {
 8.     char i;
 9.
10.     i = 5;
11.     while (i > 0)
12.     {
13.         func();
14.         i--;
15.     }
16.     for (;;);
17. }
```

Результат компиляции для ARM Cortex-M3

```
    ...
8000136:   b580    push    {r7, lr}
8000138:   b082    sub sp, #8
800013a:   af00    add r7, sp, #0
    char i;

    i = 5;
800013c:   2305    movs    r3, #5
800013e:   71fb    strb    r3, [r7, #7]
8000140:   e004    b.n 800014c <main+0x16>
    while (i > 0)
    {
        func();
8000142:   f7ff fff3   bl  800012c <func>
        i--;
8000146:   79fb    ldrb    r3, [r7, #7]
8000148:   3b01    subs    r3, #1
800014a:   71fb    strb    r3, [r7, #7]
while (i > 0)
800014c:   79fb    ldrb    r3, [r7, #7]
800014e:   2b00    cmp r3, #0
8000150:   d1f7    bne.n   8000142 <main+0xc>
8000152:   e7fe    b.n 8000152 <main+0x1c>
```

МЭИ

Заключение

- Язык программирования С:
  - Объявление и вызов функций;
  - Указатели;
  - Структуры, объединения, перечисления;
  - Оператор ветвления;
  - Операторы цикла;
  - Препроцессор.

- Язык С позволяет напрямую обращаться к памяти через указатели, но не позволяет обращаться к регистрам процессора.

- Применение для встраиваемых систем:
  - Обращение к регистрам специальных функций через указатели;
  - Обработка прерываний;
  - Ассемблерные вставки.