

MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN
FEDERATION

FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF HIGHER
EDUCATION
"NORTH CAUCASUS FEDERAL UNIVERSITY"

SUGGESTIONS FOR TEACHING
for the performance of practical work
discipline "Principle and Application of MCU"
for students of the field of study

11.03.02 Infocommunication Technologies and Communication Systems,
Specialization "Intelligent Data Processing in Infocommunication Systems and
Networks"

Stavropol, 2026

	3
TABLE OF CONTENTS INTRODUCTION	4
1. PRACTICAL WORK No1. STRUCTURE MICROCONTROLLERS.....	5
2. PRACTICAL WORK No2. STRUCTURE OF THE PROCESSOR CORE MICROCONTROLLER.....	10
3. PRACTICAL WORK No3. MEMORY OF PROGRAMS AND DATA MICROCONTROLLER.....	15
4. PRACTICAL WORK No4. PORTS I/O.....	19
5. PRACTICAL WORK No5. HARDWARE TO ENSURE RELIABLE OPERATION OF THE MC	24
REFERENCES	29

Introduction

A microprocessor system can be considered as a special case of an electronic system designed to process input signals and output signals. Analog signals, single digital signals, digital codes, and sequences of digital codes can be used as input and output signals. Signals (or information) can be stored and accumulated within the system, but the essence does not change. If the system is digital (and microprocessor systems are digital), then the input analog signals are converted into sequences of sample codes using the ADC, and the analog output signals are formed from a sequence of sample codes using a DAC. Information processing and storage are carried out in digital form.

In this course of works, it is proposed to get acquainted with the internal structure of microprocessor systems. The main areas covered are: the structure of microcontrollers, the structure of the processor core of the microcontroller, the memory of programs and data of the microcontroller, I/O ports, hardware to ensure the reliable operation of the MCU.

Practical Work No1

Structure of microcontrollers

The purpose of the work is to familiarize with the operation of the main hardware of modern microcontrollers.

General information

All MCUs have a processor core. The family includes a number of models. Individual models within the family differ in a set of peripheral modules that are connected to the internal intermodule backbone. The main differences between individual representatives of the family are the type and volume of resident memory placed on the chip, the number of parallel ports and serial interface controllers. The structure of modern microcontrollers on the example of MCU MC68HC912B32 is shown in Fig. 1

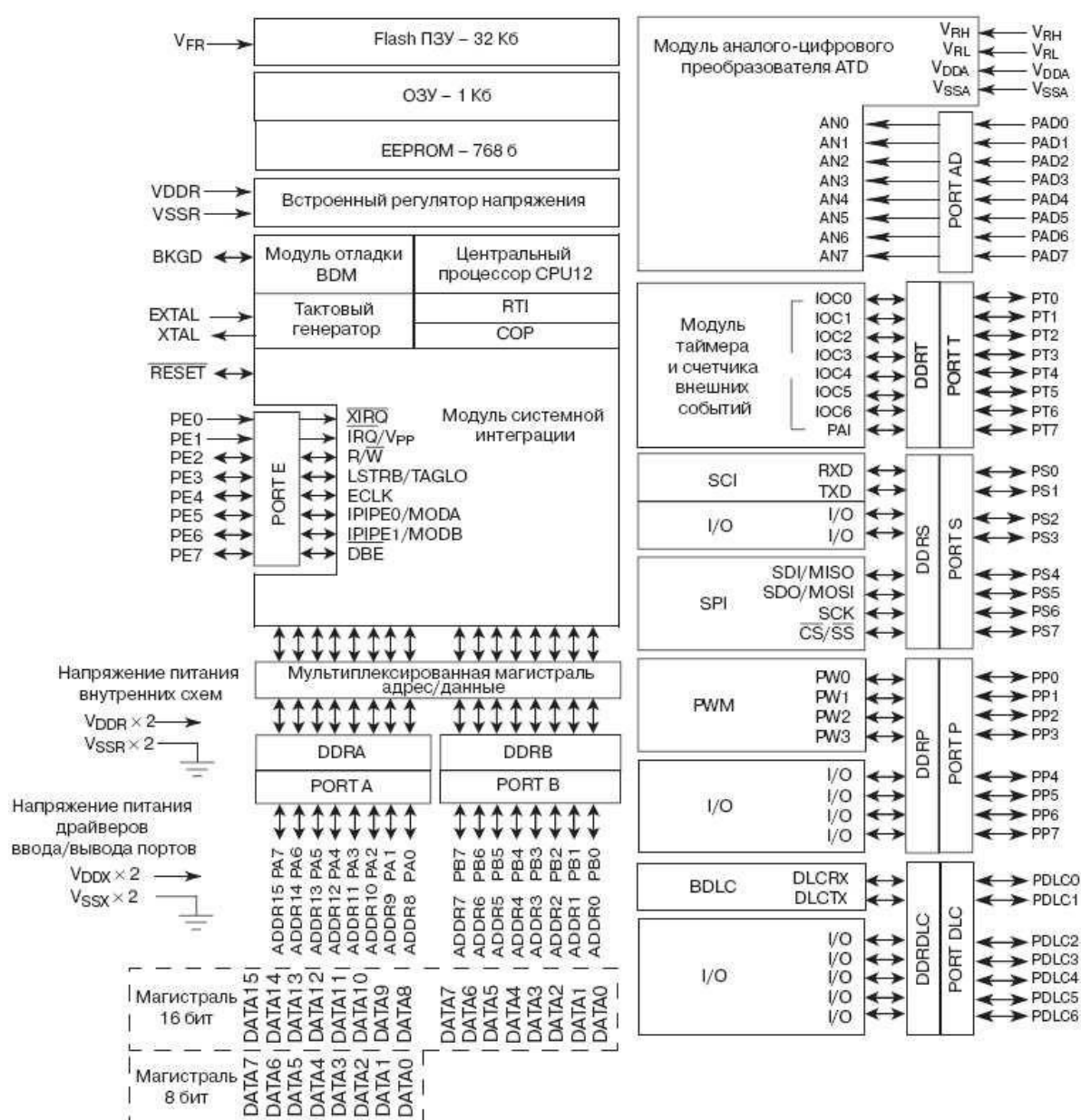


Fig.1. Structure of modern microcontrollers

Features of modern microcontrollers on the example of Motorola MC

In-house RAM of 1024 bytes (1Kb). The RAM built into the microcontroller die is used to store intermediate computational results. The number of cells equal to 1K is sufficient for most of the applications predicted for the 68HC12 family.

Resident non-volatile data memory of the EEPROM type with a capacity of 768 b. This type of non-volatile memory (non-volatile memory retains content after a power outage) is usually used to store changeable constants of an application program. For example, access codes to a given device model can be stored in the EEPROM area, or alarm counters of the actuator controlled by a microprocessor controller can be organized on the basis of EEPROM cells. Non-volatile memory of the EEPROM type allows you to write and rewrite the contents of memory cells during a session of operation of a microprocessor device under the control of an application program, as well as read memory cells in an arbitrary order.

32 KB Flash program resident memory. This type of memory is designed to store an application program that is functionally complete, has been debugged and tested on a real object. The memory capacity of MK B32 programs is 32 Kb, which allows you to place fairly large programs in it. In the process of working on examples from our book, you will get a feel for what algorithm can be implemented by 32 KB program code. Using Flash memory as program memory allows you to implement programming technology in the ISP (In System Programming) system. This technology allows you to erase and write new codes to the resident ROM of microcontroller programs without dismantling the MCU from the final product board.

Multiplexed address/data bus for addressing external memory and peripherals. The number of pins in the package that houses the semiconductor chip of the microcontroller is limited. The reason for this is the desire of the developer to produce miniature and relatively inexpensive MCUs, while increasing the number of pins in the package increases its size and cost. One way to reduce the number of MCU pins while maintaining the functions of these pins is to multiplex the address and data trunk lines to interface the MCU with external memory. In multiplexing, the same MCU pins are used to transmit information about the address of an external memory cell during one time interval, and during another time interval to exchange data with this cell. In the B32 microcontroller, the functions of the time-multiplexed address/data trunks are performed by the Port A and Port B port lines (see Fig.).

Hardware of modern microcontrollers on the example of Motorola MC

Multifunctional timer. The real-time subsystem of the MCU 68HC12 family includes several modules, but the main one is a timer with a 16-bit time base counter, a programmable clock divider and 8 channels of Input Capture (IC) or Output Compare (OC). These channels can be configured arbitrarily: any number of channels out of 8 is configured to implement the IC input capture function, the remaining channels are configured to the OC output comparison function. At the same time, configurations are possible when all channels are in IC mode or in OC mode. This arrangement of the timer module allows accurate measurements of the timing of the input signals of the MCU signals, and generates Multi-channel Pulse Sequences on his outputs. • Independent 16-bit external event counter. This module also

belongs to the real-time subsystem. It is designed to count the so-called external events, each of which is represented by a pulse at one of the MCU inputs.

Serial Peripheral Interface (SPI) and SCI (Serial Communication Interface) Controller Modules. The 68HC12 family of microcontrollers has powerful enough hardware to exchange with other devices in serial code. The SPI controller module implements synchronous mode exchange, while the SCI module is designed for asynchronous serial exchange. Synchronous exchange mode in the SPI standard is characterized by higher exchange rates compared to the standard asynchronous protocol. However, the distance between interacting devices is limited to 20... 30 cm. The SPI interface is often used to connect additional interface components installed on the MCU board to the MCU. For example, the 68HC12 MCU family does not include digital-to-analog converters (DACs). Therefore, the MCU system can be supplemented with an external DAC IC connected to the microcontroller using the built-in SPI controller module. The SCI asynchronous exchange interface is often used for data exchange between two or more controllers, because it is based on interfaces whose signals can be transmitted over long distances.

Analog-to-digital converter module ATD (Analog To Digital conversion system). Most of the signals in systems controlled by microcontrollers are analog in nature. For example, the temperature and air pressure of the environment change smoothly rather than spasmodically. In order to work with analog signals in a microprocessor system, these signals must first be converted into digital form. In Russian-language literature, it is said that these signals must be digitized. For this purpose, the MC B32 includes an analog-to-digital converter module. The module has 8 inputs for simultaneous connection of eight analog signals to be measured. However, the digitization of these signals will be carried out in series. The measured analog signals will be connected to the input of one analog-to-digital converter (ADC) by means of a multiplexer built into the ATD module. The digitized signal is presented in 8 and or 10 bit direct code without a sign. Two additional bits of the result presentation code allow you to increase the sensitivity of the ADC from 19.53 to 4.88 mV.

Pulse Width Modulation (PWM) module. Pulse width modulation (PWM) is one of the methods of generating a pulse signal with adjustable time characteristics. The pulse width modulation method is often used to control the rotational speed of DC motors, as well as to control other types of electric motors. The hardware of the TIM multifunction timer module can be used to generate a PWM signal in the B32 MCU. However, the B32 MCU is equipped with a special PWM module. This module can generate four independent pulse sequences with 8-bit resolution for setting the duty cycle, or two pulse sequences with a 16-bit duty cycle. A combination of these modes is possible. For example, the PWM signal is used to control the steering motor in RC toy cars. In order for the machine to turn to the right or left, it must receive a pulse signal in which the pulse repetition frequency is constant, and the pulse duration changes, as shown in Fig. 4.2. The ratio of the pulse width to the duration of the signal period is called the duty cycle. The machine will turn to the left if the duty cycle is less than 50%, or to the right if the duty cycle is greater than 50%. The PWM module of the B32 microcontroller allows you to organize a pulse sequence with the required values

the repetition period and fill factor using the minimum number of instructions of the application.

The CAN controller module of the msCAN12 interface (Motorola Scalable Controller Area Network) contains a set of hardware to support the communication protocol of industrial networks in the CAN 2.0 A/B standard.

Task

Programming the Analog To Digital conversion system (ATD) module. Before starting the practical part, you should familiarize yourself with the laboratory installation based on the Freescale board and the CodeWarrior software. In order to use the ATD analog-to-digital converter module to measure the voltage level on several analog inputs of the MCU, you need to perform the following steps:

1) Connect the stabilized voltage sources to the VRF and VRL reference inputs. It should be remembered that the voltage at the input of the VRF high reference voltage should not exceed 5.0 V, and at the input of the low VRL voltage should be at least 0 V. In addition, the voltage difference at the VRF and VRL inputs equal to the full-scale voltage of the ADC $UREF = URH - URL$ should not be less than 2.5 V;

2) Connect metered analog signal sources to the AN0 inputs... AN7.
The voltage of the measured signals should be in the range from 0 V to 5.0 V;

3) Perform internal switching of the supply voltage to the ATD module.
To do this, write 1 to the ADPU bit of the ATDCTL2 control register. Register address – \$0062;

4) Hold a pause of 100 μ s to complete transients in the ATD module. In the program fragment discussed below, we will show how to organize such a delay;

5) Assign the operating mode of the ATD module by writing the necessary initialization words to the module's control registers;

6) Start the measurement sequence by writing to the ATDCTL5 control register;

7) Monitor the conversion progress using the ATDSTAT module's status register flags;

8) When the measurement sequence is complete, read the data from the ADR0H... ADR7H result registers into the MCU memory.

The voltmeter.c software fragment measures the analog signal at the AN6 input. The measurement sequence consists of four transformations. The measurement mode is one-time. The results of four consecutive conversions of the same signal are located in the four ADR0H result registers... ADR3H. These measurements are averaged, which reduces the impact of noise. The resulting 8-bit binary code is converted to the true value of the measured voltage, multiplied by 100. Multiplication by the normalization factor (100 in the decimal number system) is necessary to use integer data formats in the program. The result contains one decimal digit of the integer part of the measured voltage, these are units of Volts. As well as two digits of the decimal fractional part. These are tenths and hundredths of a Volt in the representation of the result. Intermediate results of the program execution, as well as the measured voltage, are displayed on the screen of a personal computer.

```
/* _____ */
/* filename: voltmeter.c */
```

```

/* In this example, the simplest digital voltmeter is implemented. */
/* The program performs one measurement and displays the data on the screen */
/* of a personal computer. In order to perform the next measurement*/
/* the program should be restarted */
/* _____ */
/*plug files*/
#include<912b32.h>
#include<stdio.h>
#define DECIMAL 0x2E /*define the code of the point on the
screen*/ #define V 0x56 /*define the code of the symbol "V" for
the screen*/
/*used functions*/ void
delay_100us(void); void
ADC_convert(void); void
delay_5ms (void);
void main(void)
{
printf("HELLO\n"); /*display the greeting on the screen*/
ATDCTL2 = 0x80; /*turn on the power supply of the module,
prohibit */
/*interrupts from the module*/
printf("ADC \n");
delay_100us(); /*100µs delay*/ printf
("warmed up\n");
ATDCTL3 = 0x00; /*provide access to the module in debug mode*/ ATDCTL4 =
0x01; /*set 2 cycles for sample time*/
/*and a division factor of 4*/ printf(
"ready\n" );
ADC_convert(); /*implement a digital voltmeter*/
}
/* _____ */

```

Control questions

- 1) The purpose of the processor core of microcontrollers.
- 2) The main hardware of modern MCUs.
- 3) Additional hardware of modern MCUs.
- 4) Features of microcontrollers on the example of Motorola MCUs.
- 5) Multifunctional timer.
- 6) The CAN controller module of the msCAN12 interface (Motorola Scalable Controller Area Network).
- 7) Modules controllers sequential interfaces SPI (Serial Peripheral Interface) и SCI (Serial Communication Interface).
- 8) Analog To Digital conversion system (ATD) module.
- 9) Pulse Width Modulation (PWM) module.

Practical Work No2

Structure of the processor core of a microcontroller

The purpose of the work is to familiarize with the internal structure of modern microcontrollers. Study of the main functional blocks on the example of the Motorola family of MCUs.

General information

All MCUs have a processor core. The family includes a number of models. Individual models within the family differ in a set of peripheral modules that are connected to the internal intermodule backbone. The main differences between individual representatives of the family are the type and volume of resident memory placed on the chip, the number of parallel ports and serial interface controllers. The structure of modern microcontrollers on the example of MCU MC68HC912B32 is shown in Fig. 2.

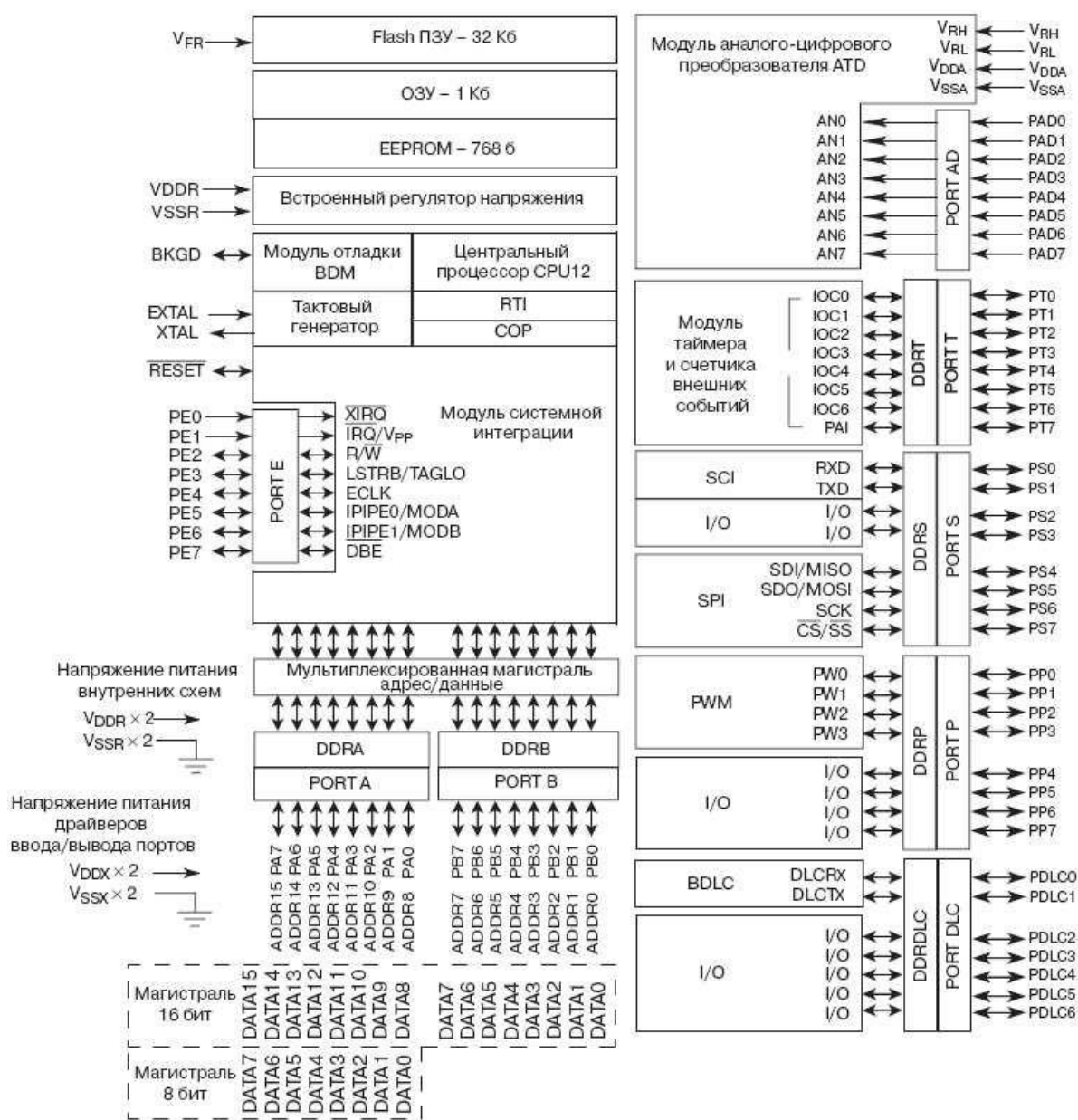


Fig.2. Structure of modern microcontrollers

Differences between modern microcontrollers on the example of Motorola MC

Low power consumption. Microcontrollers of the 68HC12 family are manufactured on the basis of CMOS technology (CMOS - Complementary Metaloxide semiconductor). This technology allows you to create transistor structures with relatively low energy losses when operating in key modes. Therefore, MCs of the 68HC12 family are characterized by low power consumption, which makes it possible to recommend them for use in products with autonomous power supply (from batteries or batteries). However, it should not be forgotten that the energy consumption for semiconductor CMOS structures is directly proportional to the switching frequency. Therefore, in order to achieve optimal energy characteristics, it is necessary to choose the clock speed of the CPU of 68HC12 microcontrollers that is the minimum possible for a particular application. 68HC12 microcontrollers have special modes of low power consumption, which also allow you to optimize the energy characteristics of the designed product.

High-performance 16-bit processor core. The 68HC12 family CPU performs operations on 16-bit operands, so the execution time of algorithms on variables larger than a byte is significantly reduced compared to 8-bit MCUs. The maximum fBUS clock rate for the 68HC12 family of microcontrollers is 8 MHz. However, the frequency of the external oscillator or crystal cavity must be 16 MHz because the microcontroller's internal circuits divide this frequency by two to produce fBUS.

The new HCS12 MCU family inherits the architecture of the processor core and most peripheral modules from its predecessor, the 68HC12 family. The main differences are as follows:

- 1) The supply voltage of most HCS12 MCU models is 5.0 V, which allows for electromagnetic compatibility in automotive, mobile and general industrial applications;
- 2) Increased CPU performance. The CPU and inter-module backbones of the HCS12 family are clocked at 25 MHz versus 8 MHz for the HC12;
- 3) The amount of resident memory has been increased. The volume of the built-in RAM in the HCS12 family MCU reaches 12 KB, the volume of Flash ROM is 512 KB. In addition, most MCU models have a significant EEPROM area (up to 4 KB) for storing reprogrammable user constants;
- 4) A large number of various serial interface controllers integrated on-chip, as the MCUs of the family are designed to work as intelligent nodes of distributed control systems.

Microcontroller Function Blocks

The HCS12 family combines a number of MCU models with the same HCS12 CPU core. Individual members of the family differ in the amount of built-in memory and the number and type of peripheral modules integrated on the MCU chip. However, each MCU from the HCS12 family contains the following functional modules:

- 1) There are three types of memory: FLASH program memory, non-volatile EEPROM memory for storing variable user constants, and static RAM for placing intermediate variables of the control application;
- 2) Multi-function 16-digit timer with 8 IC/OC outputs; 3) Multi-channel analog-to-digital converter;

- 4) Multi-standard serial exchange controllers;
- 5) A general-purpose PWM module, a number of models are equipped with a specialized PWM module to control stand-alone gate converters.

Modes of operation of microcontrollers

Microcontrollers of the 68HC12/HCS12 family operate in one of eight modes, which are divided into two groups: operating modes and special modes. Operating modes allow you to create a different hardware implementation of the embedded controller, while special modes of operation are designed for testing and diagnosing MCUs during production. Therefore, it is important for microcontroller application engineers to study only a group of operating modes. The MCU operating mode is assigned by a combination of logic signals at the inputs of the microcontroller BKGD, MODB, MODA in the MCU initial start state. The initial start state is also called the Reset state. Immediately after exiting the reset state, the MCU remembers the code combination on the listed inputs and switches to the appropriate mode of operation. It also specifies the alternative functions of the PORT A and PORT B port lines, which they acquire in each of the operating modes.

In Majority projected devices You you will Use MK
68HC12/HCS12 in one of three operating modes:

- SoC or offline mode;
- Advanced mode with 16-bit external bus;
- Advanced mode with 8-bit external bus.

Advanced modes of operation provide the ability to connect external memory and external peripherals to the MCU using parallel address and data trunks. Trunk signals are generated on the PORTA and PORTB port lines, so it is no longer possible to use the corresponding MCU pins as general-purpose I/O lines in advanced modes of operation.

Brief description of operating modes

The single-chip mode of operation (BKGD: 1, MODB: 0, MODA: 0) ensures the operation of the MCU using only the internal memory. Therefore, the codes of the control application program and its variables should be placed only in the internal ROM and RAM of the MCU.

The PORTA and PORTB ports are used as normal bidirectional I/O ports. External peripheral ICs must be connected using serial interfaces or with software support for a time swap chart on the I/O port lines.

Extended mode with 16-bit system bus (BKGD: 1, MODB: 1, MODA: 1) allows the MCU to function using both internal and external memory. A 16-bit multiplexed backbone is designed for connecting external memory

address/data ADDR15_0/DATA15_0. In this case, the high byte of the time-multiplexed ADDR15_8/DATA15_8 trunk is formed on the PORTA lines, the low byte ADDR7_0/DATA7_0 is formed on the PORTB lines.

Advanced mode with 8 bit system bus (BKGD: 1, MODB: 0, MODA: 1)

also implements the operation of the MCU using internal and external memory. But for

External memory connections are designed for a 16-bit ADDR15_0 address trunk and an 8-bit DATA7_0 data trunk. The high byte of the ADDR15_8 address trunk is output to PORTA, and the low byte of the ADDR7_0 is output to PORTB. The bidirectional 8-bit data trunk DATA7_0 uses the PORTA port lines in multiplexed mode with the most significant bits of the address trunk. In both extended modes, some PORTE port lines are used to transmit bus traffic control signals.

Task

Programming the PWM (Pulse Width Modulation) module. Before starting the practical part, you need to familiarize yourself with the laboratory installation based on the Freescale board and the CodeWarrior software. Let's consider the sequence of actions that need to be done to initialize the PWM module:

- 1) For a particular application, it is necessary to determine the resolution of the generated PWM signal, i.e. the number of discrete readings of the channel clock frequency in the period and the pulse duration of the channel output signal. Based on the data obtained, it is necessary to determine in which mode, 8_bit or 16_bit, you will use the channels of the PWM module;
- 2) For a particular application, the required frequency of the generated PWM signal should be determined. Based on the data obtained, determine the structure of the channel clock subsystem of the PWM module;
- 3) Set the bits CON23 and CON01 in the PWCLK register to select 8 times the row or 16-bit mode of operation;
- 4) Determine whether you will use centered mode or front-end PWM mode. Set the CENTR bit in the PWCTL register according to the selected mode;
- 5) Determine the active PWM signal level, according to the selection set the PPOL0... PPOL3 bits in the PWPOL register;
- 6) Assign clock sources for channels, for which to set bits PCLK0... PCLK3 в регистре PWPOL;
- 7) Set division coefficients for pulse sequences CLOCK A and CLOC K_B using the PCKA2 bits... PCKA0 and PCKB2... PCKB0 регистра PWCLK;
- 8) Set the values of the period registers and fill factor registers for the channels to be used;
- 9) Allow selected PWM channels to work using the PWEN0... PWEN3 регистра EPWM.

Let's define the settings of the PWM module to generate a PWM signal with a frequency of 976 Hz and a duty cycle of 66.7%. Let's choose the clock frequency of the PWM channel equal to $8 \text{ MHz} / 32 = 250 \text{ kHz}$. We will clock the 8_bit channel counter with this frequency. To generate a frequency of 976 Hz, 256 samples of a frequency of 250 kHz will be required, which corresponds to the maximum possible count factor of an 8_bit channel period counter. To generate a signal with a duty cycle of 66.7%, set the period code to 256 samples, and the duty cycle code to 171 counts. To generate a PWM signal, we will use channel 0 of the PWM module. The init_pwm.c program fragment performs the initial setup of the PWM module's special function registers to generate a 976 Hz PWM signal with a frequency of 66.7% at the PP0 output:

```

/* _____ */
/* The init_pwm function sets the initial settings of the PWM module */
/* _____ */
void ini_pwm(void)
{
    PWTST = 0x00; /*select the normal mode of operation of the PWM
module*/ PWCTL = 0x00; /*select the front PWM mode*/
    PWCLK = 0x28 /*channel 0 in 8_bit mode, coeff. divisions*/
/* frequency E_CLO CC is 32*/
    PWPOL = 0x01; /*set high active signal level*/ DDRP = 0xFF; /*adjust the
P port to the output*/
    PWEN = 0x01; /*allow PWM channel 0 */ PWPER0 = 255; /*set
period code*/
    PWDY0 = 171 /*set duty cycle code*/
}
/* _____ */

```

Control questions

- 1) Internal structure of microcontrollers.
- 2) Differences between modern MCs.
- 3) Functional blocks of MK.
- 4) Modes of operation of the MCU.
- 5) Single-chip operation.
- 6) Advanced mode with 16-bit system bus.
- 7) Advanced mode with 8-bit system bus.
- 8) Purpose module pulse-width Modulation).

modulator PWM (Pulse Width

Practical Work No3

Microcontroller program and data memory

The purpose of the work is to get acquainted with the types of microcontroller memory: data memory, program memory. Study of registers and stack memory of MCUs.

General information

The MCU memory subsystem of the 68HC12/HCS12 family includes four different memory modules: non-volatile Flash program memory, non-volatile EEPROM data memory, static RAM and a block of special function registers for controlling the operating modes of peripheral modules. The location of various memory modules in the MCU address space is usually reflected on the so-called memory card. The MK B32 memory card is shown in Fig. below. The addresses specified on it will be valid when the MCU exits the reset state. During the execution of the application program, the address space for each memory module can be changed. In this case, the changed virtual addresses will be used to refer to the memory cells not specified in Fig. physical addresses.

Program memory and microcontroller data memory on the example of Motorola MCUs

The B32 microcontroller is designed to be used mainly in single-chip mode of operation. It contains 32 KB of program ROM, 768 bytes of EEPROM memory, 1 KB of static RAM and 512 control registers. Strictly speaking, the abbreviation EEPROM (Electrically Erasable Programmable ROM) denotes non-volatile Memory c electric Programming and electrical erasure. Therefore, both FLASH-based resident program memory and non-volatile data memory should be characterized as EEPROM. However, non-volatile program memory and non-volatile data memory differ in their properties not only at the level of manufacturing technology, but also at the level of the developer of embedded systems. Flash-type memory allows performing an erase operation only on a certain set of cells, which is inconvenient if only one byte of information needs to be replaced. Non-volatile data memory allows you to erase and then program one byte of information. However, cells In memory c similar property occupy significant area of the semiconductor chip of the MK, so they cannot be used to create high-capacity program memory. To avoid confusion, it is customary for Russian developers to use the abbreviation EEPROM only for memory with byte-by-byte erasing and byte-by-byte programming. And memory with erasing in blocks is designated as Flash, although this memory belongs to the EEPROM by its properties. When using MCUs from Freescale Semiconductor, in particular the 68HC12/HCS12 family, you should know that the guaranteed number of erase/programming cycles for MC HCS12 is 10000, and for MC 68HC12 – only 100. That is why in many debugging tools based on MC 68HC12 it is recommended to write and execute intermediate versions of the program from RAM. Since the resident RAM of microcontrollers has insufficient space to accommodate the program, many debugging platforms use an extended mode of MCU operation with the connection of external RAM. In the HCS12 family of MCUs, internal Flash ROM programs are usually used for debugging purposes, since 10000 rewrite cycles are usually left to make all the corrections during the debugging process.

Application example

In the course of operation, EEPROM memory is often used to create counters of emergency situations at the facility. Possible accidents are preliminarily classified, sensors are installed in the system, which allow the microcontroller to attribute the emergency situation to one type or another. If the type of accident is diagnosed, the MC increases the corresponding counter and remembers its new state in the non-volatile data memory of the EEPROM type. This solution allows you to save information about accidents even when the power system is turned off.

\$0000	Регистры специальных функций
\$01FF	
\$0800	ОЗУ 1 Кб
\$0BFF	● Код/данные пользователя (\$0800-\$09FF) ● Резервная область для D-Bug12 (\$0A00-\$0BFF)
\$0D00	EEPROM 768 байт
\$0FFF	● Код/данные пользователя
\$8000	FLASH ПЗУ программ 32 Кб
\$FFFF	● код D-Bug12 (\$8000-\$F67F) ● Область, доступная пользователю (\$F6C0-\$F6FF) ● Настройка функций D-Bug12 (\$F680-\$F6BF) ● Код запуска D-Bug12 (\$F700-\$F77F) ● Вектора прерывания (\$F780-\$F7FF) ● Расширение загрузчика (\$F800-\$FBFF) ● Загрузчик EEPROM (\$FC00-\$FFBF) ● Вектора сброса и прерывания (\$FFC0-\$FFFF)

Microcontroller memory card

Special Function Registers

The block of registers of special functions of the microcontroller is very similar to the control panel in terms of the functions implemented. Each register controls or displays the state of the peripheral module of the MCU to which it is assigned in the technical description. Some of the bits of the special function registers can be associated with switches: setting a bit to 1 allows the implementation of a peripheral module function, resetting to 0 stops the execution of this function. The other part of the register bits is similar to the status indicators: setting 1 indicates the completion of the assigned function, staying at 0 indicates that the process has not yet been completed. The MC special function registers of the 68HC12/HCS12 family are combined into a block that is located in the MCU memory address space. Each of the 512 registers of the block is accessed by the same CPU instructions as the memory cells. There is no difference in terms of software maintenance between the special function registers and the RAM cells. The only exceptions are configuration registers and other, so-called protected registers, the state of which cannot be changed

at an arbitrary moment of execution of the application program. At the time of the initial start, when the MCU is in the reset state, the value specified in the technical description is written to each register of special functions. The initial states of the registers of special functions are usually such that the functions of the corresponding peripheral module are disabled. This is very convenient both from the point of view of energy consumption by the microcontroller and from the point of view of the programmer who compiles the control application. If most of the peripheral modules do not work at startup, then the MCU consumes minimal energy. Further, only those modules that are required to execute the control algorithm will be activated. The rest of the modules will remain unused, which will minimize energy consumption during the operation of the end device. When compiling a control program, the MCU first appears to the developer in a minimal configuration, when most peripheral modules do not seem to exist. Such a MCU is easy to manage, so there will be fewer errors in the application program. As the control program is implemented, the developer activates the operation of only those modules that he needs. Therefore, he does not have to think about software maintenance of peripheral modules that are not involved in the implementation of the algorithm. Each register of special functions has its own address in the memory card of the MK. When programming in assembler, changing the values of special function registers and reading their state can be done by commands to load and send data, such as LDAA, STAA, MOV, as well as by commands of bit operations BSET, BCLR. In order to save the programmer from remembering the physical addresses of the registers of special functions, it is necessary to use the pseudo-command of the EQU assignment operation of the Assembly language. This pseudo-command will assign an absolute address to the symbolic name of the register, which is the same for all models of the MC family 68HC12/HCS12. And when writing the source code of the program, it will be possible to use only the symbolic names of the registers of special functions. The use of symbolic names is also useful when transferring an application program from one MCU model to another. In this case, only the absolute addresses in the pseudo-instructions should be replaced, while the source code of the program will remain unchanged.

Task

Before starting the practical part, it is necessary to familiarize yourself with the laboratory installation based on the Freescale board and the CodeWarrior software. The practical part of this work considers the programming of the ADC module registers.

The ATDCTL2 register, located in the MCU memory at address 0x0062, controls the operation of the ATD analog_digital converter module. Bit 7 of this register allows (1) or prohibits (0) the operation of the module. This bit, designated in the datasheet as ATPU (ATD Powerup Bit), is set to 0 in the MCU reset state. To put the ATD module into operation, you need to programmatically set this bit to 1. In assembly language, this action can be performed with the following program fragment:

```
; _____
; MAIN PROGRAM
; _____
ATDCTL2 EQU $0062 ; determine the address of the register in the MC
ATD_INI EQU $80 ; determine the meaning of a word
; initialization to enable the ADC
LDAA #ATD_INI ; Load the initialization word into the ACC
STAA ATDCTL2 Write the initialization value to
```

```
; ADC Control Register
```

```
;
```

The first two lines of the program contain assembly language EQU directives, which inform the program that the value \$0062 should be used instead of the ATDCTL2 name in the program codes, and \$80 or 10000000 in binary code instead of ATD_INI. The LDAA command uses direct addressing, and as a result of its execution, the number \$80 will be loaded into the ACC battery. The direct-addressing STAA command sends the ACC contents to the cell with the ATDCTL2 address. As a result, the code in the ATDCTL2 control register will be equal to 10000000, i.e. set to 1 bit 7 will cause the ADC module to turn on. In C, the following program fragment performs a similar action:

```
/* _____ */
/*MAIN PROGRAM
/* _____ */
#include<912b32.h>
void main(void)
{
unsigned char ATD_INI = 0x80;
ATDCTL2 = ATD_INI;
}
/* _____ */
```

Compare the benefits of writing C and assembly language programs.

Control questions

- 1) Memory structure of microcontrollers.
- 2) Program memory.
- 3) Data memory.
- 4) Volatile memory.
- 5) Non-volatile memory.
- 6) Registers of special functions of MC.
- 7) ADC module control registers.
- 8) Stack memory.
- 9) Advantages of programming in C and Assembler.

Practical Work No4

I/O Ports

The purpose of the work is to get acquainted with the I/O device of microcontrollers. Study of port specification using the example of the Motorola MCU.

General information

All MCUs in the 68HC12 family have a number of data I/O lines. The lines are combined into 8-bit parallel data ports: Port A, Port B, Port E, PORT AD, etc.

With rare exceptions, all I/O lines are bidirectional. The transmission direction of I/O lines is configured in software by writing a control word to the transmission direction register of the corresponding port. It is possible to change the transmission direction during program execution by reprogramming these registers. The reset signal sets all bidirectional lines to input mode. It should be emphasized that the transmission direction of each line can be chosen arbitrarily by the designer, regardless of the other lines belonging to the same I/O port. The only exceptions are unidirectional transmission lines, which are initially specialized for input or output. Some of the I/O lines have a so-called alternative function, i.e. they provide communication between the built-in peripheral modules of the MCU and the "outside world". For example, the PORT AD port lines are used to connect the measured voltages to the built-in ADC, the PORT S port lines serve as inputs and outputs of serial exchange controllers. If the corresponding MCU peripheral module is not used, then its pins can be used as regular I/O lines. If the port lines are bidirectional, then two types of registers are provided for its service to such a port:

PORTx is the data register of port x, where x is the name of the I/O port;

DDRx is the transfer direction register of port x.

For example, the PORT A port is served by the PORTA and DDRA registers, and the PORT B port is served by the PORTB and DDRB registers. If the port has circuitry with a software-connected pull-up resistor (R pull_up), then an additional port input impedance register is provided to serve such a port. Below is a fragment of the text of a program that configures PORT T for data output, and then writes the number S62 to the port. In order for all the lines of the PORT T port to become output lines, you must write the code to the DDRT transmission direction register \$FF.

I/O Port Specification

The parallel I/O subsystem of the MCU B32 consists of 8 ports, with many of the ports having an alternative function. We will consider these ports in the order in which they are located in Fig. 4.1. clockwise.

1)PORT A. In single-chip mode, an 8-bit general-purpose I/O port. The transmission direction of each port line is determined by the corresponding bit of the DDRA register. In extended mode, port lines generate high-byte signals of the multiplexed address/data backbone ADDR15_8/DATA15_8. In extended mode with an 8-bit bus, the port lines are a multiplexed ADDR15_8/DATA7_0 backbone.

2) PORT B. In single-chip mode, an 8-bit general-purpose I/O port. The transmission direction of each port line is determined by the corresponding bit of the DDRB register. In extended mode, the least-byte signals of the multiplexed address/data backbone ADDR7_0/DATA7_0 are generated on the port lines. In extended mode, with an 8-bit bus, the port lines are a non-multiplexed ADDR7_0 backbone.

3) PORT E. 8-bit general-purpose I/O port. Two port lines PE0 and PE1 are unidirectional and only for input. The remaining port lines are bidirectional, the transmission direction of lines PE2 and PE7 is determined by the corresponding bits of the DDRE register. All port lines have an alternative function. Lines PE1 and PE0 can be used as inputs for external IRQ and XIRQ interrupts. The remaining lines are used to set the MCU operating modes (MOD A and MOD B) and as external bus control signals when the MCU is operating in extended mode.

4) PORT AD. Unidirectional 8-bit input port. All lines have an alternate function. If the ATD A/D converter module is enabled, the port lines are used to connect the analog signals being measured.

5) PORT T: A bidirectional, 8-bit, general-purpose I/O port. The transmission direction of each port line is determined by the corresponding DDRT register bit. An alternate function of the PORT T port lines is to service the timer module. If the timer is enabled, the lines are used as IC input capture inputs or OC output comparison outputs.

6) PORT S: A bidirectional, 8-bit, general-purpose I/O port. The transmission direction of each port line is determined by the corresponding bit of the DDRS register. An alternative function of the PORT S port lines is to serve the SCI and SPI serial exchange modules.

7) PORT P: A bidirectional 8-bit general-purpose I/O port. The transmission direction of each port line is determined by the corresponding bit of the DDRP register. The four lines of the PORT P port can be used as outputs of the PWM signal generator module (PWM module) if the PWM module is allowed to operate.

8) PORT DLC. Bidirectional 8-bit general-purpose I/O port. The transmission direction of each port line is determined by the corresponding bit of the DDRDLC register. An alternative feature of the two lines of the PORT DLC port is to service the serial exchange module in the BDL standard.

Port Control Registers

In the 68HC12/HCS12 family of MCUs, each bidirectional general-purpose I/O port is served by two special function registers. These are the Port Data Register and the DDRx Transmit Direction Register (the "x" should be replaced by the port letter). If the lines of the General Purpose port are configured for input, the data register read returns the status of the MCU chassis pins to which the port is associated. If the port is configured for output, the data register write sets the port-associated MCU pins to the appropriate logic levels.

The DDRx register determines the transmission directions of each port line independently of the other lines on the same port. If any bit of the DDRx register is 0, then the corresponding line is configured for input, if 1, then for output. Solutions are possible in which some of the lines of the same port are configured for input and some for output. For example, if the value is DDRx=10110010 the lines D6, D3, D2, and D0 are deployed for input, and

the D7, D5, D4, and D1 lines are on the pin. In the MCU reset state, all DDRx transfer direction register bits are reset, so immediately after powering up, all MCU port lines are configured as high input impedance inputs.

Task

Before starting the practical part, you should familiarize yourself with the laboratory installation based on the Freescale board and the CodeWarrior software.

In the practical part, we will consider the operation of the PORTA port. To do this, connect a group of green and red LEDs to the pins of this port. The connection diagram is shown in Fig. 3 below. Let's agree that if 1 is set on the port line, then the green LED will be lit, if the logic 0 is set, then the red LED. And if the port line is switched to the input state, i.e. it is represented by a load with a high input impedance for the LED circuit, then both LEDs will be canceled. Fig. 4 shows a block diagram of the program algorithm that lights up LEDs with even numbers in green for 30 ms and LEDs with odd numbers in red at the same time. The next 30 ms of LEDs "change colors", then this process continues indefinitely. Below is the text of the program in C that implements this algorithm.

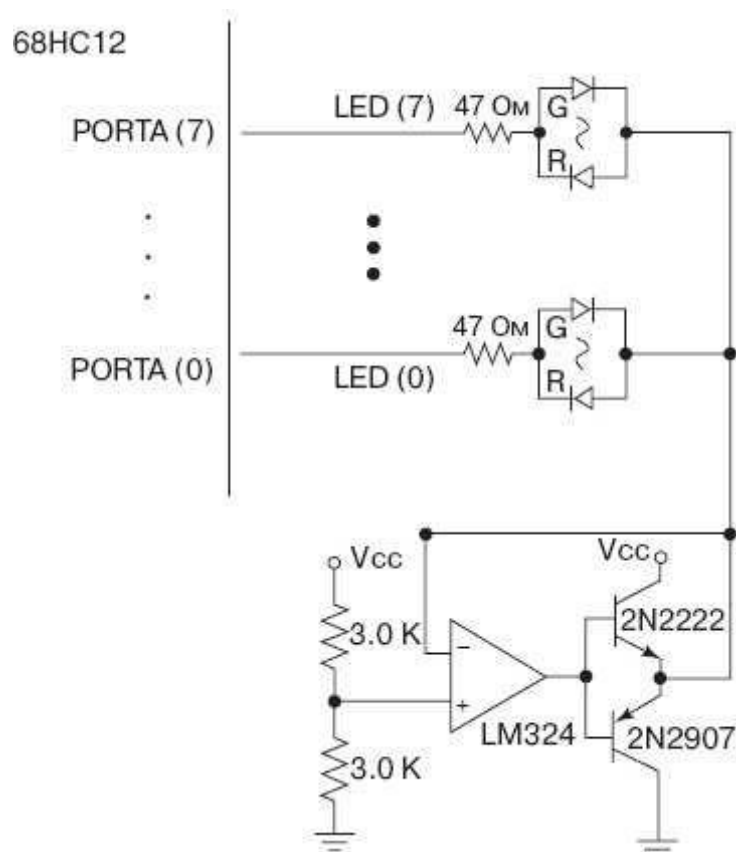


Fig.3. Scheme for displaying the state of the PORTA port



Fig.4. Block diagram of the LED control algorithm

```

/* _____ */
/* MAIN PROGRAM: This program "lights up" on the outputs of the PORTA port */
/*30 ms LEDs on port outputs with even */
/*numbers, red – LEDs on port outputs with odd */
/* numbers. For the next 30 ms, the green ones are replaced by the red ones, and vice versa */
/* _____ */
/*plug files*/
#include<912b32.h>
/*used functions*/ void
delay_100us(void)
void delay_30ms(void)
void main{void)
{
DDRA = 0xFF //set PORTA port to pin
while(1)
{
PORT = 0x55;
delay_30ms(); DOOR
= 0xAA;

```

```

delay_30ms();
}
}
/* _____ */
/* Function delay_30ms generates a delay of 30ms  $\mu$ s, clock rate*/
/* intermodule trunks MC is 8 MHz */
/* _____ */
void delay_30ms(void)
{
int i;
for (i=0; i<=299; i++)
delay_100us();
}
/* _____ */
/* Function delay_100us generates a delay of 100  $\mu$ s, clock rate */
/* intermodule trunks MC is 8 MHz */
/* _____ */
void delay_100us(void)
{
int j;
for (j=0; j<50; j++)
{
asm(«nop»\n) ;
}
}
/* _____ */

```

Note that the 30 ms delay function uses a nested 100 μ s delay function. In the above program text, 50 repetition cycles of delay_100us function operators are used to generate a 100 μ s delay. The number of repetitions was selected on the assumption that this program would be executed by a microcontroller with an internal bus frequency of 8 MHz. Based on the results of the examination of the *.lst file, it was found that assembly commands corresponding to one repetition of the cycle of the delay_100us function are implemented in 16 machine cycles. At a bus frequency of 8 MHz, 800 machine cycles are required to form a time interval of 100 μ s. Therefore, the number of repetitions of the cycle of the delay_100us function should be $800/16 = 50$. If this program were executed by the DP256 microcontroller, which has an internal bus frequency of 25 MHz, then the number of cycles of the function delay_100us would have to be increased to 156.

Control questions

- 1) Purpose of information input/output devices.
- 2) Port types and information exchange algorithms.
- 3) Direction of port work.
- 4) Port specification.
- 5) Special port management registers.
- 6) Additional port control registers.
- 7) Schema for implementing port line health checks.
- 8) I/O port programming.
- 9) A software implementation of time delay in C and Assembly.

Practical Work No5

Hardware to ensure reliable MC operation

Goal Jobs – Introduction c states reset and interruptions
of microcontrollers during the execution of the control program.

General information

In the process of executing the application program, the MC implements a monotonous repeatedly repeated sequence of actions:

- Retrieving the instruction code from program memory to the CPU's instruction register;
- Decryption of the command code;
- Retrieving the following instruction bytes from memory;
- Execution of the command;
- Storage of the results of the command execution in memory.

If a linear sequence of instructions is executed, the contents of the CPU's PC counter are constantly increased by 1, allowing the next instructions of the application program to be retrieved from memory. The linear sequence of executable instructions can be changed under the control of the program itself, for example, by the "jmp" or "branch" instructions. In this case, a new number will be written to the instruction counter under the control of the program, and the execution of the next linear fragment of the program from another segment of the program's memory will begin. Despite the obvious differences in the mechanisms for generating the value of the PC instruction counter following the execution of the current operation, in both cases this next PC value is determined by the course of the computing process and is predicted by the programmer in the course of writing the application program.

In contrast to the completely predictable flow of events just considered, which is formed by the microprocessor system itself, there is also a stream of external events, the sequence and moments of occurrence of which are not synchronized with the execution of certain instructions of the application program by the central processor. However, the MCU must respond to these events, for which it is necessary to change the sequence of execution of program operators at an arbitrary moment in time, unpredictable from the point of view of the central processor control device. Such a change is realized by forcing a new value to be written to the PC instruction counter under the control of special microcontroller hardware that react to external events. Activation of the mechanism of forced change of the current value of the instruction counter cannot be considered an emergency state of the microprocessor system. This is only a special state that allows you to organize the effective distribution of the resource of one central processor to serve several devices that generate unrelated external events in real time. According to the method of processing by the microcontroller, exceptions are divided into interrupts and reset. In Russian-language literature, the term "exception" is not usually used, and they simply refer to the interrupt state or the microcontroller reset state (translator's note).

Reset and interrupt states in Motorola family microcontrollers

MCUs of the 68HC12/HCS12 family have a powerful exception handling system. According to the way the microcontroller reacts to perturbing events, exceptions are divided into interrupt and reset.

- The MCU implements an interrupt or the MCU is in an interrupt state;
- The MCU is in the reset state or in the initial start state. Interrupts, in turn, are divided into masked and non-maskable.

Microcontroller Reset Status

The microcontroller of the 68HC12/HCS12 family enters the reset state on an external signal or when certain internal events occur. In the reset state, the program counter and some of the bits of the CPU status register, as well as the registers of special functions of peripheral modules defined in the technical description, are set to the initial state. This state uniquely determines the hardware configuration of the microcontroller, from which it will start operating after power is turned on. Therefore, the second name for the reset state is the initial start state. The MCU also uses the reset state to restore the working state after detecting an internal emergency.

There are four event sources that put the MC in the reset state:

1) External reset. All MCUs in the 68HC12/HCS12 family have a dedicated RESET housing pin to provide an external reset signal. The active signal level is logic 0. As long as the RESET input is kept low, the MCU will remain in the reset state. After the RESET line is set to logic 1, the MCU will enter the active mode when the delay of 4096 is over periods of the MC system highway.

2) Internal Power on reset (POR). The voltage increase at the input of the VDD microcontroller causes a reset state. Thus, the initial start of the MCU is implemented with a uniquely defined hardware configuration and with a known start address of the program to be executed.

3) Computer Operating Properly Reset (COP). The watchdog logic allows the microcontroller to detect intermittent errors in the execution of the application that may occur as a result of electromagnetic interference or fluctuations in the supply voltage of the microprocessor system. During debugging, the watchdog is not allowed to operate. The watchdog module is allowed to operate in the final version of the application that is used when the MCU is running in the system. A watchdog timer is a counter whose count factor is configured by the user when the system is initialized. The counter starts counting internal clock pulses when the program starts executing. If the counter is overfilled, the MCU will enter the reset state. A properly executed application program in which the order of execution of statements coincides with the order specified by the programmer should constantly reset the watchdog timer. In this case, it will not cause an internal reset. To reset the watchdog timer in the MC of the 68HC12/HCS12 family, it is necessary to write the code \$55 in the COPRST register, and then the code \$AA. When creating the final code of the application program, the developer must place the operations of writing the given sequence of codes so that the execution of the program on any

would ensure that reset commands are executed at shorter intervals than the watchdog overflow period.

4) Internal reset by deviation of the frequency of clock pulses (Clock Monitor reset). The MCU is switched to a reset state when the module of the built-in clock generator detects that the clock frequency has gone beyond the specified limits or simply stopped the clock system.

Microcontroller interrupt status

Non-maskable interrupts.

As its name suggests, non-maskable interrupts cannot be disabled by the user. However, it was mentioned in the previous paragraph that setting bit X to 1 prohibits non-maskable interrupts. The value of bit X is indeed 1 in the MC reset state. However, it can then be set to 0 under the control of the initialization program, thereby allowing non-maskable interrupts. This bit cannot be further changed under program control, and this is in contrast to the global interrupt mask bit I.

Three types of non-maskable interrupts are implemented in MC 68HC12/HCS12:

1) External XIRQ Interrupt. All 68HC12/HCS12 MCUs have an external non-maskable XIRQ interrupt output. The active signal level for generating an interrupt request is logic 0.

2) Interrupt by a non-existent instruction code. Each instruction of the MCU assembly language has its own code. In MC 68HC12/HCS12, opcodes can be one-byte and double-byte. However, not all theoretically possible codes are used to encode real instructions of the CPU12 processor core. If a non-existent instruction code is read from memory at the stage of retrieving the instruction code from memory, a request for a non-maskable interrupt is generated.

3) Program Interrupt – SWI instruction. The MC 68HC12/HCS12 instruction set has a software interrupt instruction that allows you to jump to the interrupt routine from the application.

Masked interrupts.

1) Interrupt by external IRQ. All MCUs 68HC12/HCS12 have an external masked IRQ output. The active signal level for generating an interrupt request is logic 0. In some applications, requests from multiple external signal sources must be accepted. For such cases, an additional logic member should be used that combines requests from all sources using logic OR. If an IRQ input request is received and the MCU proceeds to execute an interrupt routine, the subroutine should poll the port lines to determine which source caused the interrupt. Processing multiple OR combined queries with a programmatic search of the source that established the request is called polling.

2) RTI Real-Time Cue Timer Interrupt. The RTI Real-Time Cue Timer generates a sequence of equidistant interrupt requests. The query recurrence period is configured by the programmer. These interrupts can be used for the microcontroller to perform a task on a regular basis. For example, to measure voltage battery every three minutes to signal when it needs to be replaced. We will look at the specifics of RTI interrupts in Chapter 7 using the example of controlling the speed of an electric motor.

3) Event-based interrupt of the timer's capture/compare channel (IC/OC). Eight identical blocks in the timer module, which are referred to as "channels",

are designed to control the signal level at the input of a channel or to change the logical level of the output of a channel at a specific point in time. A programmer-specified change in the input or output signal of a channel is considered an event that generates an interrupt request. For example, if a channel is configured to monitor the input signal from 1 to 0, then when such a change occurs, an interrupt request will be issued.

4) Interrupt by timer overflow. The main block of the timer module is the 16_bit counter of the time base. This counter cannot be stopped. It is also impossible to change its count coefficient, which is $2^{16} = 65536$. Therefore, the time base counter regularly changes its code from \$FFFF to \$0000. Such a code change is called counter overflow. At the moment of overflow, at the request of the programmer, an interrupt request can be generated, while the counter continues to count further. Such interrupts are especially convenient when it is necessary to measure very large time intervals. To do this, the microcontroller counts how many counter overflows occurred during this time interval, and, knowing the counter counting period, determines the duration of the time interval under study.

5) Interrupt by external event counter overflow. When the external event counter is overflowing, an interrupt request can be generated in the same way as for the temporary base counter.

6) Interrupt by event at the input of the external event counter. This interrupt request is generated if the signal at the input of the external event counter has changed its value. The nature of the change, i.e. the difference from 0 to 1, or from 1 to 0, or any change in the logical level, is determined by the programmer.

7) Interrupt from SCI and SPI serial I/O controller modules. Each serial I/O module forms a whole group of interrupt requests: when a word transmission is completed, when a word is received, when various types of violations are detected in the information transfer protocol.

8) Interrupt from the ADC module. The analog-to-digital converter module generates an interrupt request when the process of digitizing the next signal is completed, and the binary signal code can be read into the MCU memory.

9) Interrupt when the MCU exits power-saving modes. This interrupt allows the MCU to be removed from the STOP or WAIT state in which it was in order to reduce power consumption. Such interrupts are very useful when combining multiple MCUs into an information network. We will discuss this type of interrupt in more detail in the next paragraph.

Task

Before starting the practical part, you should familiarize yourself with the laboratory installation based on the Freescale board and the CodeWarrior software.

In this practical work, we will consider the main features of the formation of the source code of a program with interrupts in C.

When programming in C, you must implement the following stages of writing the source text:

1. When writing an interrupt handling program in C, the name of the interrupt handling routine must be declared using a special preprocessor directive.

```
#pragma interrupt_handler <name>
```

In the <name> field, you should write the name of the interrupt routine that you will use in the program text. This entry informs the compiler that the function with the named name is an interrupt routine.

2. Further in the text, the interrupt routine is formalized as a regular function. The compiler, in the process of translating the source code of this function into C in the assembler instruction, will automatically substitute the return command from the RTI interrupt at the end of the subroutine, because this function was declared by the interrupt routine (the #pragma directive in step 1).

3. For the MCU to function properly during an interrupt, it is necessary to initialize the stack pointer. Its value must be equal to the highest address of the RAM area of the MCU increased by one. Since user memory ranges (both permanent and RAM) are necessary settings in the compiler configuration, the function of initializing the stack pointer is performed by the compiler automatically. Therefore, the programmer should not write any text in the program to initialize the stack pointer. Instead, it is necessary to check the memory card in the compiler settings, which must necessarily coincide with the actual system being designed.

4. The interrupt subsystem will function correctly if an interrupt vector table is formed for it. We have already discussed that the interrupt vector table in MCU B32 is located in a Flash memory area that is protected from overwriting information. In order for the user to be able to write his own table of reset and interrupt vectors, fixed vectors are written to this indelible memory area, which transfer control to known addresses in the area of the EEPROM memory being rewritable. At these addresses, the programmer must enter the JMP unconditional jump command with the address of the corresponding interrupt processing routine.

5. Each masked interrupt request source must be resolved by setting the appropriate bit in the peripheral module's control register. We'll look at how to write this to C in subsequent examples.

6. After setting all the individual bits to allow the interrupt, you need to reset the global interrupt mask I. In assembler, the CLI command is used for this. When programming in C, we will also use this command through the following macros:

```
#define CLI ( ) asm(«cli\n»); Allow masked interrupts
#define SEI ( ) asm(«sei\n»); Disallow masked interrupts
```

Further in the text of the program, if you need to allow interrupts, then you should enter CLI ().

Declaring a Function in a Module

```
void toggle_isr(void);
```

Directive #pragma to indicate that a function is a subroutine
interrupt service

```
#pragma interrupt_handler toggle_isr
```

initializing the corresponding vector in the interrupt vector table

```
#pragma abs_address: 0xF7EA //B32 RAM_based vector address
```

```
void (*Timer_Channel_2_interrupt_vector[]) ()={toggle_isr};
```

```
#pragma end_abs_address
```

Control questions

- 1) The sequence of actions of the MC when executing the executable program.
- 2) MC reset states.
- 3) MC interrupt states.
- 4) Masked and non-maskable interrupts.
- 5) Exception vectors.
- 6) Priority system for exceptions.
- 7) Interrupt subsystem registers.
- 8) The process of transition to the interrupt system.
- 9) Software implementation of reset systems.
- 10) Software implementation of interrupt systems.

References

Reference citations:

1. Tanenbaum Andrew S. Computer Architecture. 5th ed. (+CD) . St. Petersburg, Piter Publ., 2007. 848 p. (in Russian).
2. Broydo V.L., Ilyina O.P. Arkhitektura EVM i sistem: Uchebnik dlya vuzov [Architecture of computers and systems: Textbook for universities]. 2nd ed. – St. Petersburg, Piter Publ., 2009. 720 p. (in Russian).
3. Novikov Yu.V., Skorobogatov P.K. Fundamentals of Microprocessor Technology – Internet University of Information Technologies, INTUIT.ru, 2006, 440 p. Free Internet access: <http://www.intuit.ru/department/hardware/mpbasics/>

Further reading:

4. Shevkoplyas B.V. Microprocessor Structures. Engineering Solutions: Reference Book. Ed. 2nd, revised and supplemented — Moscow: Radio and Communication, 1990 — 512 p.: ill.
5. Novikov Yu.V. Functional modules of control and measurement systems based on microcomputers: Microprocessor means and systems. — 1990.—N 3.— pp. 75—77.
6. Novikov Yu.V., Kalashnikov O.A., Gulyaev S.E., Ed. by Yu.V. Novikov. Development of interface devices for personal computers of the IBM PC type. Manual. Moscow, EKOM Publ., 1997. 224 p.: ill.
7. Predko M. Guide to microcontrollers. Volume 1. Moscow, Postmarket Publ., 2001. — 416 S.: ill.
8. Barrett S.F., Park D.J. Embedded systems. Design of applications on MCs of the 68HC12/HCS12 family using the C language. — Moscow: DMK - Press, 2007. — 640 p.

Control questions

- 1) The sequence of actions of the MC when executing the executable program.
- 2) MC reset states.
- 3) MC interrupt states.
- 4) Masked and non-maskable interrupts.
- 5) Exception vectors.
- 6) Priority system for exceptions.
- 7) Interrupt subsystem registers.
- 8) The process of transition to the interrupt system.
- 9) Software implementation of reset systems.
- 10) Software implementation of interrupt systems.

MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE
RUSSIAN FEDERATION

FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF HIGHER EDUCATION
"NORTH CAUCASUS FEDERAL UNIVERSITY"

SUGGESTIONS FOR TEACHING

for independent work of students in the discipline

"Principle and Application of MCU"

for students of the field of study

11.03.02 Infocommunication Technologies and Communication Systems,
Profile "Intelligent data processing in
Information and Communication Systems and Networks"

Stavropol, 2026

Methodical instructions for the organization of students' independent work are compiled in accordance with the requirements of the discipline program

"Principle and Application of MCU" for bachelors of the field of study 11.03.02 Infocommunication Technologies and Communication Systems. The textbook is devoted to the planned educational work of students, performed in extracurricular (classroom) time on the assignment and under the methodological guidance of the teacher, but without his direct participation, with the partial, direct participation of the teacher.

CONTENTS

1. Independent work as the most important form of the educational process
2. Goals and main objectives of the ACS
3. Types of independent work
4. Organisation of SSDs
5. General recommendations for the organization of independent work
6. Independent work of a student is a necessary link in the formation of a researcher
7. Methodical recommendations for students on certain forms of independent work
8. Educational, methodological and information support of the discipline

1. Independent work as the most important form of the educational process

Independent work is the planned educational, research, research work of students, carried out in extracurricular (classroom) time on the assignment and under the methodological guidance of the teacher, but without his direct participation (with the partial direct participation of the teacher, leaving the leading role for the work of students).

It is necessary to familiarize yourself with the technological map of the student's independent work.

№	Section (topic) of the discipline and summary	Emerging competencies, indicators	Full-time				Current Forms control Academic Performance
			Contact work of students with the teacher / of them in the form of practical training, hours			Independent work, hours	
			Lectures	Practical exercises	Laboratory work		
1	Topic 1. Features of programming microprocessor systems. Levels of representation of a microprocessor system. Program model of a microprocessor. Microcontroller architecture. Typical composition of microcontrollers. Low-level programming. Practical work: Structure of microcontrollers	ID-1 OPK-4 ID-2 OPK-4 ID-3 OPK-4 ID-4 OPK-4 ID-5 OPK-4	-	4	-	6	Interview

2	Topic 2. Software development tools for embedded microprocessor systems. Purpose and classification of software development tools (IDE). Composition of a typical IDE. Comparative analysis of Arduino IDE, Microchip Studio, Keil uVision, STM32 Cube IDE. Domestic software development tools. Creation of a block diagram of algorithms for the functioning of developed software products and software documentation according to the USPD using IDE Practical work: Structure of the processor core of a microcontroller	ID-1 OPK-4 ID-2 OPK-4 ID-3 OPK-4 ID-4 OPK-4 ID-5 OPK-4	-	6	-	6	Interview
3	Topic 3. Organization of the computing process in real-time systems. Organization of the computing process using dispatching. Organization of interrupt processing. FreeRTOS real-time operating system. Organization of the computing process using RTOS Practical work: Structure of the processor core of a microcontroller	ID-1 OPK-4 ID-2 OPK-4 ID-3 OPK-4 ID-4 OPK-4 ID-5 OPK-4	-	8	-	6	Interview
4	Topic 4. Programming of drivers of microprocessor devices. Universal synchronous-asynchronous transceiver (USART). Implementation of serial information exchange using USART. Programming of RS485, RS232 interface drivers based on the ModBus protocol. Development of drivers for data exchange using proprietary protocols Practical work: Memory of programs and data microcontroller	ID-1 OPK-4 ID-2 OPK-4 ID-3 OPK-4 ID-4 OPK-4 ID-5 OPK-4	-	8	-	6	Interview

5	Topic 5. Programming of tasks related to the generation and processing of time parameters. Timers and event processors and their implementation in the structure of microcontrollers. Measurement of time intervals. Real-time clock. Built-in time measurement functions. Generation of pulse sequences with specified durations and repetition periods. Counting of events related to external signals and program results. Practical work: Memory of programs and data microcontroller	ID-1 OPK-4 ID-2 OPK-4 ID-3 OPK-4 ID-4 OPK-4 ID-5 OPK-4	-	8	-	6	Interview
6	Programming tasks related to analog signal processing. Devices, algorithms, accuracy of analog-to-digital conversions. Comparison of analog quantities. Digital-to-analog converters. PWM-Based Analog Signal Generation Hands-On: I/O Ports	ID-1 OPK-4 ID-2 OPK-4 ID-3 OPK-4 ID-4 OPK-4 ID-5 OPK-4	-	8	-	6	Interview
7	Topic 7. Complex debugging of software for microprocessor systems. Debug boards and debugging tools. Emulators. The JTAG interface and its use for software verification of a microprocessor device. In-circuit programming using the JTAG interface. Programming of microcontrollers using the SPI interface. Estimation of the computational complexity of the developed software products Practical work: Hardware ensuring reliable operation of the MC	ID-1 OPK-4 ID-2 OPK-4 ID-3 OPK-4 ID-4 OPK-4 ID-5 OPK-4	-	8	-	6	Interview

8	Topic 8. Programmable logic controllers Purpose of programmable logic controllers. Aries programmable logic controllers. General characteristics of PLC programming languages. PLC graphical programming languages. PLC text-based programming languages. PLC integrated programming environments Practical work: Hardware ensuring reliable operation of the MC	ID-1 OPK-4 ID-2 OPK-4 ID-3 OPK-4 ID-4 OPK-4 ID-5 OPK-4	-	8	-	6	Interview
	Exam preparation					54	
	TOTAL		-	42	-	48	

Independent work of students at the university is an important type of educational and scientific activity of the student. Independent work of students plays a significant role in the rating technology of education. Education at the university includes two parts that are almost the same in volume and mutual influence - the process of learning and the process of self-learning. Therefore, the SRS should become an effective and purposeful work of the student.

The concept of modernization of Russian education defines the main tasks of vocational education - "training a qualified worker of the appropriate level and profile, competitive in the labor market, competent, responsible, fluent in his profession and oriented in related fields of activity, capable of effective work in his specialty at the level of world standards, ready for constant professional growth, social and professional mobility".

The solution of these problems is impossible without increasing the role of students' independent work on educational material, strengthening the responsibility of teachers for the development of independent work skills, for stimulating professional growth of students, fostering creative activity and initiative.

Society imposes a fairly wide list of requirements for a modern specialist, among which it is important that graduates have certain abilities and ability to independently obtain knowledge from various sources, systematize the information received, and assess a specific financial situation. The formation of such a skill occurs during the entire period of study through the participation of students in laboratory classes, the performance of control tasks and tests, the writing of term papers and graduation qualification papers. At the same time, the independent work of students plays a decisive role in the entire educational process.

The forms of independent work of students are diverse. In this discipline "Programming of microprocessor systems" the following are provided:

The use of the material of the educational and methodological complex of disci- Plin

At the first stage, it is necessary to familiarize yourself with the mandatory minimum content of the main educational program for the discipline studied. Next, it is necessary to study the work program of the discipline, which considers the content of the topics of the discipline of the lecture course, the relationship between the topics of lectures and laboratory classes, topics and types of independent work. For each type of independent work, certain forms of reporting are provided in the curriculum for the discipline

"Programming of microprocessor systems".

Work with literature

For the successful development of the discipline, it is necessary to independently study the presented topics according to the recommended sources of information that are presented in the curriculum.

Independent work introduces students to scientific creativity, the search for and solution of urgent modern problems.

2. Goals and main objectives of the ACS

The leading goal of the organization and implementation of the SRS should coincide with the goal of the student's education - the preparation of a bachelor with higher education. In the organization of the SRS, an important and necessary condition is the formation of the ability to work independently to acquire knowledge, skills and the possibility of organizing educational and scientific activities.

The purpose of students' independent work is to master fundamental knowledge, professional skills and abilities in the field, experience in creative and research activities. Independent work of students contributes to the development of independence, responsibility and organization, a creative approach to solving problems of the educational and professional level.

The objectives of the SRS are:

- systematization and consolidation of the theoretical knowledge and practical skills of students;
- deepening and expanding theoretical knowledge;
- formation of skills to use regulatory, legal, reference documentation and special literature;
- development of cognitive abilities and activity of students: creative initiative, independence, responsibility and organization;

- formation of independent thinking, abilities for self-development, self-improvement and self-realization;
- development of research skills;
- the use of material collected and obtained in the course of independent studies at seminars, laboratory classes, when writing course and final qualification papers, for effective preparation for final tests and exams.

3. Types of independent work

In the educational process of a higher professional educational institution, two types of independent work are distinguished: classroom, under the guidance of a teacher, and extracurricular. The close relationship between these types of work provides for the differentiation and effectiveness of the results of its implementation and depends on the organization, content, logic of the educational process (interdisciplinary connections, promising knowledge, etc.):

Classroom independent work on the discipline is carried out in classes under the direct guidance of the teacher and on his assignment.

Extracurricular independent work is carried out by the student on the instructions of the teacher, but without his direct participation.

The main types of independent work of students without the participation of a teacher in this discipline are:

- formation and assimilation of the content of lecture notes on the basis of educational literature recommended by the lecturer, including information educational resources (electronic textbooks, electronic libraries, etc.);
- preparation for laboratory work, their design.

The main types of independent work of students with the participation of teachers are:

- ongoing consultations;
- acceptance and defense of laboratory work (during the L/R).
- Implementation of the course work.

4. Organisation of SSDs

The methodology of organizing students' independent work depends on the structure, nature and features of the discipline "Programming of Digital Signal Processors", the number of hours for its study, the type of tasks for students' independent work, the individual qualities of students and the conditions of educational activity.

The process of organizing students' independent work includes the following stages:

- preparatory (definition of goals, drawing up a program, preparation of methodological support, preparation of equipment);
- the main one (implementation of the program, the use of methods of searching for information, assimilation, processing, application, transfer of knowledge, fixing the results, self-organization of the work process);
- final (assessment of the significance and analysis of the results, their systematization, assessment of the effectiveness of the program and methods of work, conclusions about the directions of labor optimization).

Students' Activities for the Formation and Development of Skills of Educational Independent Work

In the process of independent work, the student acquires the skills of self-organization, self-control, self-management, self-reflection and becomes an active independent subject of educational activity.

Performing independent work under the supervision of a teacher, the student must:
master the minimum content submitted for independent work of students and proposed by the teacher in accordance with the educational standards of higher education;

☐ plan independent work in accordance with the schedule of independent work proposed by the teacher;

☐ the student should carry out independent work in the organizational forms provided for by the curriculum and work program of the teacher;

Perform independent work and report on its results in accordance with the schedule for submitting results, types and deadlines for reporting on students' independent work.

A student can:

in addition to the one proposed by the teacher (with justification and agreement with him) and the minimum mandatory content determined by the program for this discipline:

– independently determine the level (depth) of the study of the content of the material;

☐ suggest topics and questions for self-study;

– Offer a reasonable, individual timetable for independent work and reporting on the results of independent work as part of the overall timetable;

– offer their own options for organizational forms of independent work;

☐ to use for independent work methodological manuals, teaching aids,

developments in addition to the list proposed by the teacher;

□to use not only control, but also self-control of the results of independent work in accordance with the methods of self-control proposed by the teacher or chosen independently.

Independent work of students should have an important impact on the formation of the personality of the future bachelor, it is planned by the student independently. Each student independently determines the mode of his work and the degree of work spent on mastering the educational content in each discipline. He performs extracurricular work according to a personal individual plan, depending on his training, time and other conditions.

5. General recommendations for the organization of independent work

The main form of independent work of a student is the study of lecture notes, their supplementation, recommended literature, active participation in the in laboratory classes. But for successful educational activity, its intensification, it is necessary to take into account the following subjective factors:

1. Knowledge of school program material, the presence of a solid system of gaps necessary for mastering the main university courses. This is especially important for mathematical disciplines. It is necessary to distinguish gaps in knowledge that make it difficult to assimilate new material from small abilities.

2. By spending effort to overcome these gaps, the student will ensure normal academic performance and believe in his abilities.

3. Availability of skills, skills of mental work:

a) the ability to take notes at lectures and when working with a book;

b) mastery of logical operations: comparison, analysis, synthesis, generalization, definition of concepts, rules of systematization and classification.

4. The specifics of cognitive mental processes: attention, memory, speech, observation, intelligence and thinking. Poor development of each of them becomes a serious obstacle in learning.

5. Good working capacity, which is ensured by a normal physical condition. After all, serious learning is a large, versatile and diverse work. The result of learning is evaluated not by the amount of information communicated, but by the quality of its assimilation, the ability to use it and the development of one's ability to further independent education.

6. Correspondence of the chosen activity, profession to individual abilities. It is necessary to develop the ability to self-regulate one's emotional state and eliminate circumstances that disturb the business mood, interfere with the planned work.

7. Mastering the optimal style of work that ensures success in activity. Alternating work and breaks in work, rest periods, individually justified sleep duration,

preference for evening or morning classes, stress resistance to exams and features of preparation for them,

8. The level of requirements for oneself, determined by the existing self-esteem. Adequate assessment of knowledge, advantages, disadvantages is an important component of the self-organization of a person, without it it is impossible to successfully manage one's behavior and activity.

One of the main features of education in higher school is that constant external control is replaced by self-control, and the active role in learning belongs not so much to the teacher as to the student. Knowing the basic methods of scientific organization of mental work, it is possible to achieve the best results with the least expenditure of time, money and labor efforts. The effectiveness of assimilation of incoming information depends on the ability of a person to work at a particular moment of his activity.

Working capacity is the ability of a person to work with a high degree of intensity for a certain time. There are internal and external factors of working capacity.

Internal factors of working capacity include intellectual characteristics, will, and state of health.

External:

- organization of the workplace, work and rest schedule;
- the level of labor organization - the ability to obtain information and use information;

- the amount of mental load.

The outstanding Russian physiologist N. E. Vvedensky identified the following conditions for the productivity of mental activity:

- every work must be entered into gradually;
- dimensionality and rhythm of work. Different people have a more or less different pace of work;
- the usual sequence and systematicity of activity;
- correct alternation of work and rest.

Rest does not necessarily imply complete inaction on the part of a person, it can be achieved by a simple change of business. During the day, the ability to work changes. The most fruitful time is the *morning time (from 8 to 14 hours)*, and the maximum working capacity falls on the period from 10 to 13 hours, then *the afternoon (from 16 to 19 hours)* and *the evening (from 20 to 24 hours)*. It is better to study very difficult material at the beginning of each period of time (preferably in the morning) after a good rest. After 1-1.5 hours, breaks of 10 - 15 minutes are needed, after 3 - 4 hours of work, the rest should be long - about an hour.

An integral part of the scientific organization of mental labor is the mastery of the technique of mental labor.

The time available to the student to complete the curriculum consists of two components: one of them is classroom work at the university according to the schedule of classes, the other is extracurricular independent work. Tasks and materials for independent work are given during training sessions according to the schedule, in the same classes the teacher controls independent work, and also assists students in the correct organization of work.

Self-study will require intensive mental work, which must not only be properly organized, but also stimulated. At the same time, it is very important to be able to maintain a steady attention to the material being studied. The development of attention requires significant volitional efforts. That is why, if a student notices that he is often distracted during self-study, he needs to force himself to concentrate. Such a procedure must be done constantly, as this is attention training. Sustained attention appears when a person treats the matter with interest.

It is necessary to properly organize your classes in time: 50 minutes - work, 5-10 minutes - break; after 3 hours of work, a break - 20-25 minutes. Otherwise, the growing fatigue will lead to instability of attention. A very important

Systematic physical education is a factor influencing the increase in mental performance. The organization of active recreation provides for the alternation of mental and physical activity, which fully restores a person's working capacity.

6. Independent work of a student is a necessary link in the formation of Researcher's Work

The progress of science and technology, information technology leads to a significant increase in scientific information, which imposes higher requirements not only on the moral and ethical qualities of a person, but also, in particular, constantly increasing requirements in the field of education - renewal, modernization of general and professional knowledge, skills of a specialist.

Any education should act as a dynamic process that exists in a person and continues throughout his life. Mastering the scientific thought and language of science is a necessary component in the self-organization of a future research specialist. This is understood not so much as the accumulation of knowledge, but the mastery of scientifically based ways of acquiring it. This, generally speaking, is the main task of the university.

The specificity of the university educational process, in the organization of which more and more space is given to the independent work of the student, is that it seems to be the last and most adequate link for the implementation of this task. For during the study at the university, the development of the style and skills of educational (cognitive) activity takes place, the rational nature of which will contribute to the constant renewal of the knowledge of a highly qualified university graduate.

However, there are certain difficulties before this path, in particular, the transition of the student from the synthetic process of learning in secondary school to the analytical one in higher school. This is due to both the new content of education (expansion of general education and deepening of professional training) and new, hitherto unknown forms of learning (lectures, seminars, laboratory classes, etc.). The student not only acquires the knowledge provided by the program and teaching aids, but he must also become acquainted with the ways of acquiring knowledge in such a way as to be able to assess what we know, how we know it and how we have achieved this knowledge. All this is achieved through one's own independent work.

This is also because self-acquired knowledge is more operational, it becomes personal property, as well as a motive for behavior, develops intellectual traits, attention, observation, criticality, and the ability to evaluate. The role of the teacher is mainly to help accumulate knowledge (in relation to first-year students), and in the subsequent years of study, in the senior years, to jointly identify problems and take care of the student's independent search, as well as to control their activities. It should be noted that it is not possible to limit oneself only to the acquisition of knowledge provided for by the program of the discipline being studied, it is necessary to constantly deepen the knowledge gained, focusing it on some narrow defined area that corresponds to the interests of the student. In-depth study of all subjects provided for by the program is possible in practice, and good organization of work saves time, which creates conditions for a deep, systematic, interested study of the topic independently chosen by the student.

Of course, all the advice, examples, recommendations in this area given by the teacher, or certain publications, or other sources, do not guarantee any success without the manifestation of one's own activity in this matter, i.e. they do not give ready-made recipes, but should contribute to the analysis of one's own work, its goals, and organization in accordance with individual characteristics. Taking into account personal capabilities, existing living and working conditions, skills, on the basis of these recommendations, it is possible to develop an individually justified set of methods, ways, find one's own style or improve it, in order to study certain material, have time to assess its significance, suitability and possibilities of its application, in order to ultimately ensure the success of one's studies from future professional activity

7. Methodological recommendations for students on individual forms of independent work

The system of higher education implies a much greater independence of students in planning and organizing their activities.

Work with literature

When working with literature, it is necessary to select literature, learn how to read it correctly, and keep notes. For the selection of literature in the library, alphabetical and systematic catalogs are used.

It is important to remember that rational skills of working with a book are always a great saving of time and effort.

The correct selection of textbooks is recommended by the teacher who reads the lecture course. The necessary literature can also be indicated in the methodological developments for this course.

When studying the material from the textbook, one should move on to the next question only after the previous one has been correctly understood, describing on paper all the calculations and calculations (including those that are omitted in the textbook or given in the lecture for independent conclusion).

In the study of any discipline, independent individual work plays a large and important role.

Particular attention should be paid to the definition of the basic concepts of the course. The student should analyze in detail the examples that explain such definitions, and be able to build similar examples independently. It is necessary to achieve an accurate idea of what you are studying. It is useful to make reference notes. When studying the material from the textbook, it is useful to supplement the lecture notes in the notebook (in specially designated margins). There you should also note the questions selected by the student for consultation with the teacher.

It is recommended to highlight the conclusions obtained as a result of the study in the notes so that they are better remembered when rereading the notes.

Experience shows that many students are helped by compiling a list of reference signals containing the most important and most frequently used formulas and concepts. Such a sheet helps to memorize formulas, the main provisions of the lecture, and can also serve as a permanent reference for the student. There are two types of reading: primary and secondary. *Primary reading* is attentive, unhurried reading, in which you can dwell on difficult places. may not always be understood after the initial reading.

The task of *secondary* reading is the complete assimilation of the meaning of the whole (this reading may not be the second, but the third or fourth).

Reading a scientific text is a part of cognitive activity. Its purpose is to extract the necessary information from the text. The effectiveness of the action largely depends on how conscious the reader is of his own internal attitude when referring to the word (to find the necessary information, to assimilate the information in whole or in part, to critically analyze the material, etc.).

There are four main attitudes in reading a scientific text:

1. information retrieval (the task is to find, highlight the information sought)

2. assimilative (the reader's efforts are aimed at understanding and remembering as fully as possible both the information presented by the author and the entire logic of his reasoning)

3. analytical-critical (the reader seeks to critically comprehend the material, analyzing it, determining his attitude to it)

4. creative (creates in the reader a readiness in one form or another – as a starting point for his reasoning, as an image for action by analogy, etc. – to use the author's judgments, the course of his thoughts, the result of observation, the developed methodology, to supplement them, to subject them to a new check).

The existence of several types of reading *is associated with the presence of different attitudes to addressing a scientific text:*

1. bibliographic – viewing catalog cards, recommendation lists, summary lists of journals and articles for the year, etc.;

2. browsing – used to search for materials containing the necessary information, usually it is used immediately after working with references and catalogs, as a result of such viewing the reader establishes which of the sources will be used in further work;

3. introductory – implies a continuous, sufficiently detailed reading of the selected articles, chapters, individual pages, the purpose is to get acquainted with the nature of the information, to find out what issues are submitted by the author for consideration, to sort the material;

4. Studying – involves a thorough mastery of the material; In the course of such reading, the reader's trust in the author, readiness to accept the information presented, the attitude to the most complete understanding of the material is realized;

5. Analytical-critical and creative reading are two types of reading that are close to each other in that they participate in solving research problems. The first of them involves a directed critical analysis of both the information itself and the ways in which it is obtained and presented by the author; the second is the search for those judgments, facts on which or in connection with which the reader considers it necessary to express his own thoughts.

Of all the types of reading considered, the main one for students is the learning one - it allows them to accumulate knowledge in various fields when working with educational literature. That is why this type of reading within the framework of educational activity should be mastered in the first place. In addition, when mastering this type of reading, the main techniques are formed that increase the effectiveness of working with a scientific text.

The main types of systematized recording of what has been read:

1. Annotation is an extremely brief coherent description of the book (article) viewed or read, its contents, sources, nature and purpose;
2. Planning is a brief logical organization of the text, revealing the content and structure of the material being studied;
3. Thesis is a concise reproduction of the author's main statements without involving factual material;
4. Citation is a verbatim copying from the text of excerpts, extracts that most significantly reflect a particular idea of the author;
5. Note-taking is a brief and consistent presentation of the content of what has been read.

A synopsis is a complex way of presenting the content of a book or article in a logical sequence. A synopsis accumulates the previous types of notes, allows you to comprehensively cover the content of a book or article. Therefore, the ability to make a plan, theses, make extracts and other notes determines the technology of compiling a note.

Methodical recommendations for compiling a synopsis:

1. Read the text carefully. Clarify the incomprehensible words in the reference literature. When writing down, do not forget to put the reference data in the margins of the note.
2. Highlight the main thing, make a plan.
3. Briefly formulate the main points of the text, note the author's description.
4. Take notes of the material, clearly following the points of the plan. When taking notes, try to express the idea in your own words. Notes should be kept clear and clear.
5. Write down quotes correctly. When quoting, take into account the conciseness and significance of the thought.

In the text of the synopsis, it is desirable to give not only thesis statements, but also their proofs. When drawing up a synopsis, it is necessary to strive for the capacity of each sentence. The thoughts of the author of the book should be stated briefly, taking care of the style and expressiveness of the writing. The number of additional elements of the synopsis should be logically justified, the notes should be distributed in a certain sequence that corresponds to the logical structure of the work. Margins should be left for clarification and addition.

Mastering the skills of taking notes requires dedication and daily independent work from the student.

Practical exercises

It should be emphasized that only after mastering the lecture material from a certain point of view (namely, from the point of view from which it is presented at the lectures) it will be consolidated in the laboratory classes both as a result of discussion and analysis of the lecture material, and with the help of completing the tasks of the laboratory work, the tasks set. Under these conditions, the student will not only learn the material well, but also learn how to apply it in practice, and will also receive an additional incentive (and this is very important) for active study of the lecture.

When performing tasks independently, it is necessary to justify each stage of the solution, based on the theoretical provisions of the course. If the student sees several ways to solve a problem (task), then it is necessary to compare them and choose the most rational one. It is useful to make a brief plan for solving the problem (task) before starting calculations. The solution of problems or examples should be presented in fractions, calculations should be arranged in a strict order, separating auxiliary calculations from the main ones. Solutions, if necessary, should be accompanied by comments, diagrams, drawings and drawings.

It should be remembered that the performance of each educational task should be carried out before the final logical answer required by the condition, and, if possible, with a conclusion. The performance of tasks of this type should be continued until the acquisition of solid skills in solving them.

Self-Test

After studying a certain topic from the notes in the notes and textbook, as well as solving a sufficient number of relevant tasks in laboratory classes and independently, the student is recommended, using a sheet of reference materials, to reproduce from memory definitions, conclusions of formulas, formulations of the main provisions and evidence.

If necessary, you need to carefully understand the material again.

Sometimes the insufficiency of assimilation of a particular issue becomes clear only when studying further material. In this case, it is necessary to go back and repeat the poorly mastered material. An important criterion for mastering theoretical material is the ability to solve problems or pass a test on the material covered. However, it should be remembered that the correct solution is

can be obtained as a result of the application of mechanically memorized formulas without understanding the essence of theoretical propositions.

Consulting

If in the process of independent work on the study of theoretical material or when performing tasks, the student has questions that cannot be solved independently, it is necessary to contact the teacher to obtain explanations or instructions from him. In his questions, the student should clearly express what he is experiencing difficulties, the nature of this difficulty. Advice should also be sought if there are doubts about the correctness of the answers to the self-test questions.

Preparation for the exam in the discipline

The study of many general professional and special disciplines ends with an exam. Preparation for the exam contributes to the consolidation, deepening and generalization of knowledge obtained in the process of learning, as well as its application to solving practical problems. Preparing for it, the student eliminates the existing gaps in knowledge, deepens, systematizes and organizes his knowledge.

To pass the exam, the student must fully complete the tasks of practical classes, prepare a report on each work and a report on the results of independent study of topics, according to the curriculum. If there are arrears in the current certification in this discipline, the student can get a grade.

Systematic preparation for classes during the semester will allow you to use the time of the exam session to systematize knowledge.

8. Educational, methodological and information support of the discipline

References	1. Bazhenov A. V., Bratchenko N. Yu., Grivennaya N. V. Programming of Embedded Microprocessor Systems: Textbook. – Stavropol : NCFU Publ., 2022. – 302 p., unlimited copies. 2. Novikov, Y. V.; Fundamentals of Microprocessor Technology: Textbook / Yu. - Fundamentals of Microprocessor Technology, 2022-07-28. - Electron. dan. (1 file). - Moscow : Internet University Information Technology (INTUIT), IPR
------------	---

	Media, 2020. - 405 p. - Electronic. - The book is in the premium version of EBS IPR BOOKS. - ISBN 978-5-4497-0677-5, Unlimited Instances 3. Vodovozov A. M. Microcontrollers for Automation Systems [Elektronnyi resurs] : uchebnoe posobie [Microcontrollers for Automation Systems] : textbook. — Elektron. Tekstovye dannye. — M. : Infra-Inzheneriya, 2016. — 164 c. — 978-5-9729-0138-8. — Regime Available: http://www.iprbookshop.ru/51727.html 4. Fundamentals of Programming Microprocessor Controllers in Digital Control Systems of Technological Processes [Elektronnyi resurs] : textbook / V. S. Kudryashov, A. V. Ivanov, M. V. Alekseev [i dr.]. — Elektron. tekstovye dannye [Electron. Text data]. — Voronezh : Voronezh State University of Engineering Technologies, 2014. — 144 c. — 978-5-00032-054-9. — Mode of access: http://www.iprbookshop.ru/47437.html 5. Shegal, A. A. Application of the software complex Multisim for Designing Devices on Microcontrollers [Elektronnyi resurs] : laboratornyy praktikum / A. A. Shegal ; ed. by V. I. Ievlev. — Elektron. Text Data. — Yekaterinburg : Ural Federal University, EBS ASV, 2014. — 116 c. — 978- 5-7996-1117-0. — Mode Access: http://www.iprbookshop.ru/65968.html
Further reading	1. Aliev, M.T.; Microprocessors and microprocessor control systems. 8-bit processors of the AVR family: laboratory practicum / M.T. Aliev, T.S. Bukanova ; Volga State Technological University. - Yoshkar-Ola : PSTU, 2016. - 64 p. : schemes, tabl., ill. - http://biblioclub.ru/. - Bibliogr. in the book - ISBN 978- 5-8158-1775-3, unlimited copies 2. Bezuglov, D. A. Digital devices and microprocessors. Textbook / D. A. Bezuglov, I. V. Kaliyevka. - Izd. 2-e. - Rostov-on-Don: Feniks, 2008. - 468 p. : ill. - (Higher Education). - Bibliogr.: p. 464-465. - ISBN 978-5-222-13917-2 3. Khartov V. Y. Microcontrollers AVR: Practicum for Beginners. Khartov. - Moscow, MSTU Publ., 2007. - 240 p. : ill. - Appendix: pp. 231-238. - References: p. 239. - ISBN 978-5-7038-3051-2 4. Melnikov, E. V.; Fundamentals of Microprocessor Technology: Laboratory Practicum / E. V. Melnikov. - Fundamentals of Microprocessor Technology, 2026-09-20. - Electron. dan. (1 file). - Samara : Samara State Technical University, EBS ASV, 2020. - 47 p. - Electronic.

