MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL AVIATION UNIVERSITY
Faculty of Aeronautics, Electronics and Telecommunications,
Department of Aviation Computer-Integrated Complexes

ACCEPT TO PROTECTION
Head of Department
_______Viktor SINEGLAZOV
“__”_________2023

QUALIFICATION PAPER
(EXPLANATORY NOTE)

HIGHER EDUCATION STUDY
“MASTER”

Specialty 151 "Automation and computer-integrated technologies"
Educational and professional program "Information support and engineering of aviation computer systems"

Subject: Cybersecure and invulnerable training automated workstation for equipment development based on STM series microcontrollers

Performer: student of the group I3-225M Oleh Yaremchuk
Supervisor: Associate professor, Ihor Sergeyev

Consultant of the "Environmental Protection" section__________ Radomska M.M.

Consultant of the "Occupational safety and health" section______Kazhan.K.I.

Norm control:________Fylashkin M.K.

Kyiv 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет аеронавігації, електроніки та телекомунікацій

Кафедра авіаційних комп’ютерно-інтегрованих комплексів

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____Віктор СИНЄГЛАЗОВ
“___”____________2023 р.

КВАЛІФІКАЦІЙНА РОБОТА (ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ

“МАГІСТР”

Спеціальність 151 «Автоматизація та компю’терно-інтегровані технології»
Освітньо-професійна програма «Інформаційне забезпечення та інженерія авіаційних комп’ютерних система»

Тема: Кібербезпечне та невразливе навчальне автоматизоване робоче місце для розробки обладнання на основі мікроконтролерів серії STM

Виконавець: студент групи ІЗ-225М Яремчук Олег Русланович
Керівник: доцент, Сергеєв Ігор Юрійович

Консультант розділу «Охорона навколишнього середовища»_____Радомська М.М.

Консультант розділу «Охорона праці»_________Кажан.К.І.

Нормоконтролер:____________Філяшкін М.К.

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет аеронавігації, електроніки та телекомунікацій

Кафедра авіаційних комп'ютерно-інтегрованих комплексів

Освітній ступінь: Магістр

Спеціальність 151 "Автоматизація та комп'ютерно-інтегровані технології"

ЗАТВЕРДЖУЮ
Завідувач кафедри
_________Віктор СИНЄГЛАЗОВ
“____ ” __________2023 р.

ЗАВДАННЯ
на виконання дипломної роботи студента
Яремчука Олега Руслановича

1. **Тема роботи:** “Кібербезпечне та невразливе навчальне автоматизоване робоче місце для розробки обладнання на основі мікроконтролерів серії STM”
2. **Термін виконання проекту (роботи):** з 01.12.2023 р. до 27.12.2023 р.
3. **Вихідні данні до проекту (роботи):** Лабораторні роботи для розробки обладнання на базі STM32, середовище розробки CubeIDE, Atolic Studio.
4. **Зміст пояснювальної записки (перелік питань, що підлягають розробці):**

   1. Актуальність кіберзахисту автоматизованого робочого місця; 2. Аналіз існуючих рішень для вирішення проблеми; 3. Пропозиції вирішення проблеми кібервразливості; 4. Розробка лабораторних робіт.
5. **Перелік обов'язкового графічного матеріалу:**

   1. Структурна схема підключення STM; 2. Схему входів і виходів процесора; 3. схема пінів;
6. **Календарний план-графік**

№ п/п	Завдання	Термін виконання	Відмітка про виконання
1	Отримання завдання	02.10.2023 – 03.10.2023	
2	Формування мети та основних завдань дослідження	03.10.2023 – 05.10.2023	
3	Аналіз існуючих рішень	07.10.2023 – 15.10.2023	
4	Теоретичний розгляд рішення задачі	17.10.2023 – 01.11.2023	
5	Пропозиції вирішення проблеми	01.11.2023 – 15.11.2023	
6	Розробка програмного та апаратного забезпечення	20.11.2023 – 05.12.2023	
7	Оформлення пояснювальної записки	07.12.2023 – 10.12.2023	
8	Підготовка презентації та роздаткового матеріалу	12.12.2023– 17.12.2023	

6. **Консультанти з окремих розділів**

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв
Охорона праці	Доцент Кажан К.І.		
Охорона навколишнього середовища	Доцент Радомська М.М.		

7. **Дата видачі завдання:** “02” жовтня 2023 р.

Керівник: ____ Сергеєв І.Ю.

Завдання прийняв до виконання: ____________________ Яремчук О.Р.

NATIONAL AVIATION UNIVERSITY

Faculty of aeronavigation, electronics and telecommunications

Department of Aviation Computer Integrated Complexes

Educational level: Master

Specialty 151 "Automation and computer-integrated technologies"

APPROVED

Head of Department

_________Viktor SINEGLAZOV

"____" ___________2023

TASK

For the student's thesis

Yaremchuk Oleh Ruslanovych

1. **Theme of the project:** "Cybersecure and invulnerable training automated workstation for equipment development based on STM series microcontrollers"
2. **The term of the project (work):** from December 01, 2023 until December 27, 2023
3. **Output data to the project (work):** Laboratory works for the development of equipment based on STM32, CubeIDE development environment, Atolic Studio.
4. **Contents of the explanatory note (list of questions to be developed):**
1. Relevance of cyber protection of an automated workplace; 2. Analysis of existing solutions to solve the problem; 3. Proposals for solving the problem of cyber vulnerability; 4. Development of laboratory works.
5. **List of compulsory graphic material:** 1. Structural diagram of STM connection; 2. Scheme of inputs and outputs of the processor; 3. pin scheme;

6. Planned schedule:

№ п/п	**Task**	**Execution term**	**Execution mark**
1	Task	02.10.2023 – 03.10.2023	
2	Purpose formation and describing the main research tasks	03.10.2023 – 05.10.2023	
3	Analysis of existing solutions	07.10.2023 – 15.10.2023	
4	Theoretical consideration of problem solving	17.10.2023 – 01.11.2023	
5	Suggestions for solving the problem	01.11.2023 – 15.11.2023	
6	Software and hardware development	20.11.2023 – 05.12.2023	
7	Making an explanatory note	07.12.2023 – 10.12.2023	
8	Preparation of presentation and handouts	12.12.2023– 17.12.2023	

6. Consultants from individual sections

Section	Consultant	Date, signature	
		Issued the task	Accepted the task
Occupational safety and health	Associate Professor Kazhan K.I.		
Environmental protection	Associate Professor Radomska M.M.		

7. Date of task receiving: “2” October 2023

Diploma thesis supervisor: ______________________________ Sergeyev I.Y.

Issued task accepted: ___________________________________ Yaremchuk O.R.

Table of contents

SECTION 1 STRUCTURE OF MODERN MICROCONTROLLERS...13

1.1. General data about the microcontroller…………………………….13

1.2. STM microcontrollers………………………………………………18

1.3. Cortex cores……………………………………………………23

1.4. Overview of current trends in cyber defense……………………..25

1.5. Analysis of requirements for a cyber-secure workplace………….26

1.6. Analysis of threats and vulnerabilities……………………………28

SECTION 2 ADVANTAGES OF STM MICROPROCESSORS………31

2.1. Pin-to-pin compatibility…………………………………………….31

2.2 Debug boards…………………………………………………….33

2.3 Low-cost boards……………………………………………………36

2.4 High performance cores……………………………………………38

2.5 Low power consumption……………………………………………40

2.6 Reliability………………………………………………………...41

2.7 Security……………………………………………………………..42

2.8 Bus matrix…………………………………………………………..42

2.9 Low power mode entry……………………………………………..43

2.10 Memory allocation…………………………………………………44

2.11 Components of STM32 ……………………………………………45

2.11.1 System Architecture ……………………………………………..46

2.11.2 Microcontroller Configuration …………………………………47

SECTION 3 DESIGNING A CYBER-PROTECTED WORKPLACE…51

3.1. Choice of system architecture and components……………………51

3.2. Overview of the Cube MX development environment……………..53

SECTION 4 DEVELOPMENT OF LABORATORY WORKS ON PROGRAMMING OF STM32 MICROCONTROLLERS…………….61
4.1 Laboratory work №1. Project generation and PWM work programming in conjunction with CubeMX and Atollic TrueStudio**Ошибка! Закладка не определена.**............................61
4.2. Laboratory work №2. Working with a display based on HD44780..68
SECTION 5 ENVIRONMENTAL PROTECTION…………………..82
SECTION 6 OCCUPATIONAL SAFETY AND HEALTH …………91
Conclusions…………………………………………………………..98
References…………………………………………………………..100

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи “ Кібербезпечне та невразливе навчальне автоматизоване робоче місце для розробки обладнання на основі мікроконтролерів серії STM ”   100 стор., 41 іл., 16 джерел.

КІБЕРБЕЗПЕЧНЕ ТА НЕВРАЗЛИВЕ НАВЧАЛЬНЕ АВТОМАТИЗОВАНЕ РОБОЧЕ МІСЦЕ ДЛЯ РОЗРОБКИ ОБЛАДНАННЯ НА ОСНОВІ МІКРОКОНТРОЛЕРІВ СЕРІЇ STM

Об’єктом дослідження є кібербезпечне та невразливе навчальне автоматизоване робоче місце для розробки обладнання на основі мікроконтролерів серії STM.

Предметом дослідження є методи кіберзахисту автоматизованого робочого місця.

Метою кваліфікаційної роботи є розробка та впровадження механізмів безпеки, таких як системи аутентифікації та авторизації, які забезпечують надійний захист вбудованих систем, а також розробка суміжних з цим лабораторних робіт.

Широке використання комп’ютерних систем, мереж і програмного забезпечення призвело до збільшення кількості кіберзагроз, що створює ризик як для корпоративних структур, так і для пересічних користувачів. Одночасно інтеграція вбудованих систем і мікроконтролерів у різні сфери підкреслює потребу в надійних і кіберзахищених рішеннях. Однак зростання популярності мікроконтролерів STM32 привернуло увагу соціальних інженерів і хакерів.

Ця дипломна робота спрямована на створення кіберзахищеного робочого місця на основі процесора STM32 для захисту вбудованих систем від потенційних загроз. Цей підхід передбачає аналіз вимог, розробку та впровадження заходів безпеки та тестування системи на стійкість до атак. Мета полягає в тому, щоб зробити внесок у вдосконалення техніки безпеки для вбудованих систем і підвищити безпеку та надійність пристроїв на основі мікроконтролерів STM32.

ABSTRACT

Explanatory note of the qualification work "Cybersecure and invulnerable training automated workstation for equipment development based on STM series microcontrollers " 100 pages, 41 fig., 16 sources.

CYBERSECURE AND INVULNERABLE TRAINING AUTOMATED WORKSTATION FOR EQUIPMENT DEVELOPMENT BASED ON STM SERIES MICROCONTROLLERS.

The object of the research is the cybersecure and invulnerable training automated workstation for equipment development based on STM series microcontrollers.

The subject of research is methods of cyber protection of an automated workplace..

The purpose of the qualification work is development and implementation of security mechanisms, such as authentication and authorization systems, which ensure reliable protection of embedded systems, as well as development of related laboratory work.

The widespread use of computer systems, networks and software has led to an increase in the number of cyber threats, which poses a risk to both corporate structures and ordinary users. At the same time, the integration of embedded systems and microcontrollers in various fields emphasizes the need for reliable and cyber-protected solutions. However, the rise in popularity of STM32 microcontrollers has attracted the attention of social engineers and hackers.

This thesis is aimed at creating a cyber-secure workplace based on the STM32 processor to protect embedded systems from potential threats. This approach involves analyzing requirements, developing and implementing security measures, and testing the system for resistance to attacks. The goal is to contribute to security improvements for embedded systems and improve the security and reliability of STM32 microcontroller-based devices.

INTRODUCTION

The wide distribution of computer systems, networks, and software contributes to an increase in cyber threats and attacks that can seriously harm both corporate structures and average users. At the same time, the integration of embedded systems and microcontrollers in various spheres of life encourages the development of dependable and cyber-protected solutions based on such devices. These issues of cyber protection are becoming more and more relevant and necessary at the current stage of development of information technologies and cyberspace.

STM32 microcontrollers, which are widely used in a variety of fields, including automotive, medical devices, home appliances, industrial automation, and others, are one of the promising platforms for the development of cyber-protected systems. However, as STM32 microcontrollers gain popularity, so does the interest of social engineers and hackers in potential attacks against these systems.

The goal of this thesis is to develop and implement a cyber-protected workplace based on the STM32 processor to protect embedded systems from potential threats. To accomplish this goal, the requirements for a cyber-protected workplace will be analyzed, appropriate security measures will be developed and implemented, and the system will be tested for attack resistance.

This thesis seeks to contribute to the advancement of security techniques for embedded systems, as well as to the security and dependability of devices based on STM32 microcontrollers.

Scientific novelty

1. **Development of a cyber-protected workplace based on the STM32 microcontroller:** The scientific novelty is the creation of a set of security measures and the implementation of a cyber-protected workplace based on the STM32 microcontroller, which was previously a less studied area in the context of cyber security.
2. **Analysis of threats and vulnerabilities for embedded systems:** The work includes a comprehensive analysis of threats and vulnerabilities of embedded

systems, which allows identifying potential attacks and developing protection measures.

3. **Development and implementation of security mechanisms:** The thesis describes the development and implementation of security mechanisms, such as authentication and authorization systems, which provide reliable protection of embedded systems.

Practical value

1. **Protection of embedded systems:** The workplace developed in this work can be used to protect embedded systems in a variety of industries, including automotive, medical, and industrial automation.
2. **Improving Internet of Things (IoT) security:** The developed secure mechanisms can be used to protect IoT devices, which is becoming increasingly relevant due to the growing proliferation of connected devices.
3. **Protection of important information systems:** The results of the study can be useful for organizations and enterprises that seek to increase the level of protection of their information systems and embedded devices.
4. **Creating a base for further research:** Work on this topic can serve as a platform for further research in the field of cyber protection of embedded systems and microcontrollers.

SECTION 1
STRUCTURE OF MODERN MICROCONTROLLERS

1.1 General data about microcontrollers.

Household appliances, medical devices, elevator control systems, telephones, walkie-talkies, and other communication devices, electronic musical instruments and car stereos, computer peripherals (keyboards, joysticks, printers, and so on), traffic lights, automatic gates and barriers, interactive children's toys, cars, locomotives, and airplanes, robots, and industrial machines all use microcontrollers.

In contrast to a microprocessor, a microcontroller typically has a low bit rate (8 to 16 bits) and a rich set of instructions for manipulating individual bits. Bit instructions can be used to operate discrete equipment (e.g., raise/lower the barrier, switch on/off the lamp and heater, start/stop the engine, open/close the valve, and so on).

Another significant difference between a microcontroller and a microprocessor is that the controller chip contains all of the components needed to build a basic (and sometimes rather complicated) control system. Yes, there is data memory (RAM), program memory (permanent memory), a clock generator, timers, counters, parallel and serial ports, and so on inside the microcontroller.

A typical microcontroller architecture consists of a synchronization and control system (1), an arithmetic logic device (2), general-purpose registers (3), data memory (4) and program memory (5), ports (6), functional devices (timers, counters, pulse-width modulators, interfaces) and their registers (7), and other components.

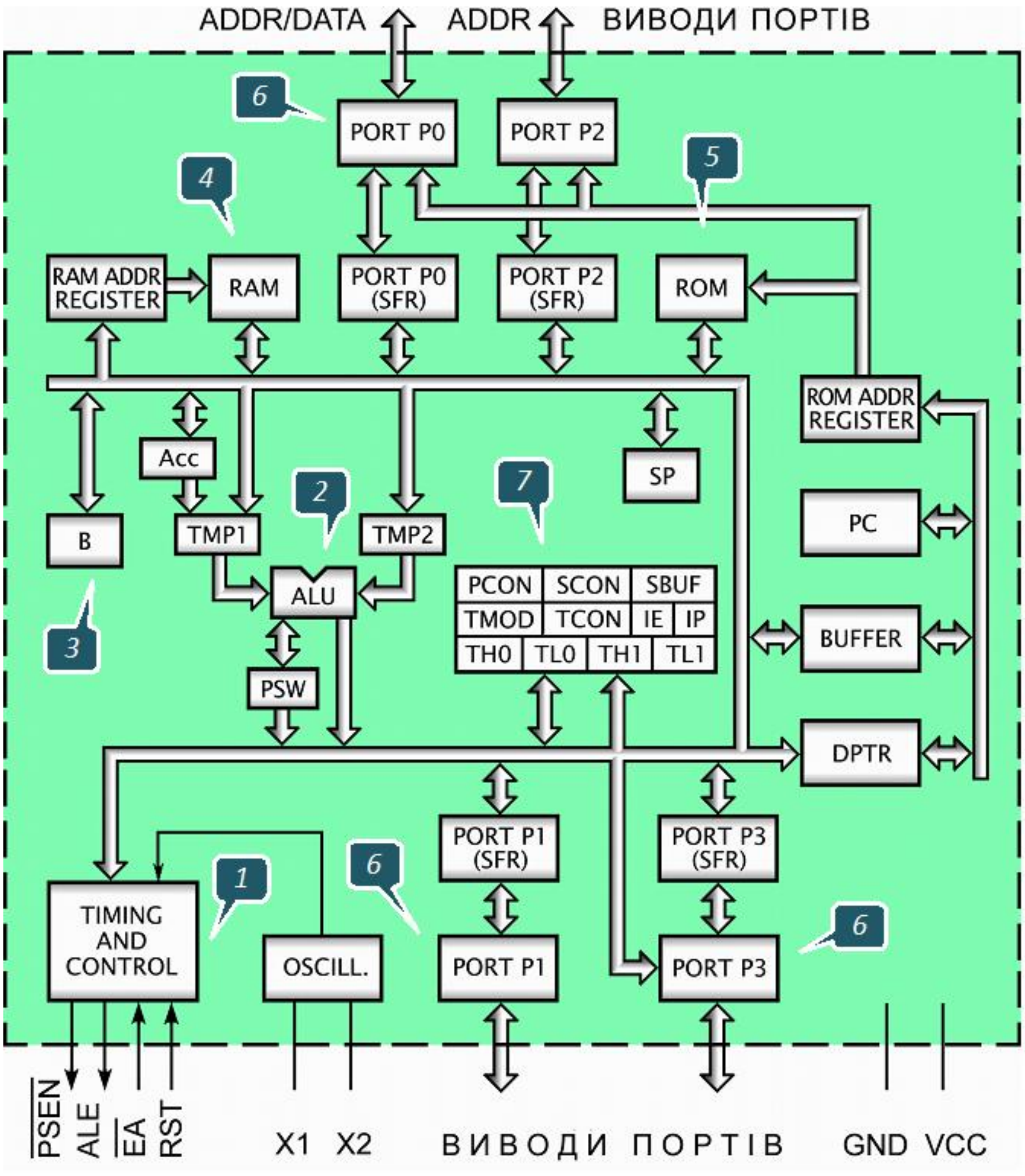
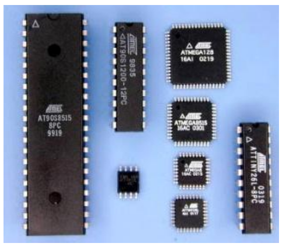


Fig. 1.1: Port input/output

Aside from general-purpose microprocessors and microcontrollers, the market also offers real-time signal processors, which are used in measuring devices, communication, transmission, and reproduction of audio and video streams, locating systems, space, and military equipment, among other applications.

In addition to the CPU, analog-to-digital and digital-to-analog converters are provided on the same chip. An Analog-to-Digital Converter (ADC) converts a continuous input signal into a digital data stream, which is then processed by the processor section, and the processed digital data is converted into an analog signal using a Digital-to-Analog Converter (DAC).

General purpose microprocessors	**Microcontrollers**	**Signal processors**	**Other** (neurochips, partitioned and hybrid processors)
Used: to build personal computers, servers, and multiprocessor systems.	**Used:** to implement simple automation functions.	**Used:** to implement complex algorithms for streaming data in real-time.	**Used:** to implement unique experimental or specific systems.
Features: • high computing performance, • high bit rate, • universal architecture.	**Features:** • built-in program memory and data memory, • bit processor, • timers, counters, ports, interfaces.	**Features:** • high computing performance, • commands to implement typical signal processing algorithms, •built-in ADCs, DACs, or media interfaces.	**Features:** • construction of one processor on several chips, • combination of several types of processors in one product, specific architecture

The so-called programmable logic controller (PLC) is a special microprocessor system used to automate technical operations as well as conventional industrial facilities and complexes (conveyors, roller conveyors, cranes, shredders, mills, classifiers, mixers, packers, robotic and flexible production complexes, and so on).

PLCs are primarily used in industrial manufacturing, but they are also used in building automation (control of access to the premises, control of lighting, heating, ventilation, and air conditioning, control of elevators, escalators, and so on). PLCs can also be used to generate a microclimate in greenhouses, poultry farms, and cattle farms.

A PLC is typically a single-board mini-computer built on a single-chip microcontroller and housed in a standard-sized box (brick-sized). Modular controllers are available. The PLC inputs can be connected to buttons, joystick contacts, switches (ie controls), sensors, and actuators (motors, lighting, heating elements, valves, valves, actuators, and so on).

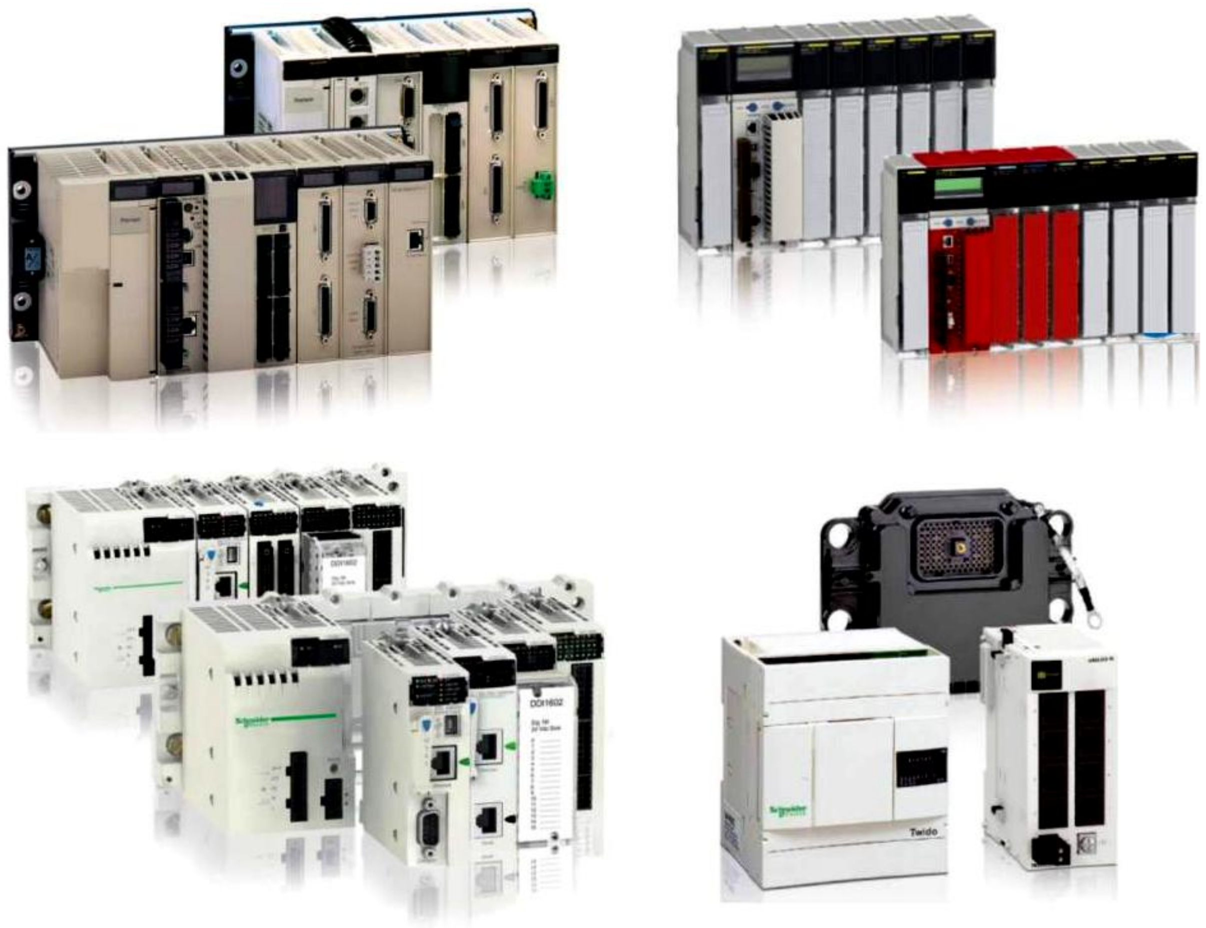


Fig. 1.2: ***Programmable Logic Cinspector***

Apart from hardware unification (using standard sizes, voltage levels, and signal types), the development of "general engineering" programming languages for PLCs has been instrumental in the breakthrough spread of these devices. One no longer needs to hire a top-tier coder to create a user program; this can be done by a technician, an electrician, a chemist, and, of course, an automation professional (sometimes even better). In addition, these programming languages blur the lines between programmers and engineers in complex jobs, making them equally evident to the client (an engineer) and the performer (a programmer).

PLC increases automation system flexibility by making it easy to switch to third-party controllers and transfer programs from one system to another. There are six such programming languages (five of which are standardized), four of which are visual (the program is input visually rather than textually), but as a set of graphic elements (blocks) connected to each other. In most cases, the same controller may be written in many languages of the user's choosing.

1.2. STM Microcontrollers

A popular and extensively utilized platform for creating expert automation solutions across many industries is the STM32 microcontroller.

Because of the microcontroller's performance, good architecture, low power consumption, and low cost, STM32 is a platform that includes STMicroelectronics ARM-based microcontrollers, a multitude of modules and peripherals, and software solutions (IDE) for interacting with hardware. STM32 already has multiple lines for a variety of applications.

For the STM32 series, there are two groups: performance and access. The Performance group includes all peripherals and can run at up to 72 MHz clock speed, while the Access group has a smaller selection and can run at up to 32 MHz. It is important to note that both the Access and Performance groups have the same enclosure types and microcontroller output locations. This means that you can switch out different STM32 versions in the device without having to modify the software structure.

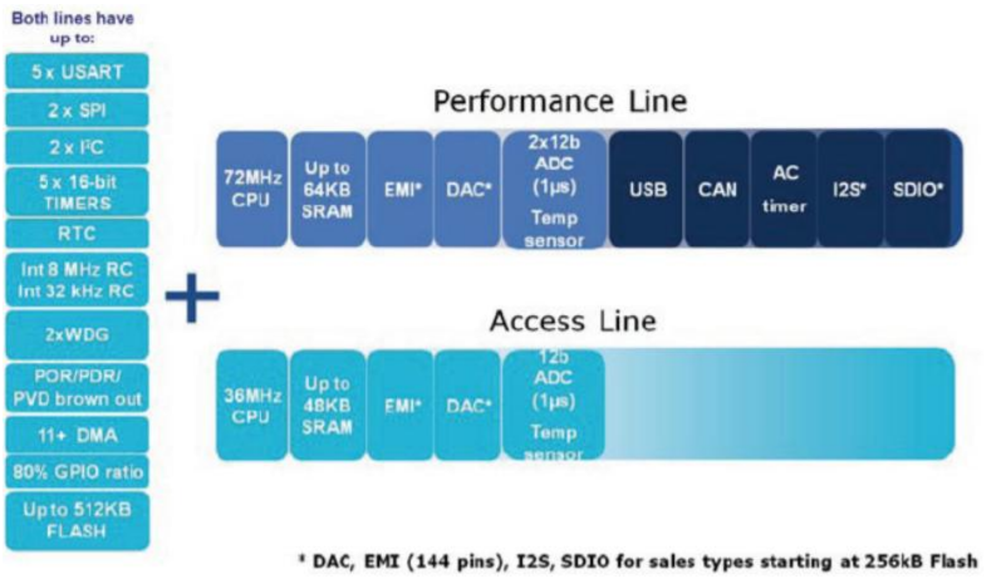


Fig. 1.3: 2 lines of STM microcontrollers

Translation to Fig.: Both lines have up to - Both bands contain up to, 5x16-bit TIMERS

- 5 16-bit timers, Int 8 MHz RC - Int. 8MHz RC, Int 32 kHz RC - Inside 32kg RC,

2xWDG - 2 Watch Timers, 80% GPIO ratio - 80% GPIO, Up to 512KB FLASH - up to

512KB FLASH, 72MHz CPU - 72MHz CPU, Up to 64KB SRAM - up to 64 KB SRAM, DAC - DAC,

ADC (1mus) - ADC (1 μs), Temp sensor - Temperature sensor, AC timer - AC timer, 36 MHz

CPU - 36 MHz CPU, Up to 48KB SRAM - up to 48 KB SRAM,

DAC, EMI (144 pins), I2S, SDIO for sales types starting at 256 kB Flash - DAC, Interface

External Memory, I2S, SDIO in microcontrollers with Flash capacity of 256 KB.

ARM marks each of its processors according to the

architecture audit (ARMV6, ARMV7, etc.). Cortex-M3 has a revision of the ARMV7 M architecture.

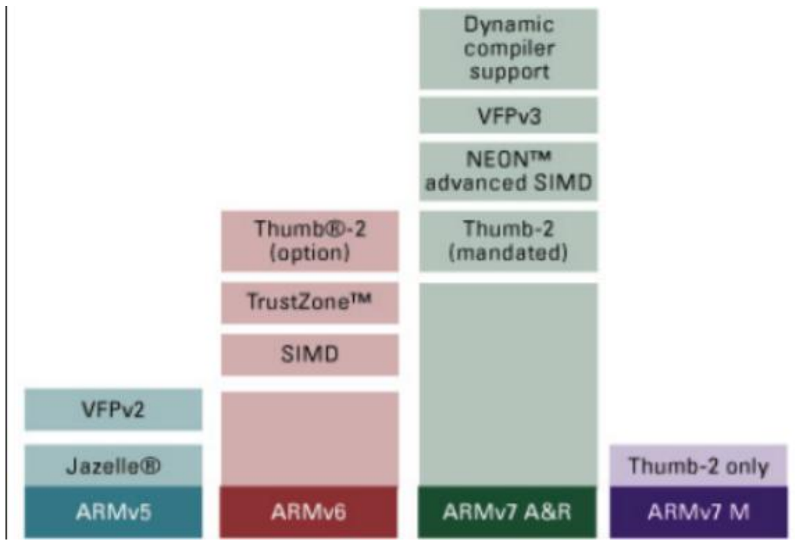


Fig. 1.4: ARM architecture

With the Cortex CPU, most commands may be completed in one cycle. This is achieved by a three-stage conveyor, similar to what is found in the ARM7 and ARM9.

This works great with linear code, but prior to branching, the pipeline is reset and reloaded, allowing computer code to continue running. The ARM7 and ARM9 branches are quite popular. The first command is being executed, the second is being decoded, and the third is being fetched from memory.

The three-stage Cortex CPU processor adds a branch forecasting function to the conveyor, which implies that when a conditional transition command is issued, the next command is issued; commands that are anticipated are removed. Consequently, both conditional transition choices are accessible for execution, resulting in no productivity drop. In terms of performance, it is cost-effective.

Additional program code considerations would be redundant because the pipeline controls the Cortex CPU's overall performance. The Cortex CPU is a load-store architecture RISC processor; before data processing instructions can be executed, operands must be loaded into a central register file. These registers must be used for data operations, and the results must be saved in memory. The worst-case scenario is an indirect transition, in which the predicted transition is unknown and the only way out is to reset the conveyor.

Consequently, all program activity is focused on the CPU register file, which consists of sixteen 32-bit registers. R0-R12 are basic registers that can be used to store variables. The Cortex CPU's registers R13–R15 carry out specific functions. The R13 register stores the stack pointer. This is a grouped register that allows the Cortex CPU to function in two modes with different stacks. This is usually used by RTOS, which runs its "system" code in secure mode. The main stack and the working stack in the Cortex CPU are referred to as the main stack and the working stack. The next register, R14, is called the communication register. When entering the procedure, it is used to save the return address.

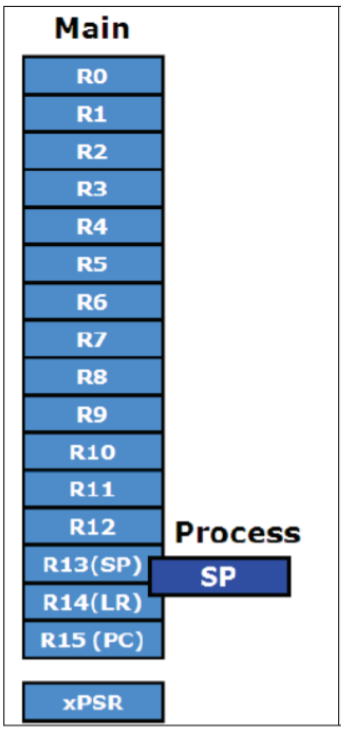


Fig. 1.5: Cortex registers

The values in the bit regions of the xPSR register affect the Cortex CPU. The family of STM32F103xx Microcontrollers consists of an ARM Cortex-M3 32-bit RISC core, high speed embedded memories (Flash memory is up to 128 Kbytes and Static Random Access Memory (SRAM) is up to 20 Kbytes), I/Os (Input/Output) and peripherals which they are cooperating together by connecting to two APB (Advanced Peripheral Bus) buses. In addition to the register file, there is another register called the Program Status Register, which is only accessible using two specific instructions.The values in the bit regions of the xPSR register affect the Cortex CPU. The family of STM32F103xx Microcontrollers consists of an ARM Cortex-M3 32-bit RISC core, high speed embedded memories (Flash memory is up to 128 Kbytes and Static Random Access Memory (SRAM) is up to 20 Kbytes), I/Os (Input/Output) and peripherals which they are cooperating together by connecting to two APB (Advanced Peripheral Bus) buses. In addition to the register file, there is another register called the Program Status Register, which is only accessible using two specific instructions. Fig. 1.6 presents the pinout for the STM32F103 family which is used in this project

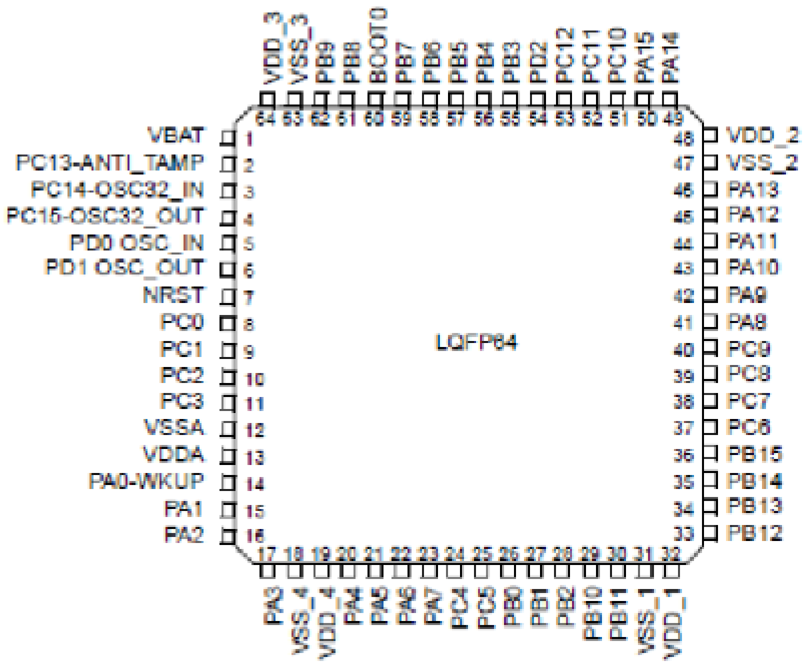


Fig. 1.6: STM32F103xx pins

1.3. Cortex cores

The ARM Cortex-M3 core, the microcontroller CPU and one of the most important components of the microcontroller, is the last version of ARM processors applied to embedded systems. It has well-specified features like: performance at zero state memory access; single-cycle multiplication and hardware division; nested interrupt controller (maskable interrupt 43 channels, interrupt processing); low-power consumption, low-price, low-gate count, etc. The majority of STM32 microcontrollers are based on ARM Cortex-M cores, which offer high performance and a wide range of functions.

The microcontroller is powered by the ARM Cortex-M3 core, which has a number of advantages that will be covered in more detail below, but for now, its most valuable attribute is its adaptability. In just two years, Cortex-M3 has become the industry standard, as evidenced by the number of manufacturers that have joined this architecture; all major microcontroller manufacturers, with the exception of Microchip, have either developed or are developing solutions based on this architecture: STMicroelectronics, Texas Instruments, NXP, ATMEL, Analog Devices, Renesas, and others; ST was one of the first to market with their Cortex-M3 microcontrollers in 2007, quickly rising to the top of the industry. Fig. 1 shows the total number of Cortex-M3 cores sold worldwide, with the majority of ST—nearly 80% in 2009 and roughly 70% in 2010.

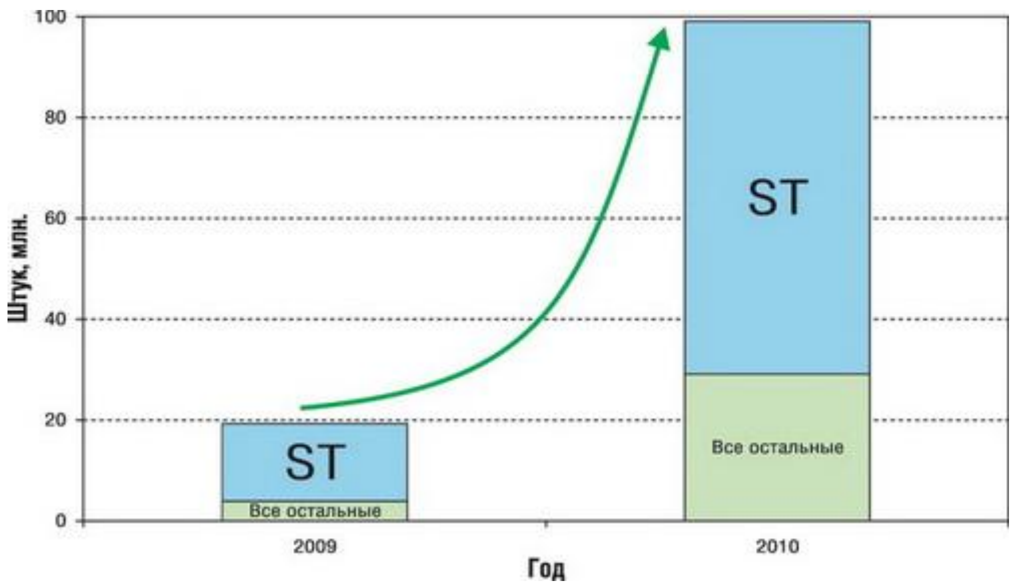


Fig. 1.7: amount of microcontrollers produced

As assemblers are not often used by developers in the 32-bit segment (mostly C and other high-level languages), the portion of the program code that deals with the kernel will not even need to be changed; only work at the peripheral driver level will be required. Consequently, if you write code that distinctly divides the kernel from the perimeter, you may set the stage for a relatively quick switch from one manufacturer to another.

The Cortex-M architecture of the STM32 series encompasses not just Cortex-M3 core microcontrollers, but also Cortex-M0 and Cortex-M4 cores. Cortex-M0 is essentially a Cortex-M3 with a truncated set of instructions designed for lower-performance, lower-cost solutions.

Cortex-M4 is a Cortex-M3 that has been enhanced with new data processing instructions and is intended for applications requiring better performance and more complicated signal processing (floating-point operations at the hardware level). Cortex-M4 can be used in lower-level DSP applications. Cortex-M0 will primarily replace 16-bit and, to a lesser extent, 8-bit microcontrollers.

Since all Cortex-M0 instructions are applicable to the Cortex-M3, the Cortex-M0 kernel code will also function fully on the Cortex-M3 kernel. Conversely, since all Cortex-M3 instructions are still applicable to the Cortex-M4, the Cortex-M3 kernel code will function on the Cortex-M4. For instance, following the development of a product on the Cortex-M3, it will be possible to develop more expensive and sophisticated goods on the Cortex-M4 with minimal fees for program code processing. Since Cortex-M3 is presently a global standard, and since Cortex-M0 and Cortex-M4 are logical extensions of Cortex-M3, it will not be surprising if Cortex-M3 and Cortex-M4 also become standards soon. STM32 with Cortex-M0 and Cortex-M4 processors is scheduled for release by STMicroelectronics until 2011.

1.4. Overview of current trends in cyber defense

a) Increasing the number and complexity of attacks:

Scale of Attacks: Today's Internet attacks are becoming larger in scale, sometimes affecting large companies, government agencies, or even entire countries. For example, DDoS attacks (distributed denial of service attacks) can use a large number of bots to simultaneously attack a server and cause a denial of service.

Complexity of attacks: With an arsenal of different methods such as phishing, carefully planned attacks using software vulnerabilities, and human influence through social engineering, attacks are becoming more complex and difficult to detect.

Attacks on embedded systems: As embedded systems become an integral part of many devices (from home appliances to medical devices), they are becoming a focus for cybercriminals. Attacks can be aimed at hacking embedded systems to gain control over the device or leak confidential information.

b) Widespread use of the Internet of Things (IoT):

Growing number of connected devices: Every year, there are more and more gadgets that are linked to the Internet. These gadgets range from industrial equipment to smart home appliances like refrigerators and thermostats, and they can all serve as potential entry points for cybercriminals.

Inadequate security in IoT: Many IoT devices come with limited resources and computing power, which can lead to a lack of adequate security measures. Insufficient authentication and protection against vulnerabilities can make these devices easy targets for attacks.

Possible consequences of attacks: Cybercriminals can use vulnerabilities in IoT devices to create botnets, leak sensitive information, or even carry out physical attacks (such as disabling industrial equipment).

c) Growing importance of authentication and authorization:

User and device authentication: As threats increase, it is important to ensure that only authorized users and devices have access to systems. Two-factor authentication and biometric methods can be used to increase security.

Access control: Ensuring the principle of least privilege and effective access control can prevent unauthorized access and minimize risks.

Anti-spoofing: Cybercriminals can attempt to spoof credentials or gain unauthorized access. Developing and implementing effective anti-counterfeiting methods helps prevent these threats.

1.5. Analysis of requirements for a cyber-secure workplace

In a world where technology is rapidly penetrating all areas of our lives, cyber security is becoming a key element in ensuring the security of information and infrastructure. A cyber-secure workplace is defined by a number of requirements that form the fundamental basis for protection against cyber threats.

One of the first and fundamental requirements is an authentication and authorization system. The use of strong passwords and two-factor authentication are important elements in preventing unauthorized access. The requirement for regular updates and patches is necessary to eliminate vulnerabilities and increase resistance to attacks.

An integral part of a cyber-protected workplace is the use of anti-virus and anti-malware measures. Malware protection is becoming extremely important, and periodic system scanning for viruses is critical to preventing potential threats.

Another key requirement is the use of encryption for sensitive information. Encryption helps prevent information leakage if devices are lost or stolen. Disallowing unknown or untrusted peripherals is also a necessary requirement to prevent possible threats.

User training and awareness is also an essential element. People need to understand the basics of Internet security, social engineering, and other aspects of cyber defense. Conscious users can detect and avoid potential threats in time.

The most modern element is monitoring and logging. The ability to detect unusual or suspicious activity allows timely response to potential threats, which ensures complete system security.

In general, a cyber-secure workplace is not just a set of technical measures, but also a comprehensive strategy aimed at protecting information in the digital world. Only a comprehensive approach that takes into account technical, organizational and pedagogical aspects will ensure effective protection against modern cyber threats.

Monitoring and detection of threats: The workplace has the ability to monitor system activity and detect possible security threats. This includes I/O analysis, network status, and system diagnostics.

1. **Authentication and authorization systems:** Strong authentication systems such as biometric methods, smart cards or passwords are used to ensure access to the workplace. Users and devices must be authenticated before accessing system functionality.
2. **Data encryption:** The workplace provides encryption of data in the stored and transmitted state, which guarantees the confidentiality of information and protection against interception or modification of data.
3. **Intrusion detection and prevention systems (IDS/IPS):** The workplace includes intrusion detection and prevention systems that analyze network traffic and user actions for unusual or suspicious activity.
4. **Event logging and analysis:** The workplace supports logging of all system events and activities, which allows for detailed analysis of incidents and identification of potential threats.
5. **Updates and security patches:** Workplace security updates and patches are provided regularly to address identified vulnerabilities and ensure an up-to-date level of protection.
6. **Offline mode:** The workplace can go offline if serious threats or emissions are detected to ensure reliable system operation in conditions of limited communication or attacks.
7. **Integration with other systems:** The workplace can interact with other system components such as servers, network elements, and other embedded systems to provide overall cyber protection.

These workplace features and capabilities are designed with modern cyber security standards in mind and aim to provide reliable protection of embedded systems against possible threats and attacks.

1.6. Analysis of threats and vulnerabilities

In today's digital landscape, where automation is becoming a necessity for organizations to function effectively, it is important to be aware of the potential threats and vulnerabilities of automated workplaces. Analysis of these aspects determines the success of cyber defense strategies and the resilience of infrastructure to modern cyber threats.

One of the key threats is the increasing number and complexity of cyber attacks. Automated systems that process and store large amounts of data become an attractive target for attackers. Attacks can be aimed at stealing confidential information, breaking access to systems or introducing malicious programs into the system.

The Internet of Things (IoT) poses a significant concern due to inadequate cyber security. Networked devices in automated workplaces can be used as attack vectors or serve as a gateway for intruders to access corporate networks.

The second aspect is internal vulnerabilities related to people and processes. Social engineering, used to trick employees and gain access to systems, is a serious threat. Also, insufficient awareness of personnel regarding cyber security can cause careless use of systems and leakage of confidential information.

In order to overcome these obstacles, a comprehensive strategy for cyber defense must be put in place. Modern authentication and authorization systems must be installed, data must be securely encrypted, and software must be updated on a regular basis to remove vulnerabilities. Cyber awareness and staff training must also be prioritized.

In conclusion, automated workplaces play a key role in today's business, but their successful operation requires systematic analysis and effective cyber defenses. Awareness of potential threats and timely measures can ensure the safe and reliable operation of automated workplaces in the digital age.

Ability to integrate with other devices or systems

1. **Network interaction:**
   - A cyber-secure workplace must be able to connect to local or remote networks to exchange data and information with other devices or servers.
   - Support for standard network protocols and tools ensures secure communication with other devices.

2. **Integration with monitoring systems:**
    - The workplace can be integrated with monitoring and management systems to provide centralized control over cyber security and system health.
    - Notifications about events detected by the system can be sent to central monitoring.
3. **Integration with other embedded systems:**
    - The workplace can be integrated with other embedded systems that are part of a larger system.
    - This allows combining the functionality of different devices and ensures interaction between them.
4. **Interfaces and APIs:**
    - The presence of open interfaces and APIs allows other developers and systems to interact with the cyber-protected workplace and use its functionality.
    - This makes the system more flexible and extensible.
5. **Compatibility with standards:**
    - It is important to consider the compatibility of a cyber-secure workplace with the standards and protocols used in other systems and devices.
    - This ensures seamless integration and data exchange with other systems.
6. **Interaction security:**
    - It is important to ensure secure interaction with other systems and devices to prevent possible threats or attacks that may occur during
    -
7. **Security requirements:**
    - Measures to protect the confidentiality, integrity and availability of data and functions.
    - User and device authentication and authorization requirements.
    - Protection against internal and external threats, including hacking attacks, outages, and Man-in-the-Middle attacks.

SECTION 2

ADVANTAGES OF STM

2.1 Pin-to-pin compatibility

One of the reasons for the STM32 family's global appeal is its pin-to-pin compatibility. If the STM32 kernel's versatility allows you to change the manufacturer with minimal cost of software code, the STM32 family's pin-to-pin compatibility allows you to change the amount of memory (flash memory and RAM) and peripherals (Ethernet, USB, CAN, etc.) without touching the printed circuit board.

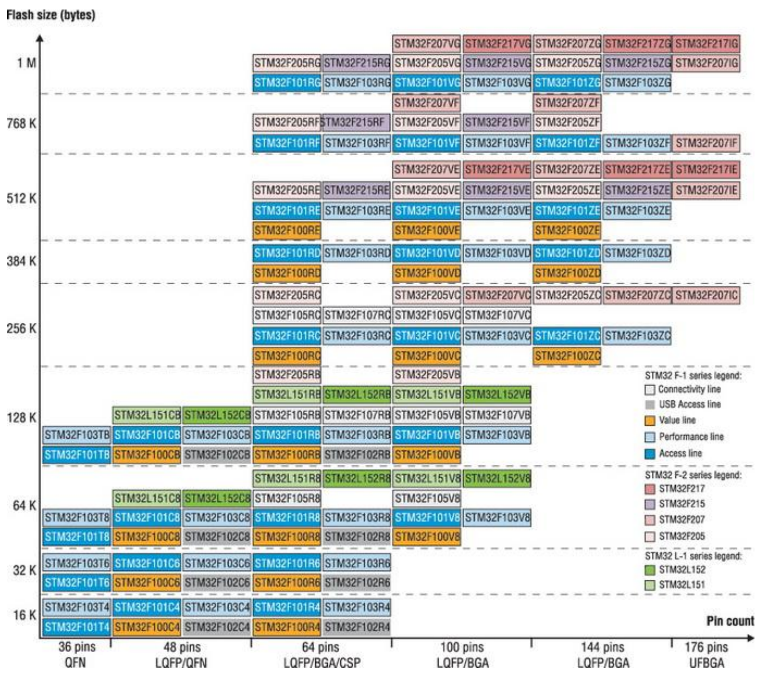


Fig. 2.1: Products of the STM32 family depend on flash memory and the case.

If, for example, the developer required a LQFP 64 casing as well as a minimal set of memory and functionality, his options would be limited to the chip STM32F100R4 (16 Kbytes of flash memory, 4 Kbytes of RAM, 24 MHz). If I then require a faster ADC and 128 KB of flash memory, the STM32F101RB chip is used. Because all of the I/O in these chips is similar, the board does not need to be touched.The developer then switches to the STM32L151RB microcontroller, which uses less power while retaining all of the STM32F103RE's memory and connections. In the latter case, the developer should be aware of one minor change (the Vbat signal becomes Vlcd), but he should not need to change anything on the board.You can take a simpler microcontroller from the line, lowering component costs.Pin-to-pin compatibility is the ideal lever for lowering the cost of your selections because it makes the procedure significantly easier (less memory, fewer peripherals, etc.). You can put many different goods into production with high efficiency if you plan all future generations of your product on the basis of pin-to-pin compatibility throughout development.

It should be noted that the level of pin-to-pin compatibility varies. The graphic below displays the pin-to-pin compatibility of the various STM32 series, as well as a concise description of their main characteristics. Because standard peripherals like as SPI, USART, I2C, clocks, watch timers, reset methods, and so on are shared by all lines, they are not detailed. The STM32W line (Cortex-M3 + ZigBee) is not included in Fig. 4 because it lacks pin-to-pin compatibility with the STM32F-1, STM32F-2, and STM32L families. This is due to the fact that it was developed in conjunction with Ember, which donated the radio frequency component, restricting ST's development flexibility.

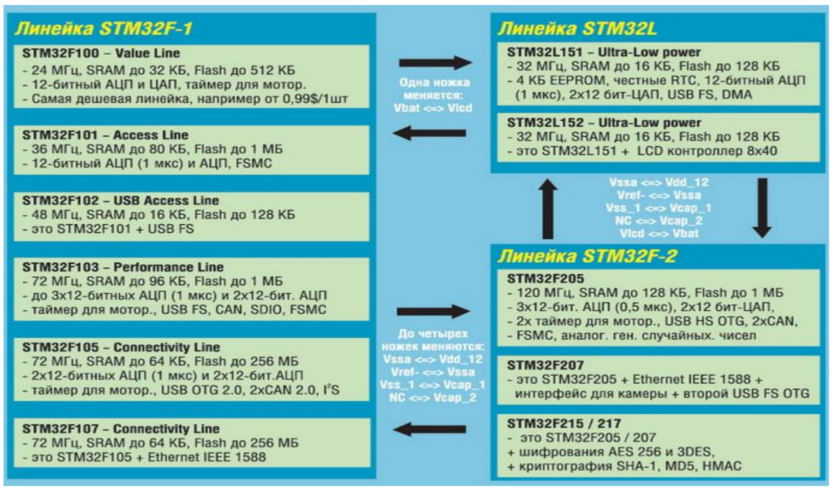

Линейка STM32F-1
STM32F100 – Value Line
- 24 МГц, SRAM до 32 КБ, Flash до 512 КБ
- 12-битный АЦП и ЦАП, таймер для мотор.
- Самая дешевая линейка, например от 0,99$/1шт
STM32F101 – Access Line
- 36 МГц, SRAM до 80 КБ, Flash до 1 МБ
- 12-битный АЦП (1 мкс) и АЦП, FSMC
STM32F102 – USB Access Line
- 48 МГц, SRAM до 16 КБ, Flash до 128 КБ
- это STM32F101 + USB FS
STM32F103 – Performance Line
- 72 МГц, SRAM до 96 КБ, Flash до 1 МБ
- до 3x12-битных АЦП (1 мкс) и 2x12-бит. АЦП
- таймер для мотор., USB FS, CAN, SDIO, FSMC
STM32F105 – Connectivity Line
- 72 МГц, SRAM до 64 КБ, Flash до 256 МБ
- 2x12-битных АЦП (1 мкс) и 2x12-бит.АЦП
- таймер для мотор., USB OTG 2.0, 2xCAN 2.0, I²S
STM32F107 – Connectivity Line
- 72 МГц, SRAM до 64 КБ, Flash до 256 МБ
- это STM32F105 + Ethernet IEEE 1588
Одна ножка меняется: Vbat <=> Vlcd
До четырех ножек меняются: Vssa <=> Vdd_12 Vref- <=> Vssa Vss_1 <=> Vcap_1 NC <=> Vcap_2
Линейка STM32L
STM32L151 – Ultra-Low power
- 32 МГц, SRAM до 16 КБ, Flash до 128 КБ
- 4 КБ EEPROM, честные RTC, 12-битный АЦП (1 мкс), 2x12 бит-ЦАП, USB FS, DMA
STM32L152 – Ultra-Low power
- 32 МГц, SRAM до 16 КБ, Flash до 128 КБ
- это STM32L151 + LCD контроллер 8x40
Vssa <=> Vdd_12 Vref- <=> Vssa Vss_1 <=> Vcap_1 NC <=> Vcap_2 Vlcd <=> Vbat
Линейка STM32F-2
STM32F205
- 120 МГц, SRAM до 128 КБ, Flash до 1 МБ
- 3x12-бит. АЦП (0,5 мкс), 2x12 бит-ЦАП,
- 2x таймер для мотор., USB HS OTG, 2xCAN,
- FSMC, аналог. ген. случайных. чисел
STM32F207
- это STM32F205 + Ethernet IEEE 1588 + интерфейс для камеры + второй USB FS OTG
STM32F215 / 217
- это STM32F205 / 207
+ шифрования AES 256 и 3DES,
+ криптография SHA-1, MD5, HMAC

Fig. 2.2: Pin-to-pin compatibility within STM32 families.

The STM32F-1, STM32F-2, and STM32L pin-to-pin series have full capacity. The STM32F-1 series, for example, includes more than 90 chips divided into five families based on application, with full pin-to-pin compatibility. The STM32F-1 and STM32L have excellent pin-to-pin compatibility, while the newer STM32F-2 series will necessitate more work.

If you still need to adjust the number of inputs and outputs (change the case), the requirement for re-tracing is minimized since all signals in the STM32 family are stored to the maximum on one or the other side of the case, independent of its size. It also facilitates the work.

Software compatibility supplements the previously described pin-to-pin compatibility. Software compatibility is complete across and within families, including the STM32W family. According to STMicroelectronics, future STM32 families based on the Cortex-M0 and Cortex-M4 cores will be pin-to-pin and software-compatible with existing ones. Finally, STM32 is a platform that you can utilize now and in the future.

2.2 Debug boards

When you're interested in a microcontroller, you want to test it out. This necessitates the use of debugging boards. There are an incredible number of debuggers and their features available for the STM32 family. STM32VLDiscovery debugging, for example, is provided by the manufacturer. It comes with a programmer and is compatible with the STM32F100 microcontroller. This is a low-cost and rapid way to become acquainted with the STM32 family. Third-party manufacturers, on the other hand, such as Keil, IAR, Raisonnance, and others, provide more advanced debugging with all of the necessary functionality. In addition, several small enterprises have developed their own STM32 debugging solutions. The STM32VLDiscovery plug-in is also available. The fascinating inemo v2 module (STEVAL-MKI062V2), which combines a collection of sensors (temperature sensor, pressure sensor, two gyroscopes, three-axis accelerometer with compass) and software for rapid solution building, is another novel setting for STM32 microcontrollers. There is also a complete configuration for working with the STM3210B-MCKIT electric drive, which includes a three-phase motor, a ten-amp three-phase inverter, a power supply, and its own Microcontroller part.

Sometimes difficult solutions necessitate not just debugging but also real-time debugging. One advantage of the Cortex-M3 core is that it includes an Embedded Trace Module (ETM) that allows you to do this function. All STM32F-1 microcontrollers with memory sizes more than 512 KB, as well as the STM32L and STM32F-2 series, incorporate this module. A specialized coder, such as J-trace or Ulink-pro, will be necessary as well.

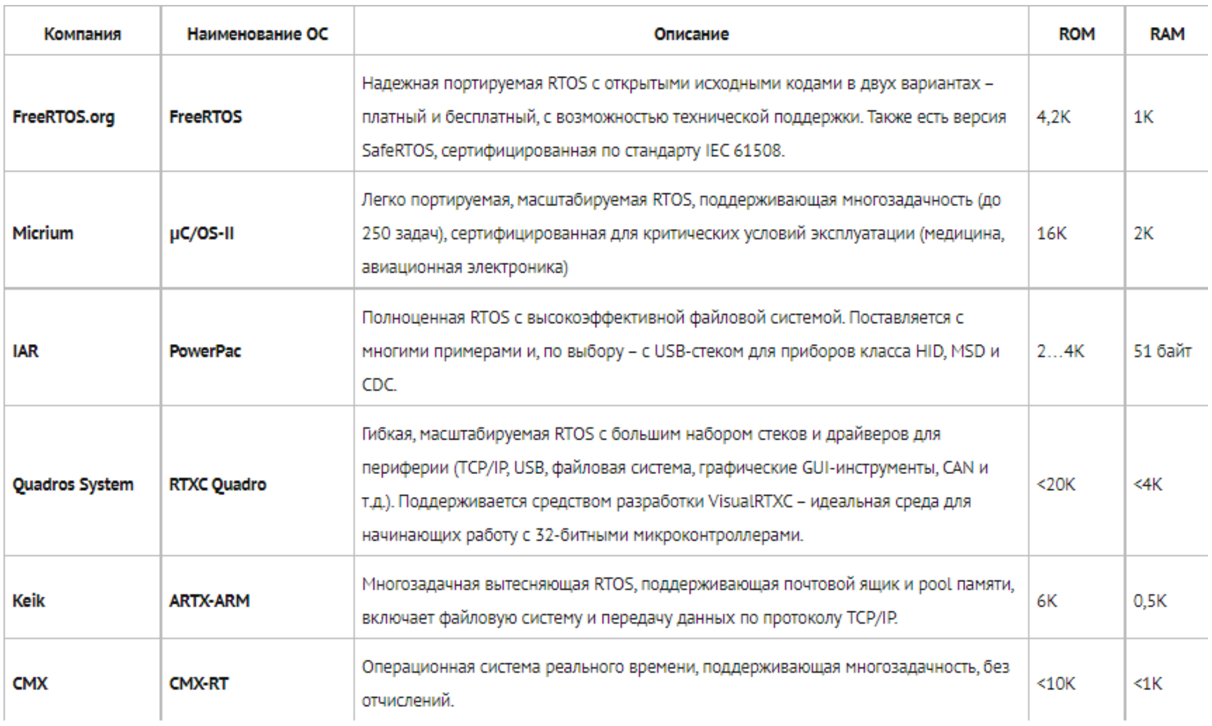
Microcontroller software development necessitates the use of a programming environment and C-tools. Because ARM-based microcontrollers are so common, there are several commercial and free choices for development tools. If you've previously used an ARM-core solution, the transition to ARM Cortex-M3 is simple; simply update the software in your development tools.

We not only have a large selection of debuggers and development tools. Libraries are critical components of the development process. There are several free libraries available for the STM32 family, including the following:

1. All standard peripherals are supported (UART, SPI, I2C, CAN, ADC, DAC, timers, all clock sources, FSMC, IO, DMA, RTC, and so on).
2. USB interface library (including modes like mass storage, HID, DFU, CDC, audio, and full-speed host);
3. Ethernet interface library (MAC layer from ST and free complete TCP / IP layer from Interniche); motor control libraries (three-phase brushless motors);
4. DSP libraries (PID, IIR, FFT, FIR); DSP libraries for sound reproduction (decoding and encoding based on the SPEEX codec, with high sound quality); DSP libraries for graphic solutions.

Each of these libraries has application documentation. Aside from free libraries, there are several paid libraries available from companies such as Keil, IAR, Micrium, Segger, Greenhills, Quadros, CMX, and others.

I must briefly discuss operating systems. Unfortunately, the Cortex-M3 kernel does not support Linux. Developers that need it should concentrate on cores such as ARM9. In terms of RTOS, however, the Cortex-M3 core exceeds the ARM7. The kernel has two privileged modes and two specialized stacks. It also offers a great deal of flexibility in terms of interrupt priority and an additional Systick system timer. All of this allows you to build better real-time operating systems. Many operating systems have already been developed for STM32 microcontrollers.



Компания	**Наименование ОС**	**Описание**	**ROM**	**RAM**
FreeRTOS.org	**FreeRTOS**	Надежная портируемая RTOS с открытыми исходными кодами в двух вариантах – платный и бесплатный, с возможностью технической поддержки. Также есть версия SafeRTOS, сертифицированная по стандарту IEC 61508.	4,2K	1K
Micrium	**µC/OS-II**	Легко портируемая, масштабируемая RTOS, поддерживающая многозадачность (до 250 задач), сертифицированная для критических условий эксплуатации (медицина, авиационная электроника)	16K	2K
IAR	**PowerPac**	Полноценная RTOS с высокоэффективной файловой системой. Поставляется с многими примерами и, по выбору – с USB-стеком для приборов класса HID, MSD и CDC.	2…4K	51 байт
Quadros System	**RTXC Quadro**	Гибкая, масштабируемая RTOS с большим набором стеков и драйверов для периферии (TCP/IP, USB, файловая система, графические GUI-инструменты, CAN и т.д.). Поддерживается средством разработки VisualRTXC – идеальная среда для начинающих работу с 32-битными микроконтроллерами.	<20K	<4K
Keik	**ARTX-ARM**	Многозадачная вытесняющая RTOS, поддерживающая почтовой ящик и pool памяти, включает файловую систему и передачу данных по протоколу TCP/IP.	6K	0,5K
CMX	**CMX-RT**	Операционная система реального времени, поддерживающая многозадачность, без отчислений.	<10K	<1K

Fig. 2.3: List of real-time operating systems for STM32 microcontrollers

Specific libraries for the Simulink module in Mathworks' Matlab software are developed to generate ready-made C-code from Simulink for STM32 microcontrollers. In other words, computer code may be produced graphically. National Instruments is planning to port their popular Labview software to the Cortex-M3.

As a result, we can assert that all of the conditions are in place for successful, convenient, and speedy development. Individuals on a restricted budget can also take use of extremely low-cost debugging, free programming environments, free libraries, and RTOS operating systems. In other words, the developer's intellect and imagination are all that is necessary to succeed with STM32!

2.3 The low-cost

STMicroelectronics separates itself from other semiconductor manufacturers by offering exceptional value for money while adhering to the highest quality standards. The STM32 series is a good example. This family comprises the low-cost Value Line STM32F100 series microcontrollers, which cost less than a dollar and come with a wide range of peripherals. When compared to the original Cortex-M0 solutions on the market, the developer gains several times the capability for a few dozen cents extra.

The STM32F100, like other STM32 microcontrollers, has an inbuilt DMA controller, which frees up the core from data processing and transmission. Its absence in the current implementation of Cortex-M0 is a significant constraint in the overall performance of a microcontroller.

It should be noted that the STM32 family has high-quality analog peripherals. The STM32F100, for example, has a 12-bit 16-channel ADC with a measurement duration of 1.2 seconds. Other series and families have faster ADCs (1 s or even 0.5 s). This ADC has several advantages, including the ability to conFig. batch measurements (channel measurement order), the ability to vary measurement duration on each channel, the ability to work in analog guard mode (two programmable threshold voltages), and a built-in temperature sensor (approximate), external trigger. with addition to the aforementioned, with STM32 microcontrollers with multiple ADCs, collaboration with multiple ADCs can increase conversion speed several times. For example, in the STM32F-1 line you can reach a speed of 0.5 μs, and in the case of the STM32F-2 line - up to 160 ns (!).

The STM32F100 also features a large number of general-purpose inputs and outputs (GPIO). The STM32 family distinguishes itself from its competitors in the market by having a high proportion of GPIO in relation to the total number of legs of the case: for example, in the STM32F100 series, the LQFP48 case has 37 GPIO, the LQFP64 case has 51 GPIO, and the LQFP100 case has 80 GPIO. These GPIOs are quite versatile; they can not only be conFig.d in many common modes (two-stroke circuit, open collector, pull-up, pull-down, etc.), but they can also be reassigned inputs or outputs for peripherals (remapping). To avoid electromagnetic interference, GPIO speed is regulated: in the case of the STM32F100, it can be 2 MHz, 10 MHz, and, in principle, up to 50 MHz, but the limiting factor will be the core's clock frequency, which is 24 MHz. GPIO speeds in the STM32F-2 series can approach 100 MHz. This creates new opportunities for data management and transmission. You may construct your own interfaces programmatically. The STM32 series also has an outstanding built-in CAN 2.0 controller licensed directly from the standard's creator, Bosch.

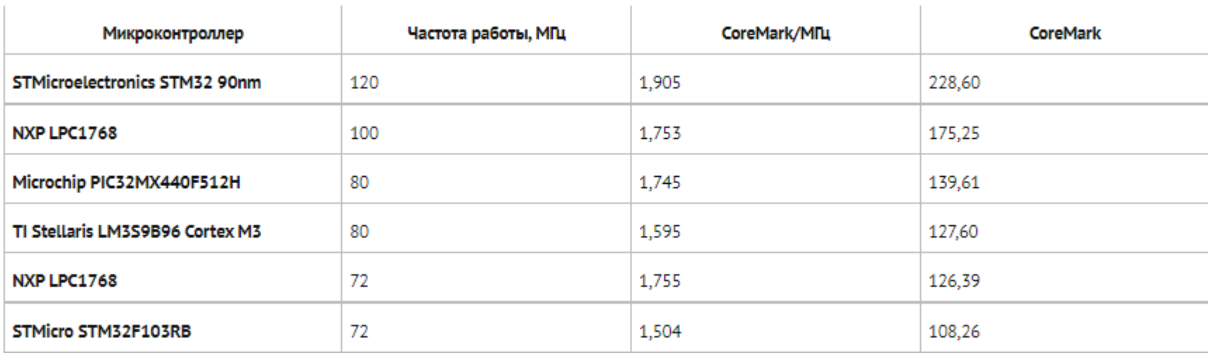
Some STM32 lines have useful FSMC and SDIO peripherals (for example, the STM32F101 and STM32F103 series). SRAM, NOR Flash, and NAND Flash are examples of external memory interfaces. SDIO is a standard interface for working with memory cards such as SD, mini SD, micro SD, and MMC. The only type of memory that the STM32 series does not support is DRAM, however most developers do not require it. Keep in mind that the STM32 series has a hardware CRC (Cyclic Redundancy Check - cyclic redundancy code). This functionality allows you to verify the data exchange quality without loading the kernel. All STM32 microcontrollers, with the exception of the STM32W, have a significant number of multifunction 16-bit timers (the STM32F-2 has 32-bit timers) and one or more 12-bit DACs. The STM32F-2 series features an unusual feature: an interface for connecting DCMI cameras that can function at rates of up to 54 MB/sec, allowing cameras with matrix sizes of up to about 1 megapixel to be supported. The camera's compressed JPEG data can also be accepted via this interface. This opens up new possibilities for security systems that don't require complex video processing, but rather more simple tasks like snapping a photo during a theft or employing a camera to relay data via built-in Ethernet. The FSMC and SDIO interfaces will help in the dynamic storing of a dense flow of camera data in external memory. An MPU memory protection module is included in all STM32L, STM32F-2, and STM32F-1 devices with more than 512 KB of memory, which increases system security even further. Memory can be partitioned into segments and given varied access rights (read, write, execute - like in Unix). This allows you to restrict program code access to certain memory zones and set up fully functional privileged and unprivileged modes. The MPU issues an interrupt if the software attempts to access protected memory regions. STM32 series microcontrollers with more than 512 KB of flash memory have a useful feature: the memory is divided into two banks. This allows you to put two different microcontroller firmwares in each bank and select one or the other firmware at the start of the microcontroller, dramatically altering the device's functionality. It also enables you to dynamically download a new version of the firmware to another bank via one of the communication ports without halting the principal running program in one bank, and then quickly transition the microcontroller to a new version of the software.

The STM32 series stands out from the crowd by functioning effectively in the temperature range of -40 to 85 ° C. The performance of the core and peripherals is completely preserved. A number of STM32 series devices have been verified for extended temperature ranges spanning from -40 to 105°C.

2.4 High-performance cores

One of the Cortex-M3 core's advantages is its high performance, which stems from ARM's significant experience in CPU development. The Cortex-M3 core, for example, has its own data bus, instruction bus, and peripheral management bus (Harvard architecture). This eliminates the delays that can occur with the von Neumann design, which delivers all data to the kernel over a single channel. The kernel's mathematical abilities are also impressive: hardware multiplication of 32 bits by 32 bits is built into the Cortex-M3 core and can be accomplished in one clock cycle if the output is in 32-bit format.Depending on the intricacy of the division, hardware division takes 2...12 cycles. The whole interrupt scheme has been greatly improved: an integrated NVIC interrupt controller that allows you to establish up to 256 priority between interrupts and adjust them dynamically. The time lost due to competing interrupts is greatly reduced (12 cycles versus 24... 42 in the case of ARM7). Cortex-M3 has a set of Thumb-2 instructions (a combination of 16-bit and 32-bit instructions with no transitions) as well as optimized RAM use (bit banding). All of these enables a potential performance of 1.25 DMIPS / MHz with only a tiny amount of program code in memory.

The initial STM32F-1 line provided several opportunities for developers, but performance was limited (72 MHz), and it was quickly exceeded by the Stellaris (80 MHz) and LPC17 families (100 MHz). ST has replied by creating the STM32F-2 family, which operates at 120 MHz and achieves 150 DMIPS. The results of performance measurements based on the CoreMark standard are shown in Table 2.



Микроконтроллер	Частота работы, МГц	CoreMark/МГц	CoreMark
STMicroelectronics STM32 90nm	120	1,905	228,60
NXP LPC1768	100	1,753	175,25
Microchip PIC32MX440F512H	80	1,745	139,61
TI Stellaris LM3S9B96 Cortex M3	80	1,595	127,60
NXP LPC1768	72	1,755	126,39
STMicro STM32F103RB	72	1,504	108,26

Fig. 2.4: Comparison of the performance of 32-bit microcontrollers based on CoreMark

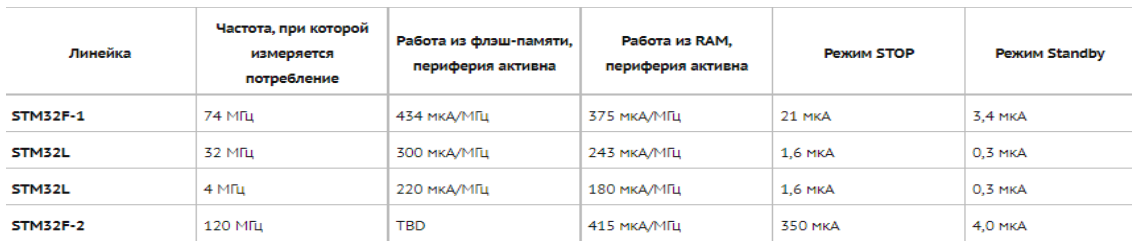
The value of the field "CoreMark" indicates the absolute performance of the microcontroller (larger values correlate to higher performance). The "CoreMark /

MHz" field provides the normalized CoreMark findings in relation to the frequency of the microcontroller. In other words, we may evaluate how well the microcontroller operates at high frequencies. The STM32F-2 family provides more computational power. The remarkable performance of the STM32F-2 series is achieved by the use of a unique flash memory accelerator and a new 90 nm manufacturing process.It should be emphasized that the high performance of the microcontroller should always be evaluated in connection with the quality of the peripherals, because the constraints of the peripherals frequently have to compensate for the shortcomings of the kernel. The STM32 peripherals are of the highest quality. It should also be mentioned that the STM32F-2 family of microcontrollers has variants with 176 outputs.

2.5 Low energy consumption

The ARM Cortex architecture has been a global success, thanks mostly to its low energy consumption. Because this is especially important in the case of phones and portable objects, the ARM processor is at the heart of any phone or smartphone. However, as energy prices continue to rise, the difficulties of energy efficiency and conservation are becoming increasingly important for all goods. STMicroelectronics designed the STM32L family specifically for stand-alone power systems, combining strong ARM Cortex-M3 core performance with low power consumption. The Cortex-M3 core has a sleep mechanism that has been supplemented with STM32F family modes. Similarly, new power modes, dynamic core voltage change mode, and redesigned energy-efficient peripherals have increased the adaptability of the STM32L power management.

The power consumption of STM32 microcontrollers is shown in the table below. The STM32L microcontroller family is constructed utilizing a specific technological process that reduces transistor leakage, resulting in optimal performance. In the table, the phrase TBD stands for "to be specified later," which signifies that the manufacturer has not yet measured certain characteristics. This is because the STM32F-2 microcontroller family was just recently introduced to the market.



Линейка	Частота, при которой измеряется потребление	Работа из флэш-памяти, периферия активна	Работа из RAM, периферия активна	Режим STOP	Режим Standby
STM32F-1	74 МГц	434 мкА/МГц	375 мкА/МГц	21 мкА	3,4 мкА
STM32L	32 МГц	300 мкА/МГц	243 мкА/МГц	1,6 мкА	0,3 мкА
STM32L	4 МГц	220 мкА/МГц	180 мкА/МГц	1,6 мкА	0,3 мкА
STM32F-2	120 МГц	TBD	415 мкА/МГц	350 мкА	4,0 мкА

Fig. 2.5: Power consumption of STM32 microcontrollers in different modes (typical value at room temperature)

The greatest advantage of the STM32L series is shown in STOP and Standby modes, when power consumption drops to 1.6 A and 300 nA, respectively. STOP mode disables all clock sources, but RAM contents are retained, and the transition to active mode takes only a few microseconds. In Standby mode, everything is turned off except the real-time clock. It already takes tens of microseconds to wake up from this state. It should also be noted that the STM32L series includes a Low Power Run mode, in which the core is solely driven by a low-frequency internal clock (32 kHz), and consumption drops to 10 A.

The STM32L microcontroller's typical consumption in active and sleep modes will likely be higher than that of rivals' best low-consumption 8-bit or 16-bit microcontrollers. When compared to 8- or 16-bit devices, the STM32L's great performance allows you to complete all tasks in shorter time and return to sleep, resulting in lower average consumption. As a result, the challenge of low consumption is more complicated than it appears to developers.

2.6 Reliability

Many modern technologies must meet additional safety standards in addition to providing excellent performance and utility. STM32 incorporates a number of hardware characteristics that aid in the smooth operation of the system. Among these are a supply voltage drop detector, a security synchronization mechanism, and two distinct watchdog clocks.

It should be updated on a regular basis. If the update happens sooner or later, the watchdog timer will be triggered, resulting in an interrupt. The second watchdog timer is self-contained and has its own external oscillator. The synchronization system provides the main external oscillator fault detection functions before switching to the built-in 8 MHz RC oscillator.

2.7 Security

One requirement for modern electronics is that software code be safeguarded from unauthorized access. The evaluation port may be used to provide read protection for STM32 flash memory.

When read protection is enabled, flash memory is write-protected against vector table interruptions. The STM32 microcontrollers additionally have a real-time clock and a small SRAM battery-powered area. The content of this section prohibits module intrusion.

2.8 Bus matrix

The system bus and data bus are linked to an external microcontroller via a set of high-speed buses arranged in the shape of a matrix. This permits the creation of a large number of parallel communication channels for data transmission via the bus between Cortex buses and external control devices like DMA and SRAM. If two bus control devices (for example, a Cortex CPU and a DMA channel) try to access the same peripherals at the same time, the built-in arbiter resolves the conflict and allows access to the device with the higher priority. The STM32 is built in such a way that the DMA and the Cortex CPU work in tandem, as we will see when we look at the DMA module.

2.9 Entering Low Power Mode

While maintaining great performance, the STM32 includes low-power modes in addition to the regular active mode. If you use these modes carefully (SLEEP, STOP, STANDBY), the application can run for an extended amount of time without needing to replace the battery. The STM32 microcontroller is a low-power microcontroller with strong CPU performance. During the Cortex review, we observed that the Cortex processor may enter low-power modes that deactivate the CPU and Pertex peripherals. When the Cortex processor enters low power mode, it may send an SLEEPDEEP signal to the rest of the microcontroller to tell the rest of the microcontroller of the change. When the WFI and WFE directives are executed, the Cortex CPU enters low power mode. STM32 is included in which mode is determined by the power mode control register values. To put the Cortex core to sleep, use the Wait For Interrupt (WFI) or Wait For Event (WFE) commands. When you give the WFI command, the Cortex kernel returns to active mode and handles the interrupt. When the interrupt handling procedure is completed, there are two possible results. To begin, the Cortex kernel runs code in normal mode. Second, the Cortex kernel will go back to sleep if the SLEEPON EXIT flag in the System Control Register is set. This feature allows you to write programs that are totally interrupt-driven, which means that the kernel will wake up, execute some code, and then sleep again.

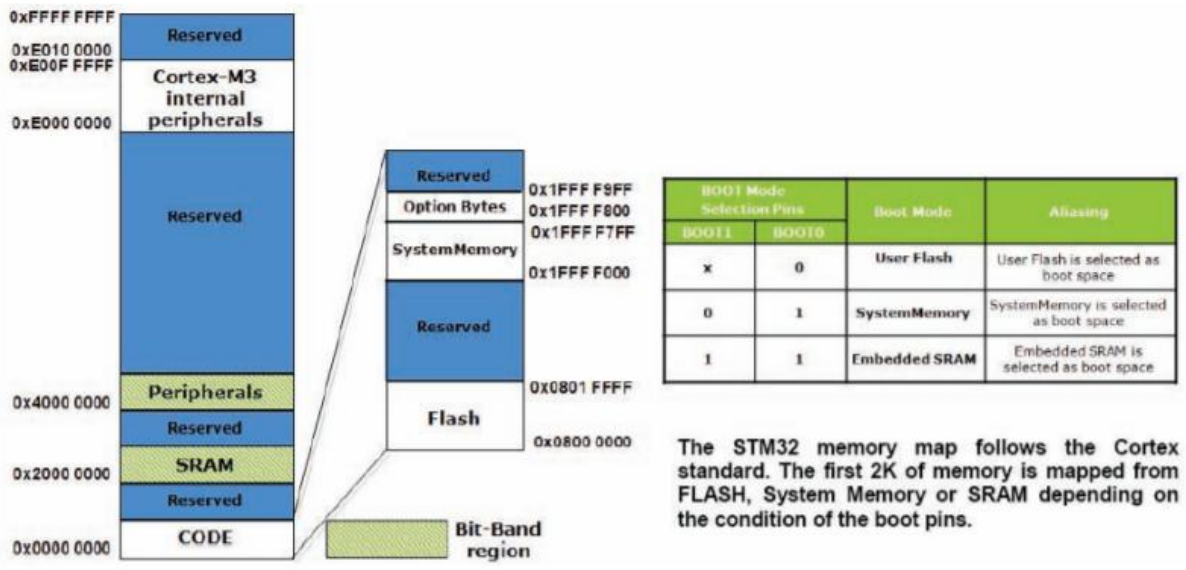
The WFE instruction asks the Cortex kernel to continue program execution from where it was stopped. There is no transition to the exception handling procedure. The kernel gets woken up by a basic peripheral interrupt that was not saved in the NVIC register. This enables peripherals to wake up the kernel and continue performing code without using the interrupt handling approach. Although the WFI and WFE commands are not available in C, the Thumb-2 command compiler has internal macros that may be used to run regular C commands:

```
__WFI
__WFE
```

In addition to the low power modes SLEEPNOW and SLEEPONEXIT, the Cortex kernel can output a SLEEPDEEP signal to another part microcontroller.

2.10 Memory Allocation

Despite the STM32's many internal busses, the programmer presents a linear 4 GB address space. Because the STM32 is a Cortex-based microcontroller, the memory card conforms to the standard Cortex memory allotment. As a result, the program memory starts at address 0x00000000. The built-in SRAM starts at 0x20000000 and is all in the bit segmentation area. The memory for peripheral users begins at address 0x40000000 and is also located in the bit segmentation area. Cortex registers are located starting at 0xE0000000.



BOOT Mode Selection Pins		Boot Mode	Aliasing
BOOT1	BOOT0		
x	0	User Flash	User Flash is selected as boot space
0	1	SystemMemory	SystemMemory is selected as boot space
1	1	Embedded SRAM	Embedded SRAM is selected as boot space

The STM32 memory map follows the Cortex standard. The first 2K of memory is mapped from FLASH, System Memory or SRAM depending on the condition of the boot pins.

Fig. 2.6: Memory allocation

FLASH memory is separated into three sections. The first is the user's Flash memory, which starts with 0x00000000. System memory, often known as a large information block, is the second type of memory. During the microcontroller manufacturing process, the bootloader is programmed in these four To Flash memory. The third section, known as the small information block, starts at 0x1FFFF800 and consists of a set of optional bytes that allow you to modify the

STM32 system. The bootloader will use USART1 to load code into FLASH user information memory. Set the external pins BOOT0 and BOOT1 to low and high, respectively, to enter bootloader mode. In this state of conclusion, the system memory block is shown at 0x00000000. After the reset, STM32 will begin executing the boot code rather than the user program from Custom Flash. The PC downloading application is available on the ST website. This program communicates with the bootloader and wipes and reflashes Custom Flash Memory.

2.11 Components of STM32F103 MCU

This section describes the microcontroller core, memory, I/Os, and peripherals shown in Block Diagram 3. Below are basic descriptions for the STM32F103 microcontroller components.

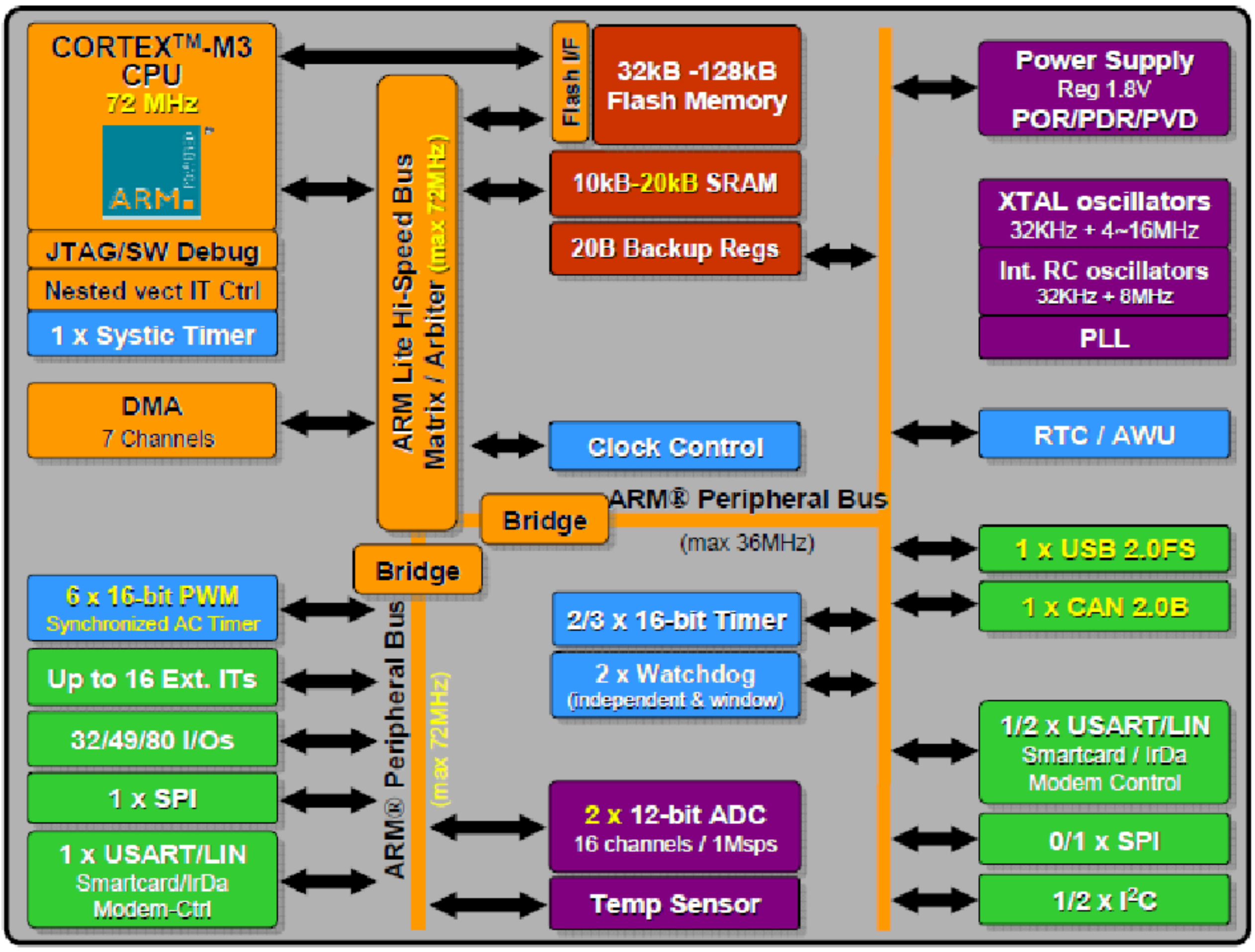


Fig. 2.7: System components

2.11.1 System Architecture

Buses, general purpose DMA (Direct Memory Access), internal SRAM, and internal flash memory are all components of system architecture, with some serving as masters and others serving as slaves. The bus architecture for the STM32 series of microcontrollers is seen in Fig. 4. BusMatrix controls access between the Core system bus and the DMA bus.

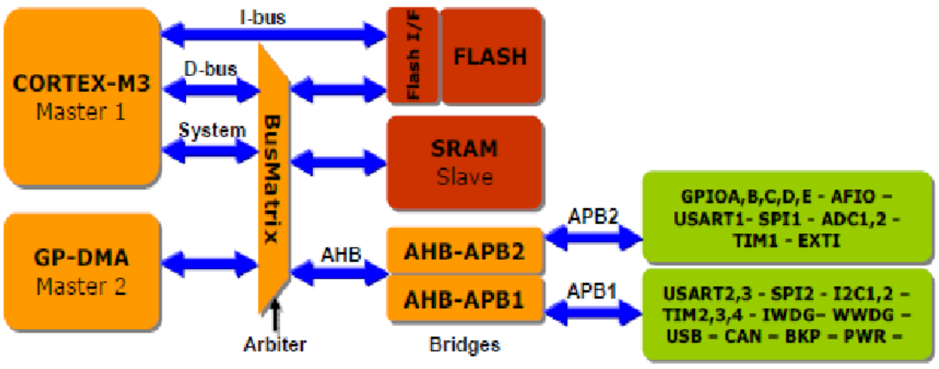

I-bus
D-bus
System
CORTEX-M3
Master 1
GP-DMA
Master 2
BusMatrix
Arbiter
AHB
Flash I/F
FLASH
SRAM
Slave
AHB-APB2
AHB-APB1
Bridges
APB2
APB1
GPIOA,B,C,D,E - AFIO - USART1- SPI1 - ADC1,2 - TIM1 - EXTI
USART2,3 - SPI2 - I2C1,2 - TIM2,3,4 - IWDG- WWDG - USB - CAN - BKP - PWR -

Fig. 2.8: System architecture

The architecture has four master pieces and three slave parts, which are listed below.

Masters:

Cortex-M3 ICode bus (I-bus): In order to do prefeching, it connects the Cortex M3 core to the Flash memory instruction.

It connects the Cortex-M3 core to the Flash memory Data interface via the D-bus (DCode bus).

The S-bus (System bus) connects the Cortex-M3 core peripheral bus to a BoxMatrix to regulate the DMA and Core arbitration.

GP-DMA bus (General Purpose DMA): It connects the CPU (Central Processing Unit) and DMA to the Flash memory, SRAM, and peripherals via BoxMatrix to allow them to communicate.

Slaves:

SRAM stored internally

Internal Flash Storage

Bridge from AHB (Advanced High Performance Bus) to APB (Advanced Peripheral Bus): This bridge separates the AHP bus into two halves, APB1 and APB2. APB1 is for peripherals with a frequency of 36 MHz, and APB2 is for peripherals with a frequency of 72 MHz..

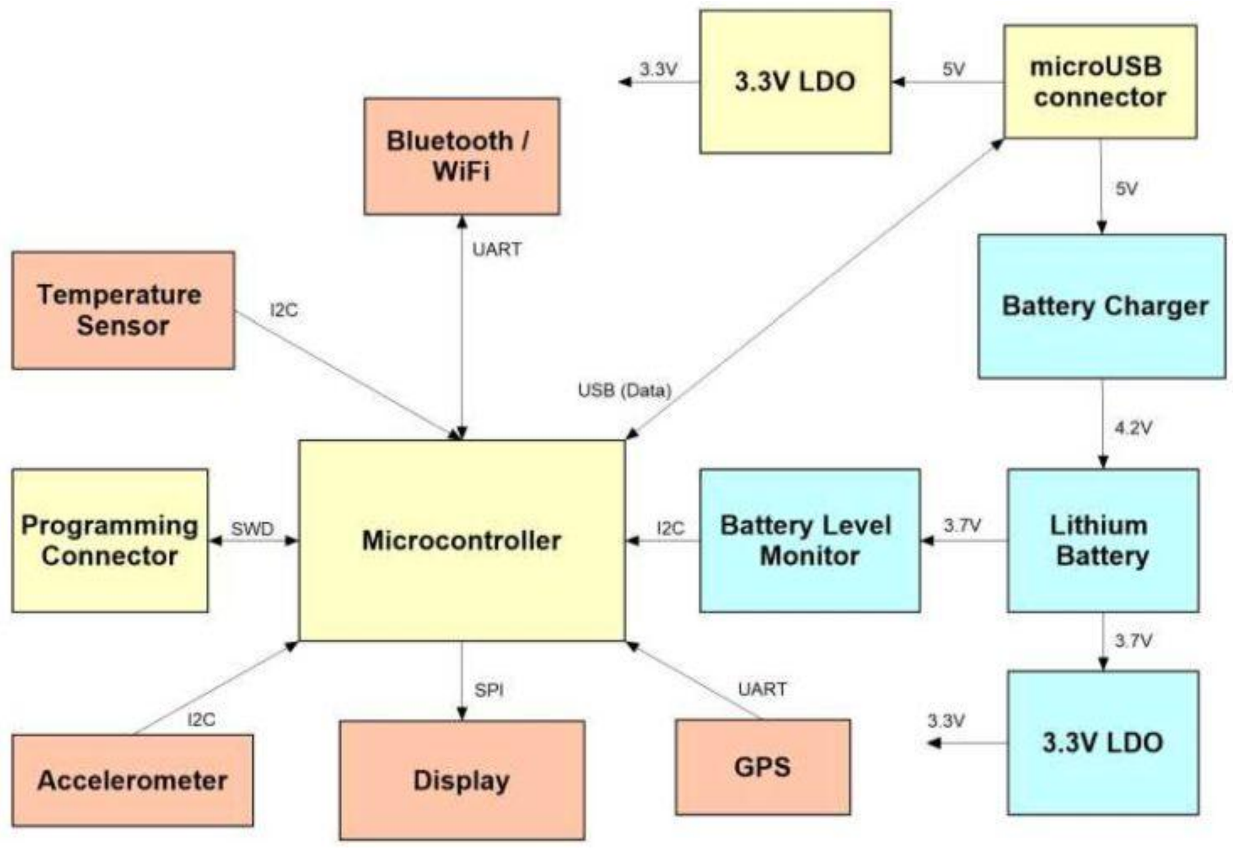


Fig. 2.9: Block diagram of PCB based on STM32

2.11.2 Microcontroller Configuration

The microcontroller setup determines the frequency clock for the CPU and peripherals of the microcontroller. Configuration determines the kind of oscillator used by the microcontroller and the precise frequency for each component. It also specifies the microcontroller's pins for peripherals like as inputs/outputs, operating modes, and so on. It simply conFig.s the peripherals that are required and selected for the project.

The application should setup the CPU, NVIC, GPIO, ADC, TIM1, and USB in order to execute the microcontroller in this job.

To setup the microcontroller, each pin, port, or peripheral that will be used during the project must be defined. For example, if it is chosen to use a USB peripheral in a project, all USB definitions must be examined. To apply for the project, it must be defined which pins of the microcontroller correspond to the USB, the USB clock must be defined, and the USB peripheral must be active. All of these points must be provided in the microcontroller's setup section.

The configuration section of the microcontroller program is always run in advance by the program before anything else. Microcontroller configuration is executed after each microcontroller reset; this implies that the microcontroller configuration is rebooted whenever the microcontroller is reset. The MCU setup is further explained in the next section.

Configuration of RCC

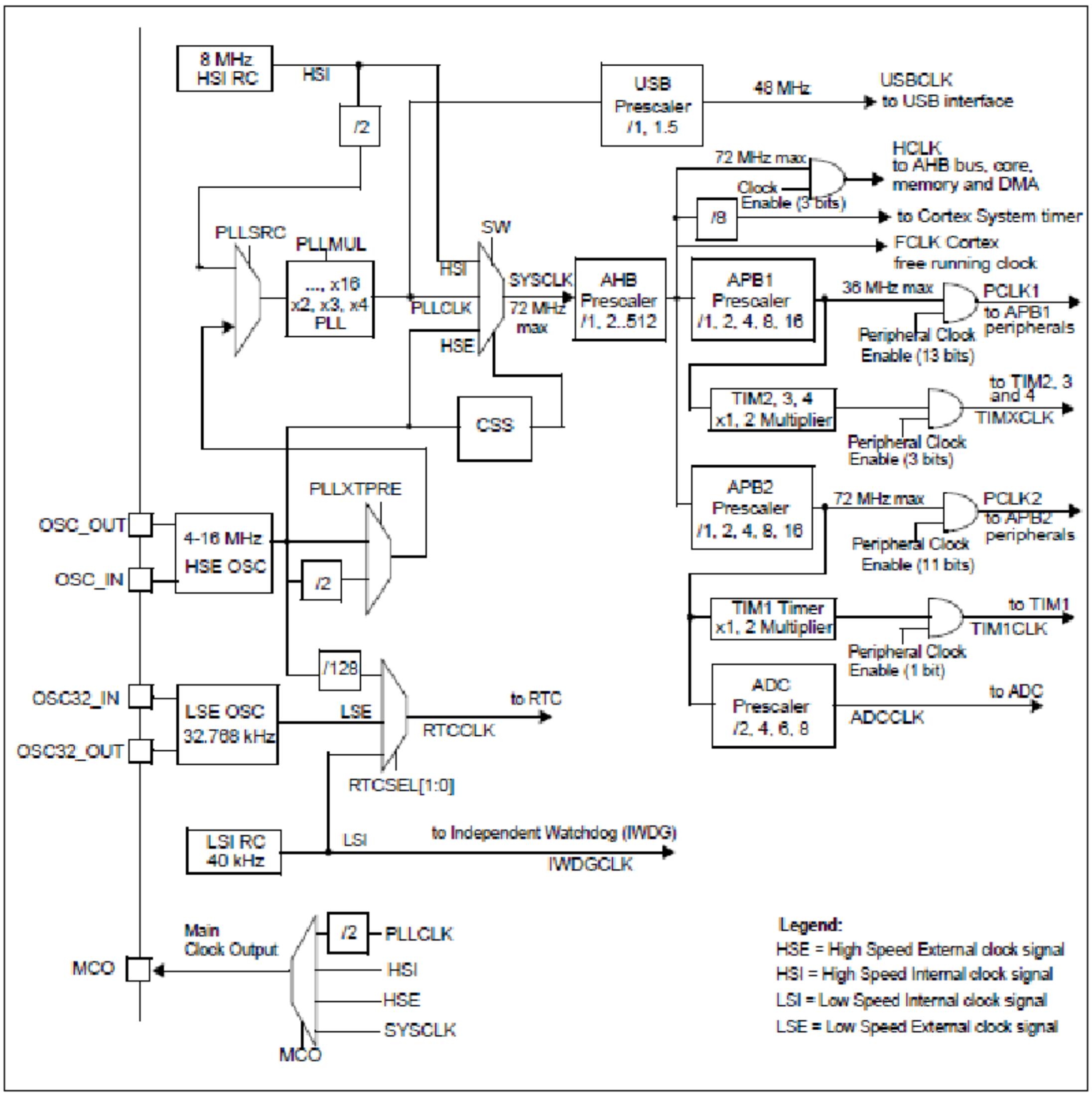
The first step in the configuration prepares the system clock for the microcontroller, which is important because the system clock turns on the microcontroller processor and is also required to activate the peripherals (each peripheral requires clock to function). It is possible to operate the system clock by configuring the RCC. This section determines which oscillator may be used with the microcontroller.

Because this microcontroller is made up of several types of source clocks (as explained in part 2.2.6Clock system (SYSCLK), the source clock for the microcontroller (CPU and all peripherals) should be defined in the RCC configuration. It is critical to pay attention to the

Before configuring clock sources for microcontroller components, create a microcontroller clock structure. In the microcontroller STM32 documentation, the clock structure is referred to as Clock Tree. Fig. 6 depicts the microcontroller's clock tree. It should be noted that any clock for any peripheral must be conFig.d in accordance with this map.

Because it is feasible to attain 72 MHz (the maximum system clock frequency) if HSE is used as a PLL clock input, the High Speed External oscillator is defined as the source clock for the microcontroller (in this work). Another reason is that it can produce highly accurate results.

MCU clock frequency. As a result, the HSE is favored as the source clock frequency for the CPU and each peripheral in the microcontroller.



8 MHz HSI RC
HSI
/2
USB Prescaler /1, 1.5
48 MHz
USBCLK to USB interface
72 MHz max
Clock Enable (3 bits)
HCLK to AHB bus, core, memory and DMA
/8
to Cortex System timer
FCLK Cortex free running clock
PLLSRC
PLLMUL
..., x16 x2, x3, x4 PLL
SW
HSI
PLLCLK
HSE
SYSCLK
72 MHz max
AHB Prescaler /1, 2..512
APB1 Prescaler /1, 2, 4, 8, 16
36 MHz max
PCLK1 to APB1 peripherals
Peripheral Clock Enable (13 bits)
CSS
TIM2, 3, 4 x1, 2 Multiplier
to TIM2, 3 and 4
TIMXCLK
Peripheral Clock Enable (3 bits)
PLLXTPRE
OSC_OUT
OSC_IN
4-16 MHz HSE OSC
/2
APB2 Prescaler /1, 2, 4, 8, 16
72 MHz max
PCLK2 to APB2 peripherals
Peripheral Clock Enable (11 bits)
TIM1 Timer x1, 2 Multiplier
to TIM1
TIM1CLK
Peripheral Clock Enable (1 bit)
/128
OSC32_IN
OSC32_OUT
LSE OSC 32.768 kHz
LSE
to RTC
RTCCLK
RTCSEL[1:0]
ADC Prescaler /2, 4, 6, 8
to ADC
ADCCLK
LSI RC 40 kHz
LSI
to Independent Watchdog (IWDG)
IWDGCLK
Main Clock Output
MCO
/2
PLLCLK
HSI
HSE
SYSCLK
MCO
Legend:
HSE = High Speed External clock signal
HSI = High Speed Internal clock signal
LSI = Low Speed Internal clock signal
LSE = Low Speed External clock signal

Fig. 2.10: Clock Tree for the microcontroller STM32F103xx

Following this configuration, the CPU and peripheral clock frequencies may be conFig.d.

The CPU may then begin to operate, and the peripherals are ready for usage.

Fig. 6 depicts the usage of the HSE oscillator as a PLL clock input and the output of the PLL stands for SYSCLK (System Clock Source). Because each peripheral uses a different frequency clock, the SYSCLK frequency to be adjusted for each portion of the microcontroller.

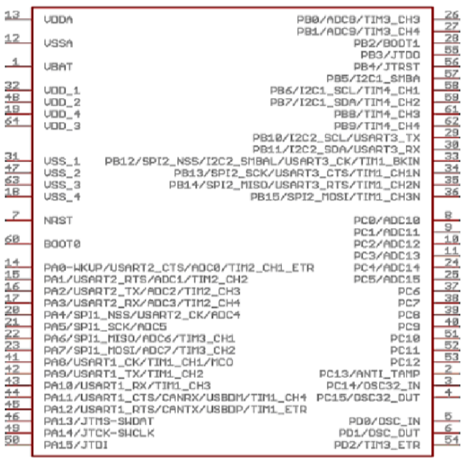
The frequency clock for all of these parts PLLCLK (PLL Clock), SYSCLK, PCLK2 (Internal APB2 clock), PCLK1 (Internal APB1 clock), and ADCCLK (ADC Clock) should be set in this configuration since they prepare clock frequency for all peripherals.

Please keep in mind that RCC Configuration is the most significant microcontroller configuration (it has the highest importance).

Configuration of GPIOs

In order to apply for the microcontroller during the work, the GPIO configuration chooses and enables the microcontroller ports or any pins of the microcontroller. This configuration specifies the inputs and outputs, as well as the condition of each port or pin.

Fig. 7 depicts the Pins Configuration of the STM32F103 microcontroller. This pins configuration is necessary for configuring the microcontroller ports or pins for programming the microcontroller and should be taken into account. Fig. 7 shows the number of pins and their specifications, which is very important for customizing the microcontroller pins.



13 VDDA
12 VSSA
1 VBAT
32 VDD_1
48 VDD_2
19 VDD_4
64 VDD_3
31 VSS_1
47 VSS_2
63 VSS_3
18 VSS_4
7 NRST
60 BOOT0
14 PA0-WKUP/USART2_CTS/ADC0/TIM2_CH1_ETR
15 PA1/USART2_RTS/ADC1/TIM2_CH2
16 PA2/USART2_TX/ADC2/TIM2_CH3
17 PA3/USART2_RX/ADC3/TIM2_CH4
20 PA4/SPI1_NSS/USART2_CK/ADC4
21 PA5/SPI1_SCK/ADC5
22 PA6/SPI1_MISO/ADC6/TIM3_CH1
23 PA7/SPI1_MOSI/ADC7/TIM3_CH2
41 PA8/USART1_CK/TIM1_CH1/MCO
42 PA9/USART1_TX/TIM1_CH2
43 PA10/USART1_RX/TIM1_CH3
44 PA11/USART1_CTS/CANRX/USBDM/TIM1_CH4
45 PA12/USART1_RTS/CANTX/USBDP/TIM1_ETR
46 PA13/JTMS-SWDAT
49 PA14/JTCK-SWCLK
50 PA15/JTDI
PB0/ADC8/TIM3_CH3 26
PB1/ADC9/TIM3_CH4 27
PB2/BOOT1 28
PB3/JTDO 55
PB4/JTRST 56
PB5/I2C1_SMBA 57
PB6/I2C1_SCL/TIM4_CH1 58
PB7/I2C1_SDA/TIM4_CH2 59
PB8/TIM4_CH3 61
PB9/TIM4_CH4 62
PB10/I2C2_SCL/USART3_TX 29
PB11/I2C2_SDA/USART3_RX 30
PB12/SPI2_NSS/I2C2_SMBAL/USART3_CK/TIM1_BKIN 33
PB13/SPI2_SCK/USART3_CTS/TIM1_CH1N 34
PB14/SPI2_MISO/USART3_RTS/TIM1_CH2N 35
PB15/SPI2_MOSI/TIM1_CH3N 36
PC0/ADC10 8
PC1/ADC11 9
PC2/ADC12 10
PC3/ADC13 11
PC4/ADC14 24
PC5/ADC15 25
PC6 37
PC7 38
PC8 39
PC9 40
PC10 51
PC11 52
PC12 53
PC13/ANTI_TAMP 2
PC14/OSC32_IN 3
PC15/OSC32_OUT 4
PD0/OSC_IN 5
PD1/OSC_OUT 6
PD2/TIM3_ETR 54

Fig. 2.11: Pins Configuration of the STM32F103

SECTION 3
DESIGNING A CYBER-PROTECTED WORKPLACE

3.1. Choice of system architecture and components.

Choosing an STM32 microcontroller for the development of a cyber-secure workplace can be justified for several important reasons:

1. Low cost:
   - One of the main advantages of the STM32 microcontroller is its affordable cost. This is especially important in mass production, where the cost of components can have a significant impact on project costs.
   - For example, as you pointed out, some STM32 microcontroller models can be had for as little as $2, making them competitive and cost-effective for projects on a tight budget.
2. High performance:
   - STM32 microcontrollers are known for their high performance and wide functional range. They have powerful computing resources, which allows you to implement complex functions without delays and interruptions in the system.
3. A wide selection of models:
   - STM32 offers a variety of models and series, which allows you to choose a suitable microcontroller for specific tasks. You can choose a model with different characteristics, the number of built-in resources and support for different communication interfaces.
4. Ecosystem and support:
   - STM32 is popular and has a large developer community, providing access to many resources such as documentation, online forums, and libraries.
   - STMicroelectronics, the manufacturer of STM32 microcontrollers, also provides professional technical support and development tools that facilitate fast and convenient project development.
5. Protection and security:

- The STM32 offers built-in security features such as hardware encryption, buffer overflow protection, access control, and other means to ensure data integrity and privacy.
- STMicroelectronics is also actively working to support cyber defense, making the STM32 an excellent choice for developing cyber-protected devices and systems.

The ideal option would be a budget board based on the STM32F103C8T6 microcontroller.

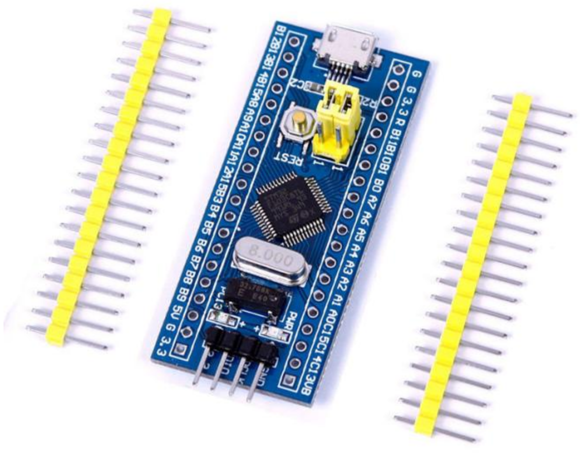


Fig. 3.1: STM32F103C8T6

In order to upload the firmware and play with debugging, you will also need a programmer. For starters, and for further use, a Chinese clone of the ST-LINK V2 programmer is ideal.

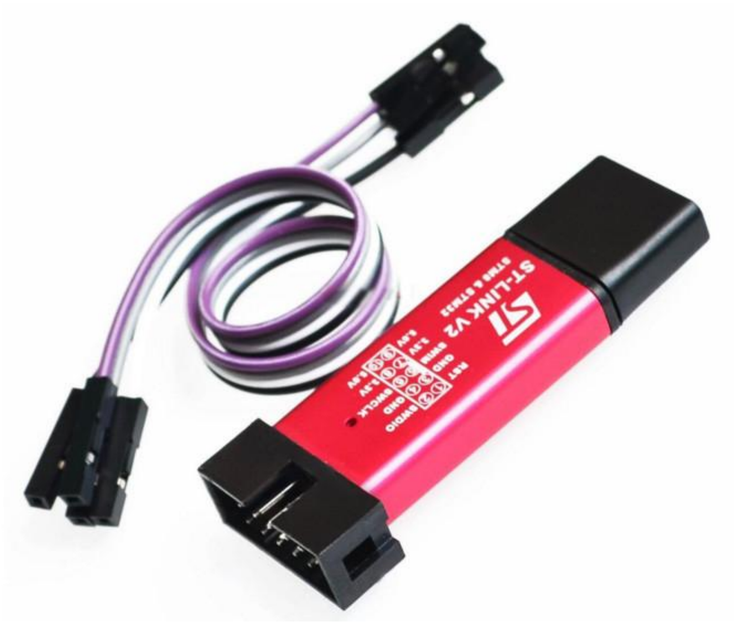


Fig. 3.2: ST-Link V2

3.2. Overview of the Cube MX development environment

After completely installing the necessary software, we can begin creating the project.

To do this, let's launch Cube MX.

In the window that appears, click on the “ACCESS TO MCU SELECTOR” button.

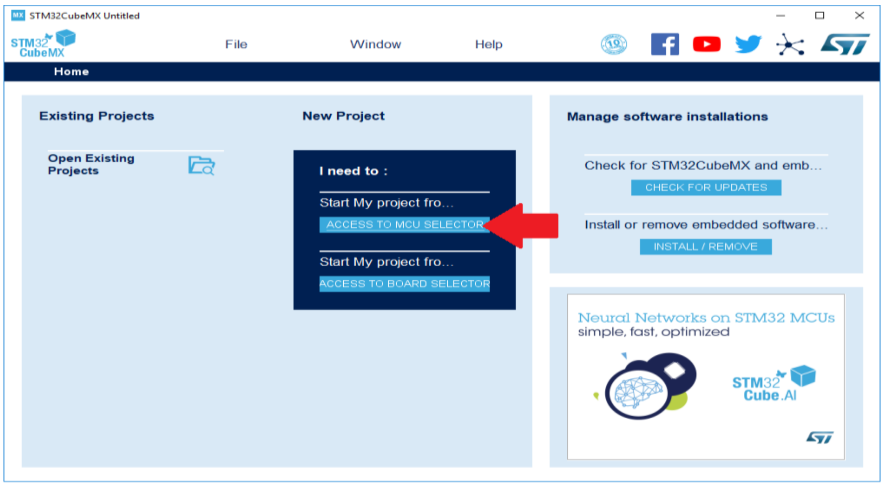
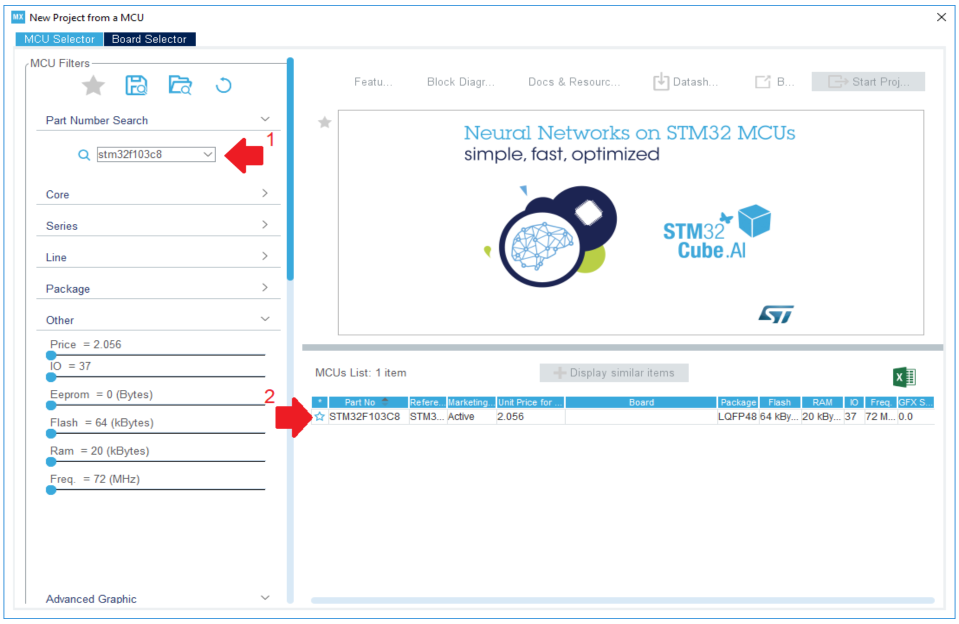


Fig. 3.3: Creating a new project

Our target board has an STM32F103C8T6 microcontroller installed.

Enter its name in the search bar and double-click to select the only option found.

The same table shows the main contents of our MK (64 kilobytes of flash, 20 kilobytes of RAM).



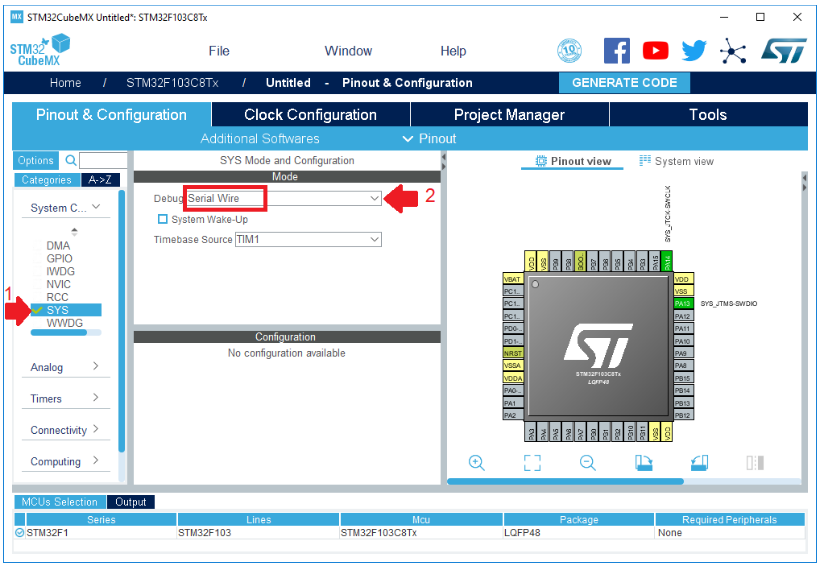
*	Part No	Refere...	Marketing...	Unit Price for ...	Board	Package	Flash	RAM	IO	Freq.	GFX S...
☆	STM32F103C8	STM3...	Active	2.056		LQFP48	64 kBy...	20 kBy...	37	72 M...	0.0

Fig. 3.4: Main contents of MK

A schematically depicted controller body with legs spread in different directions appeared in front of us.

At this stage, it is necessary to select a method for connecting the debugger.

To do this, on the Pinout & Configuration tab in the left menu, select the SYS item and in it, in the drop-down list called “Debug”, set the value to Serial Wire.



Debug Serial Wire

System Wake-Up

Timebase Source TIM1

No configuration available

Series	Lines	Mcu	Package	Required Peripherals
STM32F1	STM32F103	STM32F103C8Tx	LQFP48	None

Fig. 3.5: Connecting a debuger

The easiest way to check the connection of the board is to blink the LED. To do this, you must first find out which leg it is connected to.

The electrical circuit diagram will help us with this:

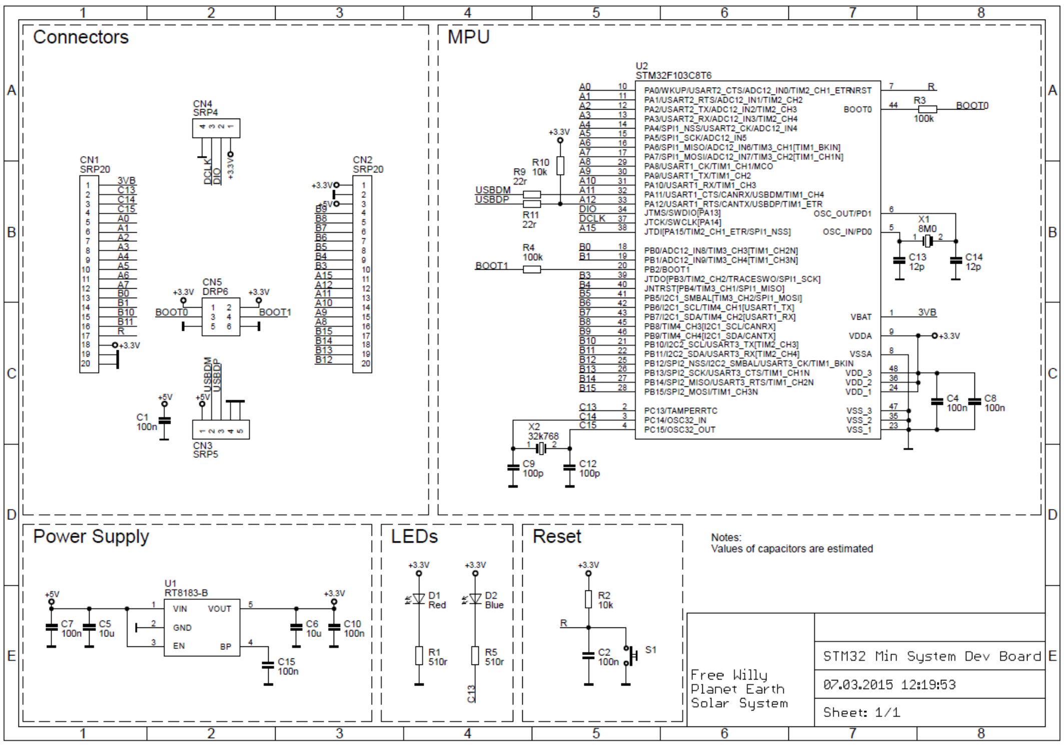


Fig. 3.6: STM electrical circuit diagram

or a more colorful and easy to understand board pinout

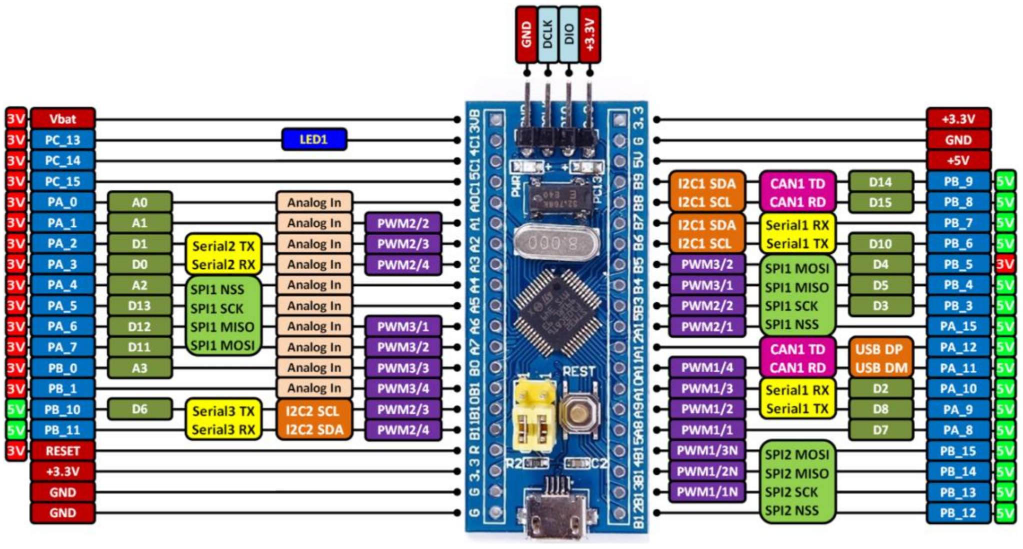

GND
DCLK
DIO
+3.3V
3V Vbat
3V PC_13 LED1
3V PC_14
3V PC_15
3V PA_0 A0 Analog In
3V PA_1 A1 Analog In PWM2/2
3V PA_2 D1 Serial2 TX Analog In PWM2/3
3V PA_3 D0 Serial2 RX Analog In PWM2/4
3V PA_4 A2 SPI1 NSS Analog In
3V PA_5 D13 SPI1 SCK Analog In
3V PA_6 D12 SPI1 MISO Analog In PWM3/1
3V PA_7 D11 SPI1 MOSI Analog In PWM3/2
3V PB_0 A3 Analog In PWM3/3
3V PB_1 Analog In PWM3/4
5V PB_10 D6 Serial3 TX I2C2 SCL PWM2/3
5V PB_11 Serial3 RX I2C2 SDA PWM2/4
3V RESET
+3.3V
GND
GND
+3.3V
GND
+5V
I2C1 SDA CAN1 TD D14 PB_9 5V
I2C1 SCL CAN1 RD D15 PB_8 5V
I2C1 SDA Serial1 RX PB_7 5V
I2C1 SCL Serial1 TX D10 PB_6 5V
PWM3/2 SPI1 MOSI D4 PB_5 3V
PWM3/1 SPI1 MISO D5 PB_4 5V
PWM2/2 SPI1 SCK D3 PB_3 5V
PWM2/1 SPI1 NSS PA_15 5V
CAN1 TD USB DP PA_12 5V
PWM1/4 CAN1 RD USB DM PA_11 5V
PWM1/3 Serial1 RX D2 PA_10 5V
PWM1/2 Serial1 TX D8 PA_9 5V
PWM1/1 D7 PA_8 5V
PWM1/3N SPI2 MOSI PB_15 5V
PWM1/2N SPI2 MISO PB_14 5V
PWM1/1N SPI2 SCK PB_13 5V
SPI2 NSS PB_12 5V
REST

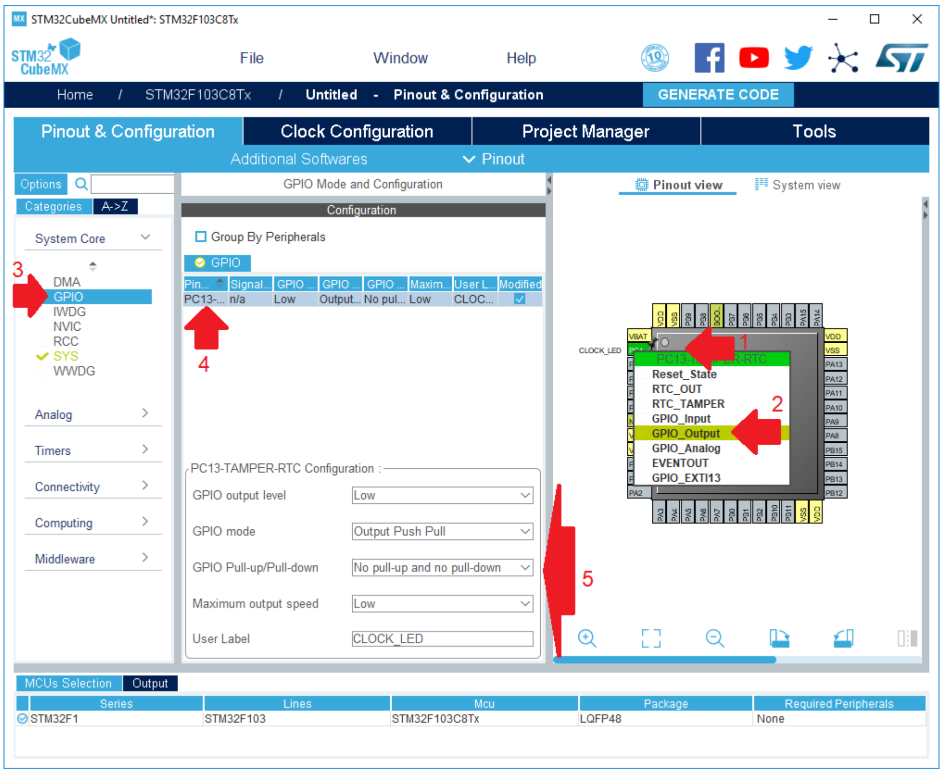
Fig. 3.7: STM 32 electrical circuit diagram

The LED you are looking for is located on the PC13 leg.

Accordingly, it is necessary to conFig. this pin to operate in output mode.

For this:

1. Finding the output on the mnemonic diagram
2. Right-click on it and select “GPIO_Output” from the drop-down menu
3. Go to the GPIO menu,
4. Select PC13 from the list
5. We fill out the PC13-TAMPER-RTC Configuration table in accordance with the screenshot, we are especially interested in the GPIO mode and User Label parameters



Pin...	Signal...	GPIO ...	GPIO ...	GPIO ...	Maxim...	User L...	Modified
PC13-...	n/a	Low	Output...	No pul...	Low	CLOC...	✓

PC13-TAMPER-RTC Configuration :

GPIO output level	Low
GPIO mode	Output Push Pull
GPIO Pull-up/Pull-down	No pull-up and no pull-down
Maximum output speed	Low
User Label	CLOCK_LED

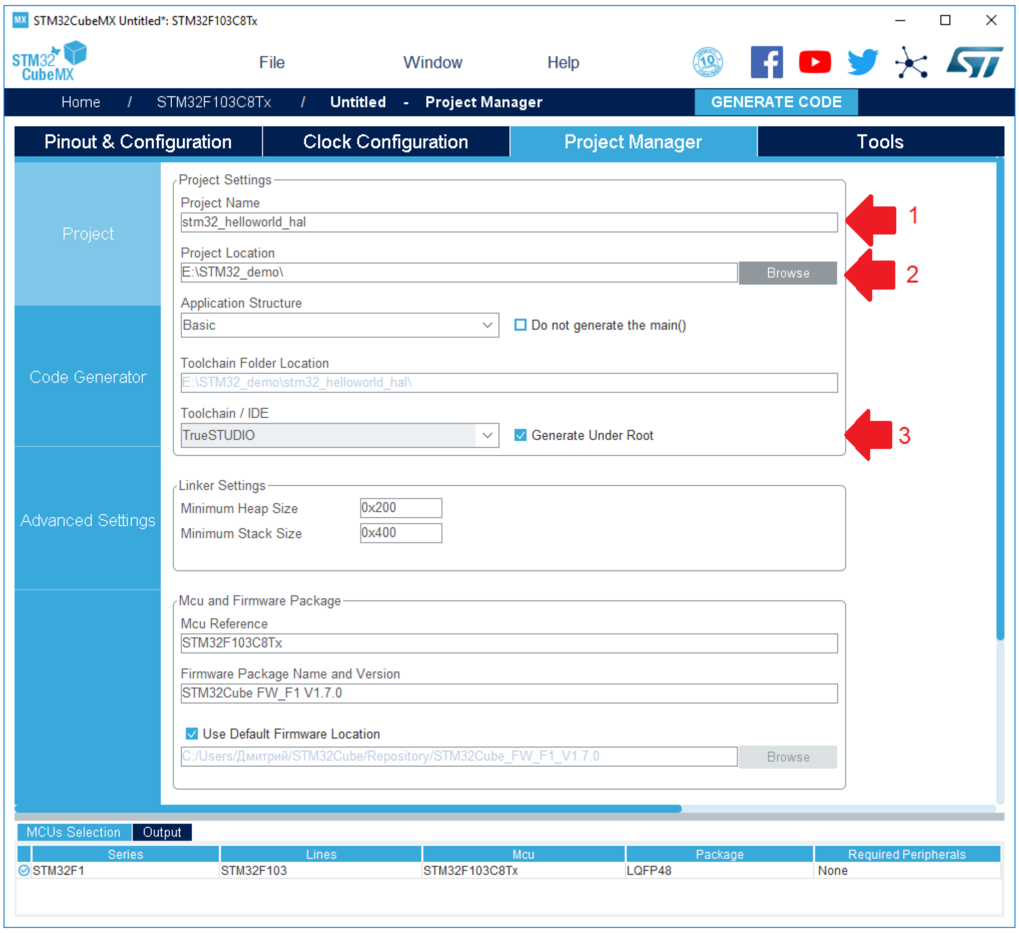
Series	Lines	Mcu	Package	Required Peripherals
STM32F1	STM32F103	STM32F103C8Tx	LQFP48	None

Fig. 3.8: Output mode configuration

Go to the Project Manager tab, under the Project tab.

Be sure to fill in the following parameters:

1. Project name (it is better to use only Latin letters)
2. The directory in which the project will be created (it is also better to use only the Latin alphabet)
3. IDE in which you plan to work on the project (we plan to use TrueSTUDIO)


STM32CubeMX Untitled*: STM32F103C8Tx

Home / STM32F103C8Tx / Untitled - Project Manager

Project Settings

Project Name: stm32_helloworld_hal

Project Location: E:\STM32_demo\

Application Structure: Basic

☐ Do not generate the main()

Toolchain Folder Location: E:\STM32_demo\stm32_helloworld_hal\

Toolchain / IDE: TrueSTUDIO

☑ Generate Under Root

Linker Settings

Minimum Heap Size: 0x200

Minimum Stack Size: 0x400

Mcu and Firmware Package

Mcu Reference: STM32F103C8Tx

Firmware Package Name and Version: STM32Cube FW_F1 V1.7.0

☑ Use Default Firmware Location

C:/Users/Дмитрий/STM32Cube/Repository/STM32Cube_FW_F1_V1.7.0

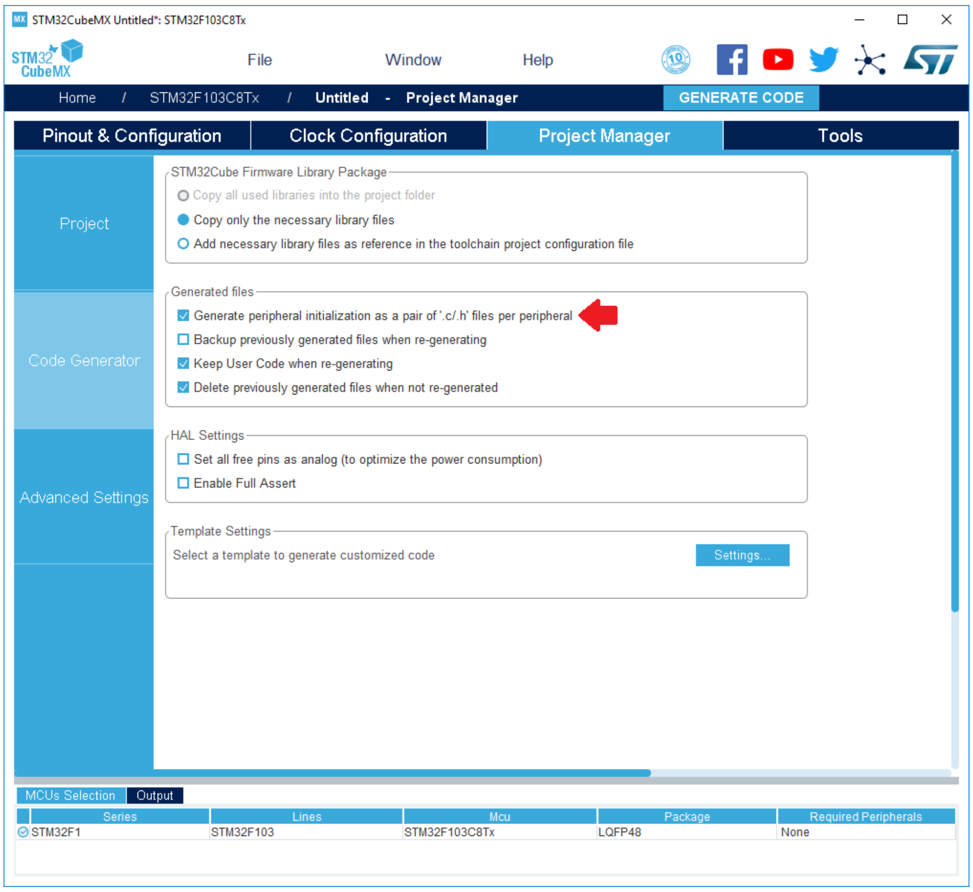
Series	Lines	Mcu	Package	Required Peripherals
STM32F1	STM32F103	STM32F103C8Tx	LQFP48	None

Fig. 3.9: Generating a code

We go down below, under the Code Generator tab. Here, be sure to check the Generate peripheral initialization as pair option. This way we get a more structured project, in which each type of peripheral has its own pair of C and H files.

There's one last step left. Advanced Settings subtab.

1. Selecting the HAL library type for all peripheral modules
2. We assemble the project with the current settings



STM32Cube Firmware Library Package
- ○ Copy all used libraries into the project folder
- ● Copy only the necessary library files
- ○ Add necessary library files as reference in the toolchain project configuration file

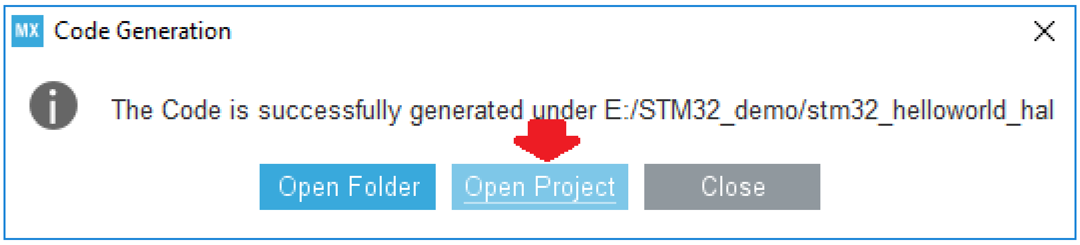
Generated files
- ☑ Generate peripheral initialization as a pair of '.c/.h' files per peripheral
- ☐ Backup previously generated files when re-generating
- ☑ Keep User Code when re-generating
- ☑ Delete previously generated files when not re-generated

HAL Settings
- ☐ Set all free pins as analog (to optimize the power consumption)
- ☐ Enable Full Assert

Template Settings
Select a template to generate customized code Settings...

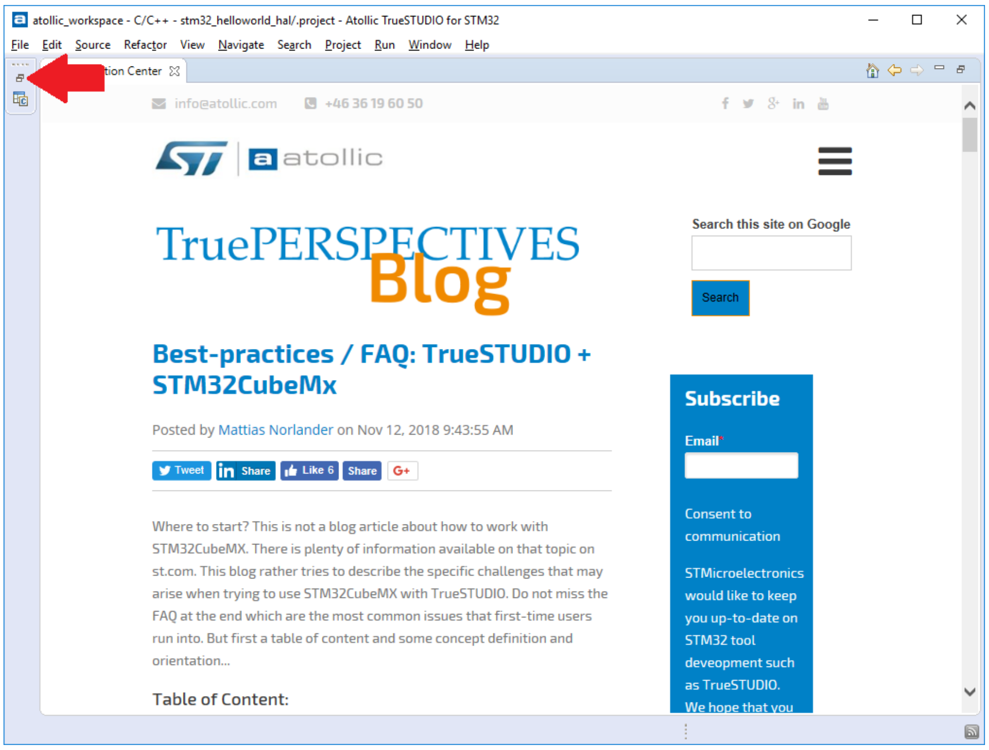
Series	Lines	Mcu	Package	Required Peripherals
STM32F1	STM32F103	STM32F103C8Tx	LQFP48	None

Fig. 3.10: Next step of generating a code



The Code is successfully generated under E:/STM32_demo/stm32_helloworld_hal

Fig. 3.11: The code is generated


atollic_workspace - C/C++ - stm32_helloworld_hal/.project - Atollic TrueSTUDIO for STM32
File Edit Source Refactor View Navigate Search Project Run Window Help
tion Center
info@atollic.com
+46 36 19 60 50
atollic
Search this site on Google
Search
Tweet
Share
Like 6
Share
G+

TruePERSPECTIVES Blog

Best-practices / FAQ: TrueSTUDIO + STM32CubeMx

Posted by Mattias Norlander on Nov 12, 2018 9:43:55 AM

Where to start? This is not a blog article about how to work with STM32CubeMX. There is plenty of information available on that topic on st.com. This blog rather tries to describe the specific challenges that may arise when trying to use STM32CubeMX with TrueSTUDIO. Do not miss the FAQ at the end which are the most common issues that first-time users run into. But first a table of content and some concept definition and orientation...

Table of Content:

Subscribe

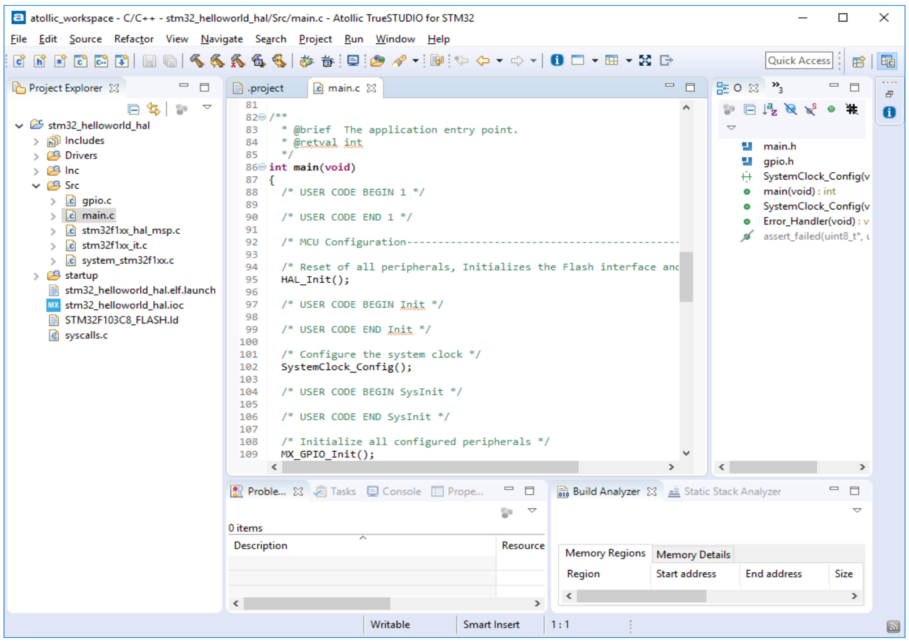
Email*

Consent to communication

STMicroelectronics would like to keep you up-to-date on STM32 tool devepoment such as TrueSTUDIO. We hope that you

Fig. 3.12: Opening a code

Let's find there the main.c file and the int main(void) function:



```
/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration----------------------------------------------------------

  /* Reset of all peripherals, Initializes the Flash interface and
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
```

Fig. 3.13: Generated code

SECTION 4

DEVELOPMENT OF LABORATORY WORKS ON STM32 MICROCONTROLLER PROGRAMMING

Laboratory work №1.

Project generation and PWM work programming in conjunction with CubeMX and Atollic TrueStudio

Purpose of work: Learn to work with PWM using CubeMX and Atollic TrueStudio - conFig. the microcontroller located on the STM Discovery board, and execute typical operations of work with the built-in PWM module - regulation of the light of LEDs.

1.1. Brief theoretical information
Pulse width modulation (PWM) is a method of controlling the average voltage across a load by changing the duty cycle of pulses controlled by a switch. Basically, microcontrollers allow you to generate digital PWM of various frequencies. Since outputting a PWM signal is not the primary function of the pins that are connected to the LEDs, it is necessary to conFig. them accordingly. To do this, you need to set the pins to perform alternative functions. The generation of PWM in them is associated with the use of additional timer modes [12].

Before connecting the device, PWM parameters such as frequency and duty cycle are known. To calculate them in STM32 controllers, it is necessary to determine the value of the prescaler and the automatically loaded value in the ARR (Auto-Reload Register) register. The calculation of the value that should be written

$$PSC = \frac{TIMxCLK}{TIMxCNT} - 1;$$

to the prescaler is performed as follows:

where PSC is the value of the prescaler,

TIMxCLK is the input frequency of the timer, TIMxCNT is the frequency of the counter.

To obtain the required output frequency, you should write the values in the ARR register, which is obtained from the following relationship:

$$ARR\_VAL = \frac{TIMxCNT}{TIMx\_out\_freq} - 1$$

where ARR_VAL is the value to write to the ARR register, TIMxCNT is the counter frequency,

TIMx_ out_ freq – desired output PWM frequency.

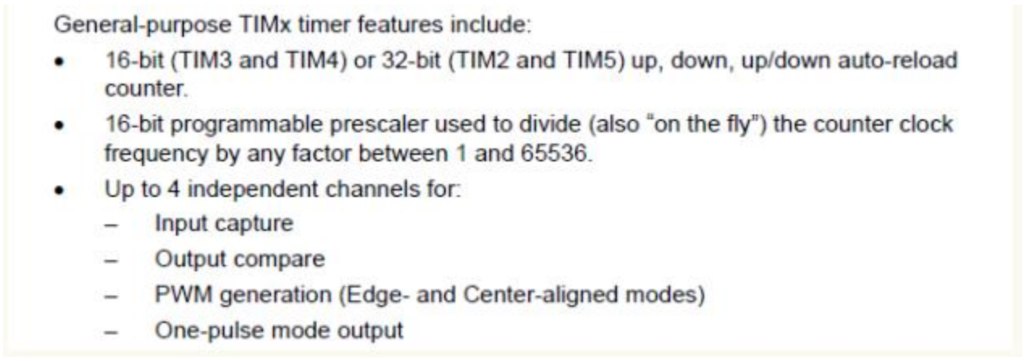
The last step is to set the required duty cycle, which will provide the desired output voltage value. This setting is made using the capture/compare register (CCRx), based on the following relationship:

$$D = \frac{CCRx\_VAL}{ARR\_VAL} * 100\%,$$

where D is the filling factor,

CCRx _VAL – value in the CCRx register (x – register number for a specific line), ARR_VAL – value to write to the ARR register.

A feature of these microcontrollers is that any value that can be described using the allotted number of bits can be written to the prescaler and other registers. Issuing a PWM signal to the output is not the main mode of operation of the port pins, but refers to additional (alternative) modes. Therefore, you first need to set the desired mode in the port settings
As an example of a timer for PWM, we will take timer 4. This timer is a 16-bit. Up to 4 separate channels are available. Also in the technical documentation for the controller it is indicated that the timer can act as a PWM generator


General-purpose TIMx timer features include:

- 16-bit (TIM3 and TIM4) or 32-bit (TIM2 and TIM5) up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also "on the fly") the counter clock frequency by any factor between 1 and 65536.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output

To perform laboratory work, you need to look at what timers are used to generate PWM available in your controller, which pins can the PWM be output to and select that timer which can be output to one of the board's LEDs (they are different on different boards).

Work order:

1. Launch Cube MX,
2. Create a project in Cube MX
3. Firstly, disconnect all the pins of the LED ports, as well as the pin of the button

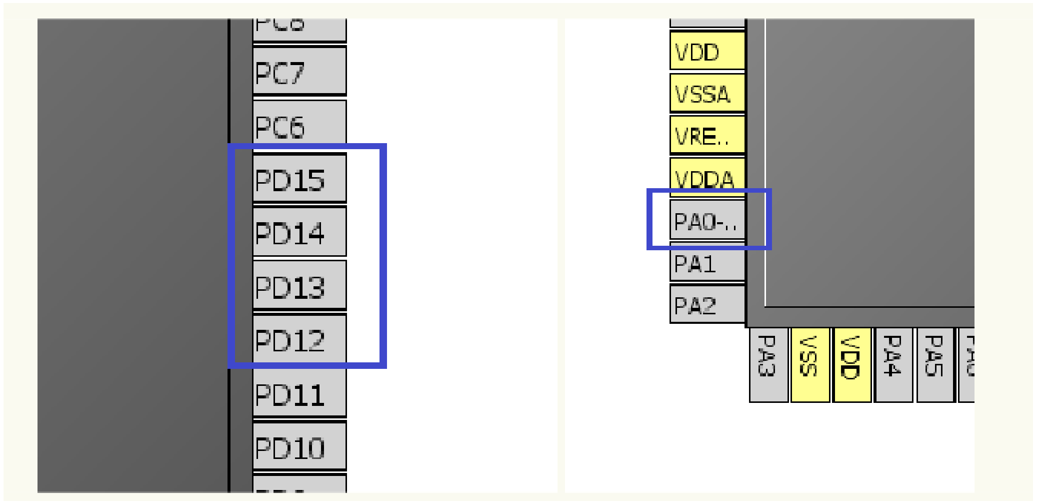
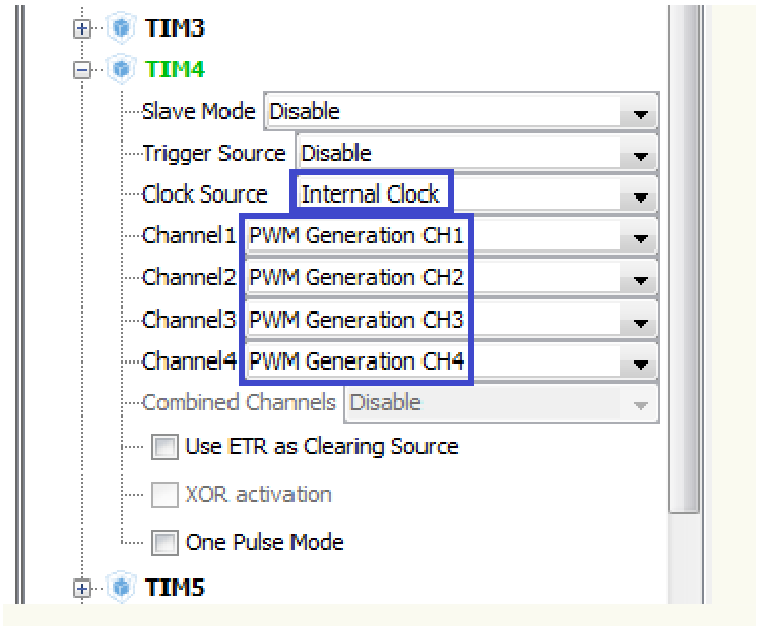


Fig. 4.1: pins I/O

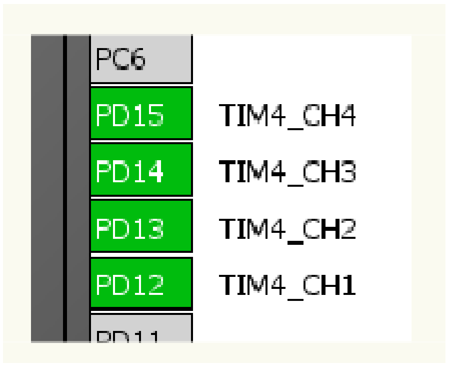
We turn on the timer by selecting internal clocking in the drop-down list, and turn it on. Turn on all channels – all 4, selecting PWM there



TIM3
TIM4
Slave Mode: Disable
Trigger Source: Disable
Clock Source: Internal Clock
Channel1: PWM Generation CH1
Channel2: PWM Generation CH2
Channel3: PWM Generation CH3
Channel4: PWM Generation CH4
Combined Channels: Disable
Use ETR as Clearing Source
XOR activation
One Pulse Mode
TIM5

Fig. 4.2: Clocking configuration

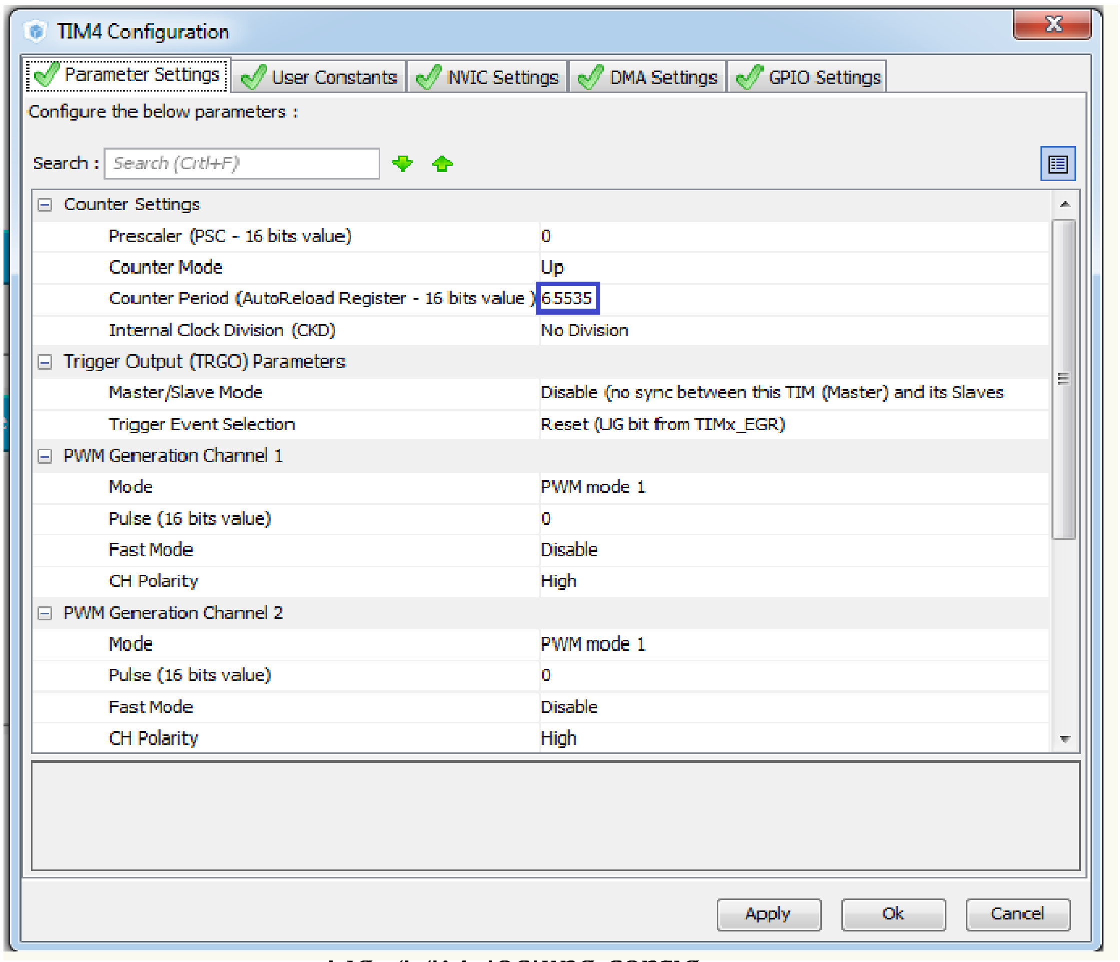
As we can see, the legs of the ports responsible for the LEDs on the board turned on themselves specific alternate mode



PC6
PD15 TIM4_CH4
PD14 TIM4_CH3
PD13 TIM4_CH2
PD12 TIM4_CH1

Fig. 4.3: Legs responsible for LEDs

In the timer settings in Configuration, set the Counter Period parameter to 65535



Configure the below parameters :

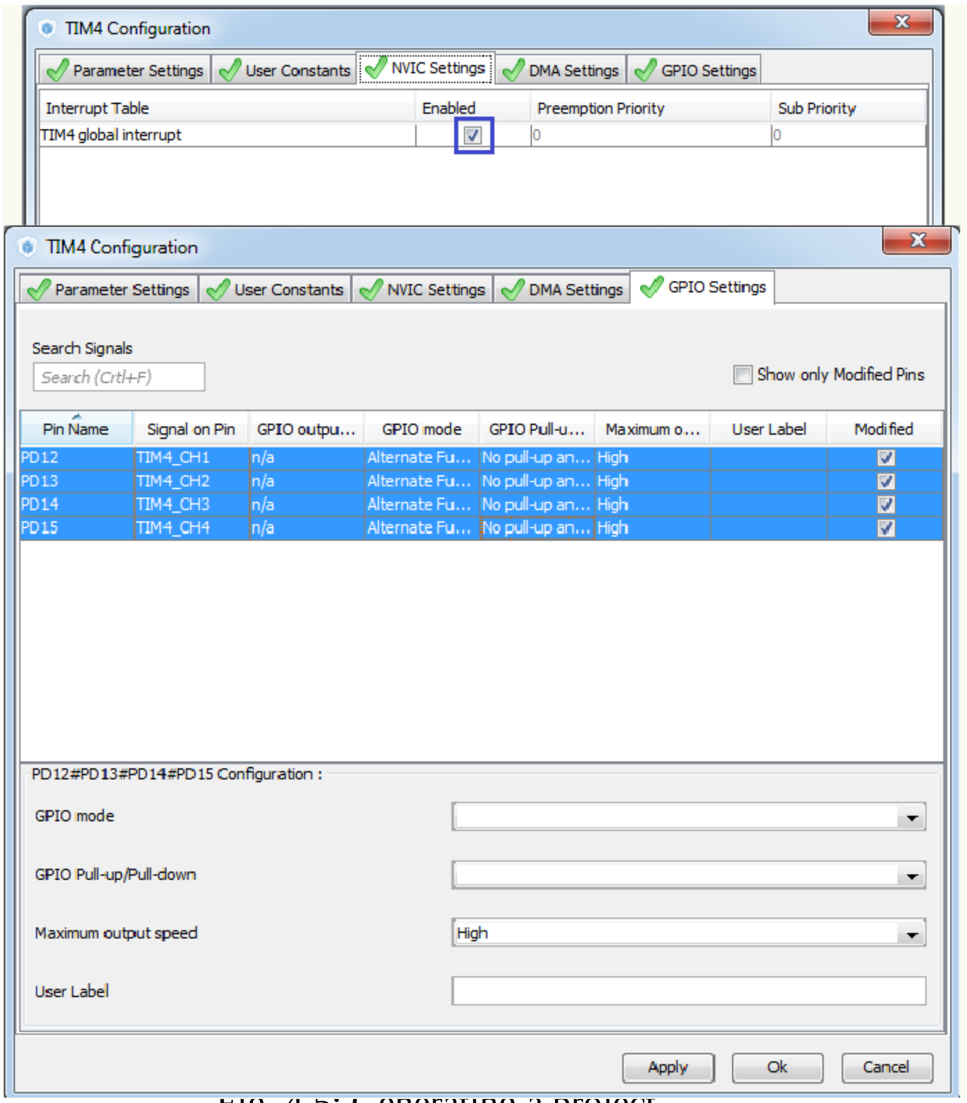
Search : Search (Crtl+F)

Counter Settings	
Prescaler (PSC - 16 bits value)	0
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	65535
Internal Clock Division (CKD)	No Division
Trigger Output (TRGO) Parameters	
Master/Slave Mode	Disable (no sync between this TIM (Master) and its Slaves
Trigger Event Selection	Reset (UG bit from TIMx_EGR)
PWM Generation Channel 1	
Mode	PWM mode 1
Pulse (16 bits value)	0
Fast Mode	Disable
CH Polarity	High
PWM Generation Channel 2	
Mode	PWM mode 1
Pulse (16 bits value)	0
Fast Mode	Disable
CH Polarity	High

Fig. 4.4: Clocking config

Just in case, enable interrupts and set all 4 ports to high speed.

Generating a project


TIM4 Configuration

Interrupt Table	Enabled	Preemption Priority	Sub Priority
TIM4 global interrupt	✓	0	0

TIM4 Configuration

Pin Name	Signal on Pin	GPIO outpu...	GPIO mode	GPIO Pull-u...	Maximum o...	User Label	Modified
PD12	TIM4_CH1	n/a	Alternate Fu...	No pull-up an...	High		✓
PD13	TIM4_CH2	n/a	Alternate Fu...	No pull-up an...	High		✓
PD14	TIM4_CH3	n/a	Alternate Fu...	No pull-up an...	High		✓
PD15	TIM4_CH4	n/a	Alternate Fu...	No pull-up an...	High		✓

PD12#PD13#PD14#PD15 Configuration :

GPIO mode	
GPIO Pull-up/Pull-down	
Maximum output speed	High
User Label	

Fig. 4.5: Generating a project

Below is a program that implements the following effect: LEDs

They light up one by one, and they light up smoothly and go out smoothly too.

/* Initialize all conFig.d peripherals */ MX_GPIO_Init();

MX_TIM4_Init();

```
/* USER CODE BEGIN 2 */
HAL_TIM_PWM_Start(&htim4,                TIM_CHANNEL_1);

HAL_TIM_PWM_Start(&htim4,                TIM_CHANNEL_2);

HAL_TIM_PWM_Start(&htim4,                TIM_CHANNEL_3);

HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_4);
```

```
/* USER CODE END 2 */
```

Also type this to the function MAIN

```
/* USER CODE BEGIN 1 */
uint32_t i,d;
/* USER CODE END 1 */
```

И в бесконечный цикл пишем

```
/* USER CODE BEGIN 3 */
for(i=0;i<=524288;i++)
{
if(i<65536)    TIM4->CCR1=i;
else if ((i>65535)&&(i<131072)) TIM4->CCR1=131071-i; else
if((i>131071)&&(i<196608)) TIM4->CCR2=i-131072; else if
((i>196607)&&(i<262164)) TIM4->CCR2=262164-i; else
if((i>262163)&&(i<327680)) TIM4->CCR3=i-262164; else if
((i>327679)&&(i<393216)) TIM4->CCR3=393216-i; else
if((i>393216)&&(i<458752)) TIM4->CCR4=i-393216;
else TIM4->CCR4=524288-i;
for(d=0;d<300;d++)
{
}
}
}
/* USER CODE END 3 */
```

Test questions and assignments:

1. PWM operating principle, scope of application. 32-bit Cortex kernel of various versions (in the microcontroller installed on the board uses a Cortex-M4 core). Implementation PWM in hardware - without a controller.

2. Implementation of PWM using a microcontroller in STM32. Capture and comparison module + timer

3. Capture/compare register (CCRx), Counter Register (TIMx_CNT)

— counter, Prescaler Register (TIMx_PSC) — prescaler, Auto-Reload Register

(TIMx_ARR) - reset register

4. Explain how to conFig. PWM using CubeMX. How to set the PWM frequency, how change the duty cycle in the program.

Task: Generate initialization code in CubeMX, transfer it to Atollic TrueStudio and using the HAL library, write code that performs the following algorithm: when pressing button PA0, the LED lights up smoothly within N seconds, where N is the number option (number according to the list in the magazine). Compile the project (build the project), when If errors appear, eliminate them, load the code into the debug board and make sure it is correct program operation. When completing the task, you need to take into account that the example given uses a different controller model than on your debug board and accordingly to you you will need to select a different timer module, and you have different legs connected than in example, a different operating frequency.

Laboratory work 2

Working with a display based on HD44780

Purpose of work: Learn to work with a **display based on HD44780**

2.1. Brief theoretical information

What is an integral part of a large number of electronic devices? Certainly,

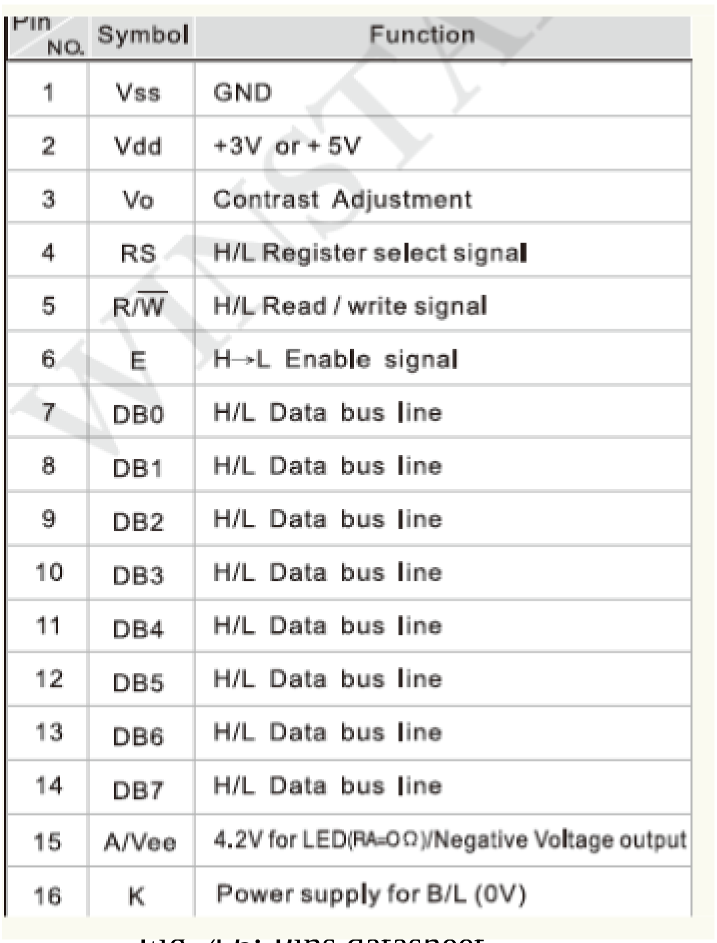
means of indication and graphical output of data. Always more convenient and enjoyable for the user

when the result of the “smart box” can be seen visually. Therefore today we

let's connect a display to the STM32 to display text and numbers - quite a popular display

WH1602 from Winstar, the technique is basically the same for all HD44780-based displays [12].

First, the display must be connected to the controller. Download the datasheet and search WH1602 pinout:



Pin NO.	Symbol	Function
1	Vss	GND
2	Vdd	+3V or + 5V
3	Vo	Contrast Adjustment
4	RS	H/L Register select signal
5	$R/\overline{W}$	H/L Read / write signal
6	E	H→L Enable signal
7	DB0	H/L Data bus line
8	DB1	H/L Data bus line
9	DB2	H/L Data bus line
10	DB3	H/L Data bus line
11	DB4	H/L Data bus line
12	DB5	H/L Data bus line
13	DB6	H/L Data bus line
14	DB7	H/L Data bus line
15	A/Vee	4.2V for LED(RA=0Ω)/Negative Voltage output
16	K	Power supply for B/L (0V)

Fig. 4.6: Pins datasheet

Pins Vss, Vdd and K need to be connected to ground and to power. Pin number 3 is used for contrast adjustment - if we apply +5V there, we will see absolutely nothing, but if If we short-circuit the output to ground, we will admire two rows of black squares. Naturally, this does not suit us, so we need to hang a potentiometer there (variable resistor resistance) to adjust the contrast. Best symbol visibility is provided by a voltage of 0.5-0.7V at this display pin.

The RS pin is already a pin that we ourselves will control using a microcontroller.

A low voltage level (0) on this pin means that a command will now follow, a high level (1) – this means that there will now be data to be written to the display memory.

Pin R/W - it’s clear here, or we read the data (display busy flag, for example), in in this case, this pin is 1, or we write the command/data to the display, then here there are 0 of us.

DB7 – DB0 – data bus.

Pin E is the so-called Enable signal. It is needed to work with the display - to record data or give a command - we need to issue a positive impulse to this pin. That is, the procedure will look like this:

1. On pins RS, R/W, DB7 - DB0 - the necessary signals corresponding to our command.

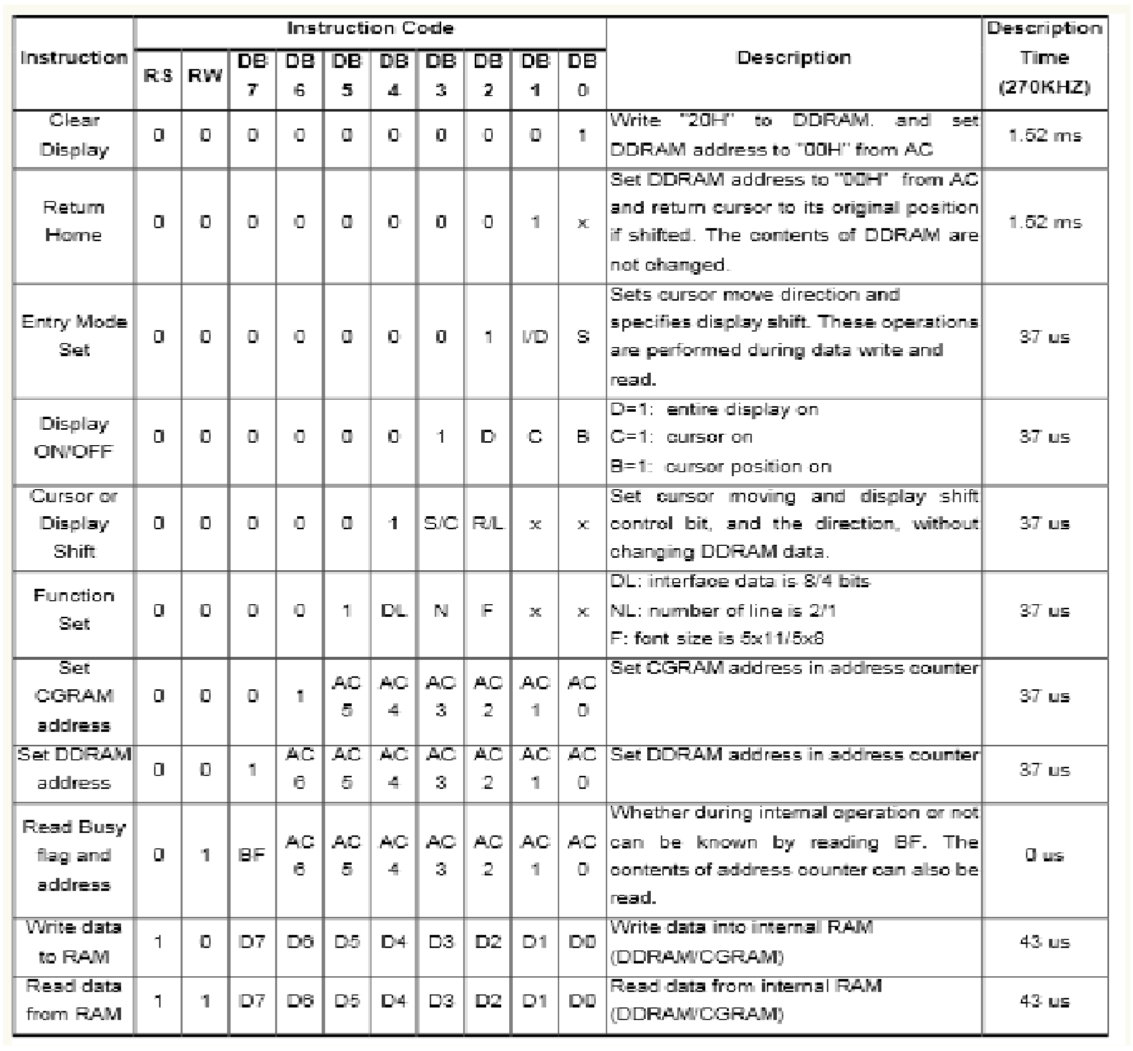
2. Apply one to pin E.

3. Wait (according to the datasheet – at least 150 ns)

4. Apply a low level (0) to pin E.

The A/Vee pin must be supplied with 4.2 V to power the display backlight.

This is how communication with the WH1602 display occurs.



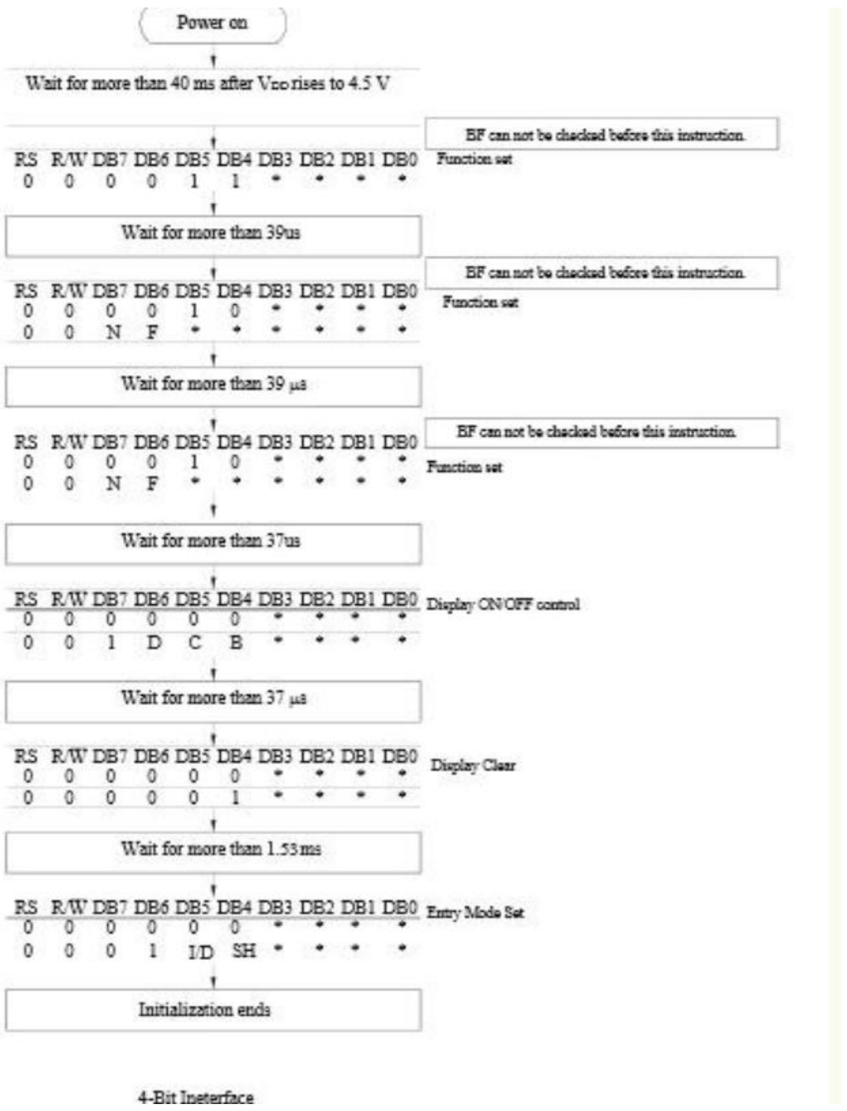
Instruction	Instruction Code										Description	Description Time (270KHZ)
	RS	RW	DB 7	DB 6	DB 5	DB 4	DB 3	DB 2	DB 1	DB 0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRAM. and set DDRAM address to "00H" from AC	1.52 ms
Return Home	0	0	0	0	0	0	0	0	1	x	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	1.52 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	37 us
Display ON/OFF	0	0	0	0	0	0	1	D	C	B	D=1: entire display on C=1: cursor on B=1: cursor position on	37 us
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	x	x	Set cursor moving and display shift control bit, and the direction, without changing DDRAM data.	37 us
Function Set	0	0	0	0	1	DL	N	F	x	x	DL: interface data is 8/4 bits NL: number of line is 2/1 F: font size is 5x11/5x8	37 us
Set CGRAM address	0	0	0	1	AC 5	AC 4	AC 3	AC 2	AC 1	AC 0	Set CGRAM address in address counter	37 us
Set DDRAM address	0	0	1	AC 6	AC 5	AC 4	AC 3	AC 2	AC 1	AC 0	Set DDRAM address in address counter	37 us
Read Busy flag and address	0	1	BF	AC 6	AC 5	AC 4	AC 3	AC 2	AC 1	AC 0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0 us
Write data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM)	43 us
Read data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM)	43 us

Fig. 4.7: Communication with a display

All commands and signals that should be on the corresponding pins are described here

WH1602 for each specific command. For example, we want to clear the display, we look at table, and here is the required command - Clear Display. We supply to pins RS, R/W, DB7, DB6, DB5, DB4, DB3, DB2, DB1 are zeros, and pin DB0 is one. Done! What's next? That's right, one per pin E, then wait a while and lower E to zero again. All, display cleared Only

Before executing the next command, you must pause as indicated in the datasheet for each team. It would be more efficient to poll the busy flag as soon as it is reset to 0 - you can work further. There is also a special command to read this flag, so with this all clear). Let's look at an example using a 4-bit data bus (as you remember there is two options for communicating with the display, namely using an 8-bit bus, or a 4-bit bit), but by and large there are no special features. If in 8-bit mode we transmit the entire command using pins DB7-DB0, then in 4-bit the command is divided into two parts and it is transmitted accordingly in parts (in this mode we use only 4 outputs display – DB7-DB4). So everything is simple and clear. Quite often when working with displays the problem arises primarily with initialization, so download the documentation for display being used and looking for something similar to the set of commands that are needed initialize the display. For example, I use the WH1602 display. Here's the one we're interested in part of the datasheet for it where a 4-bit data bus is used:



Power on

Wait for more than 40 ms after V_{DD} rises to 4.5 V

BF can not be checked before this instruction.

Function set

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	1	*	*	*	*

Wait for more than 39us

BF can not be checked before this instruction.

Function set

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	0	*	*	*	*
0	0	N	F	*	*	*	*	*	*

Wait for more than 39 μs

BF can not be checked before this instruction.

Function set

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	0	*	*	*	*
0	0	N	F	*	*	*	*	*	*

Wait for more than 37us

Display ON/OFF control

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	*	*	*	*
0	0	1	D	C	B	*	*	*	*

Wait for more than 37 μs

Display Clear

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	*	*	*	*
0	0	0	0	0	1	*	*	*	*

Wait for more than 1.53 ms

Entry Mode Set

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	*	*	*	*
0	0	0	1	I/D	SH	*	*	*	*

Initialization ends

4-Bit Ineterface

Fig. 4.8: Initializing a display

To work with the display, a ready-made library using HAL was selected - HD44780 library for Stm32 using hal library [14]. In the downloaded folder lcd_lib library there are 2 files lcd.c And lcd.h. The readme.md file describes definitions that specify the display type

Using 16xN and 20xN screens

Set macros for different screen sizes in lcd.h. By default, 16xN is enabled.

// #define LCD20xN // For 20xN LCDs

#define LCD16xN // For 16xN LCDs

As an example, here is an example of how to use the library for 4-bit output:

4bit example

Lcd_PortType ports[] = {

D4_GPIO_Port, D5_GPIO_Port, D6_GPIO_Port, D7_GPIO_Port

};

Lcd_PinType pins[] = {D4_Pin, D5_Pin, D6_Pin, D7_Pin};

Lcd_HandleTypeDef lcd;

lcd = Lcd_create(ports, pins, RS_GPIO_Port, RS_Pin, EN_GPIO_Port, EN_Pin,

LCD_4_BIT_MODE);

Lcd_string(&lcd, "4ilo - 4bit");

Lcd_cursor(&lcd, 1,6);

Lcd_int(&lcd, -500);

2.2. Work order:

1. In CubeMX, create a new project and conFig. all the pins used for the indicator to the output, also assign a name (User label) to pins D0..D7,EN,RS. (RW- we feed it to the ground, because there is no

We will read from the display, only write to it). D4..D7 will be used for 4-bit gears, the remaining D0..D3 - just in case - can be removed.

2. Generate a project, open it in Atollic TrueStudio.

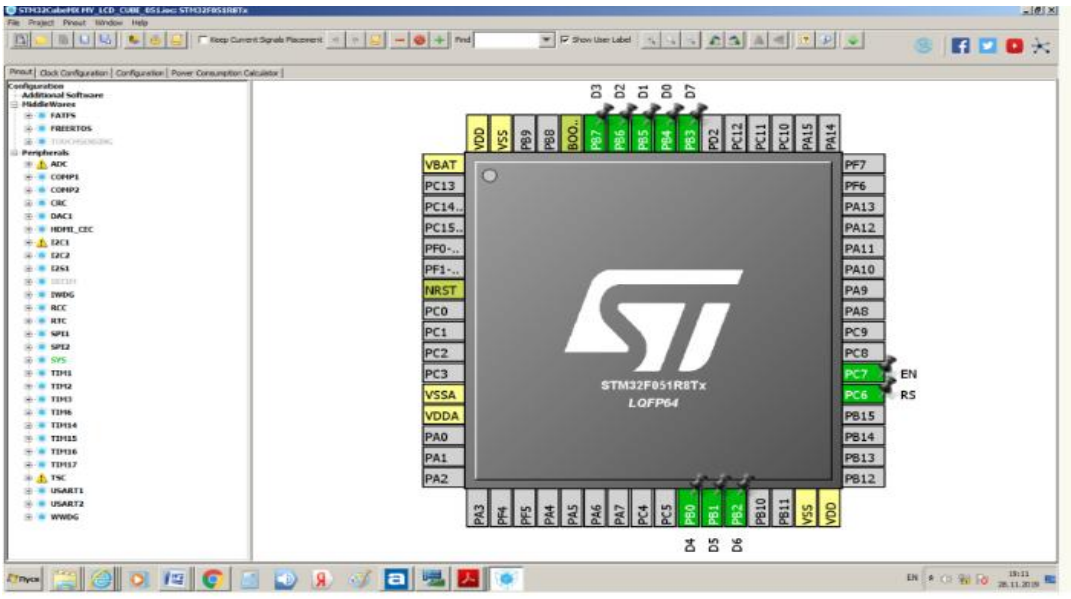


Fig. 4.9: Creating a project in CubeMX

Defines of the pins we named for connecting the display will appear in the main.h file:

```
#define D4_Pin GPIO_PIN_0
#define D4_GPIO_Port GPIOB
#define D5_Pin GPIO_PIN_1
#define D5_GPIO_Port GPIOB
#define D6_Pin GPIO_PIN_2
#define D6_GPIO_Port GPIOB
#define RS_Pin GPIO_PIN_6
#define RS_GPIO_Port GPIOC
#define EN_Pin GPIO_PIN_7
#define EN_GPIO_Port GPIOC
#define D7_Pin GPIO_PIN_3
#define D7_GPIO_Port GPIOB
#define D0_Pin GPIO_PIN_4
#define D0_GPIO_Port GPIOB
#define D1_Pin GPIO_PIN_5
#define D1_GPIO_Port GPIOB
```

```
#define D2_Pin GPIO_PIN_6
#define D2_GPIO_Port GPIOB
#define D3_Pin GPIO_PIN_7
```

3. In the main program main.c you need to connect the library:

```
#include "lcd.h"
```

Copy the lcd.c file to the src folder, lcd.h to the inc folder. Build the project for 4-bit output and

Display your last name on the top line of the indicator.

In the included file lcd.c all variables and functions of the library are declared, lcd structure:

```
#include "lcd.h"
const uint8_t ROW_16[] = {0x00, 0x40, 0x10, 0x50};
const uint8_t ROW_20[] = {0x00, 0x40, 0x14, 0x54};
/*************************************** Static declarations
***************************************/
static void lcd_write_data(Lcd_HandleTypeDef * lcd, uint8_t data);
static void lcd_write_command(Lcd_HandleTypeDef * lcd, uint8_t command);
static void lcd_write(Lcd_HandleTypeDef * lcd, uint8_t data, uint8_t len);
/*************************************** Function definitions
***************************************/
/**
* Create new Lcd_HandleTypeDef and initialize the Lcd
*/
Lcd_HandleTypeDef Lcd_create(
Lcd_PortType port[], Lcd_PinType pin[],
Lcd_PortType rs_port, Lcd_PinType rs_pin,
Lcd_PortType en_port, Lcd_PinType en_pin, Lcd_ModeTypeDef mode)
{
```

```
Lcd_HandleTypeDef lcd;
lcd.mode = mode;
lcd.en_pin = en_pin;
lcd.en_port = en_port;
lcd.rs_pin = rs_pin;
lcd.rs_port = rs_port;
lcd.data_pin = pin;
lcd.data_port = port;
Lcd_init(&lcd);
return lcd;

}

/**
* Initialize 16x2-lcd without cursor
*/
void Lcd_init(Lcd_HandleTypeDef * lcd)
{
if(lcd->mode == LCD_4_BIT_MODE)
{
lcd_write_command(lcd, 0x33);
lcd_write_command(lcd, 0x32);
lcd_write_command(lcd, FUNCTION_SET | OPT_N); // 4-bit mode
}
else
lcd_write_command(lcd, FUNCTION_SET | OPT_DL | OPT_N);
lcd_write_command(lcd, CLEAR_DISPLAY); // Clear screen
lcd_write_command(lcd, DISPLAY_ON_OFF_CONTROL | OPT_D);// Lcd-
on, cursor-off, no-blink
```

```
lcd_write_command(lcd, ENTRY_MODE_SET | OPT_INC); // Increment cursor
}
/**
* Write a number on the current position
*/
void Lcd_int(Lcd_HandleTypeDef * lcd, int number)
{
char buffer[11];
sprintf(buffer, "%d", number);
Lcd_string(lcd, buffer);
}
/**
* Write a string on the current position
*/
void Lcd_string(Lcd_HandleTypeDef * lcd, char * string)
{
for(uint8_t i = 0; i < strlen(string); i++)
{
lcd_write_data(lcd, string[i]);
}
}
/**
* Set the cursor position
*/
void Lcd_cursor(Lcd_HandleTypeDef * lcd, uint8_t row, uint8_t col)
{
#ifdef LCD20xN
lcd_write_command(lcd, SET_DDRAM_ADDR + ROW_20[row] + col);
```

```
#endif
#ifdef LCD16xN
lcd_write_command(lcd, SET_DDRAM_ADDR + ROW_16[row] + col);
#endif
}
/**
* Clear the screen
*/
void Lcd_clear(Lcd_HandleTypeDef * lcd) {
lcd_write_command(lcd, CLEAR_DISPLAY);
}
/************************************** Static function definition
**************************************/
/**
* Write a byte to the command register
*/
void lcd_write_command(Lcd_HandleTypeDef * lcd, uint8_t command)
{
HAL_GPIO_WritePin(lcd->rs_port, lcd->rs_pin, LCD_COMMAND_REG);
// Write to
command register
if(lcd->mode == LCD_4_BIT_MODE)
{
lcd_write(lcd, (command >> 4), LCD_NIB);
lcd_write(lcd, command & 0x0F, LCD_NIB);
}
else
{
lcd_write(lcd, command, LCD_BYTE);
```

```
}
}
/**
* Write a byte to the data register
*/
void lcd_write_data(Lcd_HandleTypeDef * lcd, uint8_t data)
{
HAL_GPIO_WritePin(lcd->rs_port, lcd->rs_pin, LCD_DATA_REG); //
Write
to data register
if(lcd->mode == LCD_4_BIT_MODE)
{
lcd_write(lcd, data >> 4, LCD_NIB);
lcd_write(lcd, data & 0x0F, LCD_NIB);
}
else
{
lcd_write(lcd, data, LCD_BYTE);
}
}
/**
* Set len bits on the bus and toggle the enable line
*/
void lcd_write(Lcd_HandleTypeDef * lcd, uint8_t data, uint8_t len)
{
for(uint8_t i = 0; i < len; i++)
{
58
```

HAL_GPIO_WritePin(lcd->data_port[i], lcd->data_pin[i], (data >> i) & 0x01);

}

HAL_GPIO_WritePin(lcd->en_port, lcd->en_pin, 1);

HAL_Delay(3);

HAL_GPIO_WritePin(lcd->en_port, lcd->en_pin, 0); // Data receive on falling edge

}

2.3. Control questions

1. Information display tools for microcontroller systems - LED, LCD, seven-segment, alphanumeric, graphic displays - differences, advantages and disadvantages, justify in which cases to use one or another option

2. Connection methods (8-bit, 4-bit, via I2C port expander), pinout WH1602 pins – the purpose of each pin.

3. Principles of working with displays based on HD44780 - what commands, in what order, how served. In what form and how is the data presented? Why is DDRAM needed in a display?

4. Standard sequence of operations for issuing at a specific cursor position strings of characters.

5. Use of ready-made libraries. Using the example of the lcd library. What are these files lcd.h, lcd.c

- what is in the files, where you need to put them to connect and how to connect with using the #include directive. What are and where are definitions, function declarations, Function prototypes? How can you assign the desired names to the port pins according to library - how to do it manually and how to use CubeMX. What is it intended for? main.h file?

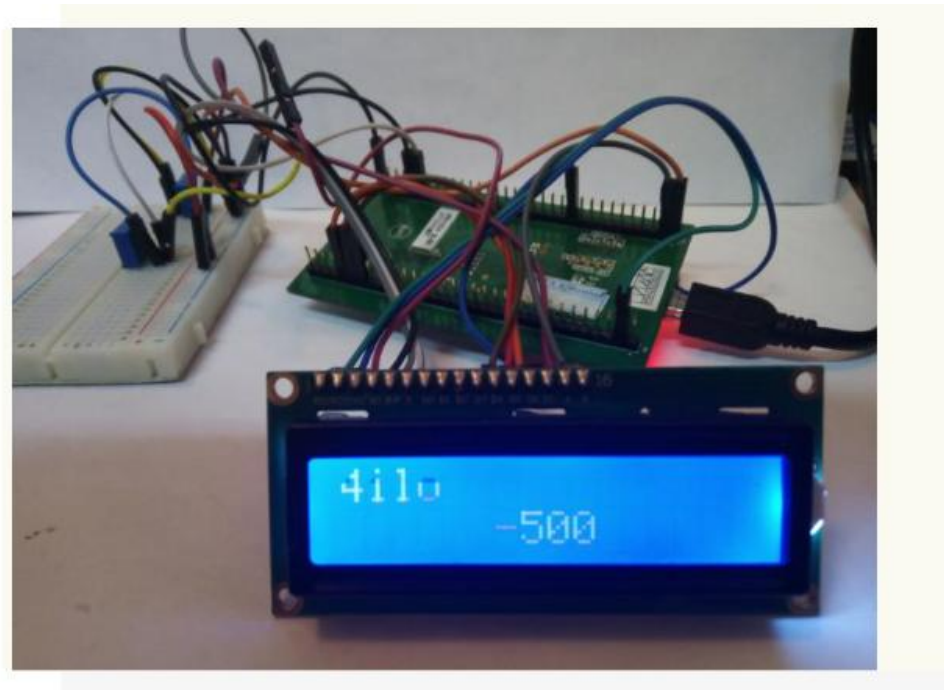


Fig. 4.10: Ready for use project

References:

1. Cortex-M3 Technical reference manual - ARM Ltd
2. ARMv7-M architectural reference manual - ARM Ltd
3. ARM Architectural reference manual Thumb2 supplement - ARM Ltd
4. STM32F103xx User Manual - ST Microelectronics
5. STM32F10xxx FLASH Programming manual - ST Microelectronics

SECTION 5

ENVIRONMENTAL PROTECTION

Introduction

The Life Cycle Assessment (LCA) method is an important instrument to evaluate the total environmental effect of a product or service from its conception to its final usage and disposal. This method enables businesses and customers to make informed decisions that consider the environment at all stages of the product's life cycle.

A product's life cycle is viewed as a complicated system that includes resource creation, transportation, usage, and recovery. We can evaluate the overall environmental impact by collecting and evaluating data at each stage. This strategy broadens the focus beyond traditional production to all stages.

The first stage of life cycle assessment looks at the resources, energy, and materials used to make a product. This comprises component manufacturing, transportation, and assembly. The research then shifts to the use phase, evaluating energy consumption and emissions during product operation. Finally, the aspects of recovery and disposal are considered, including waste treatment and material recycling.

One of the most significant advantages of life cycle analysis is the capacity to uncover green alternatives and strategies to optimize processes to decrease negative environmental effects. Companies that use this method can enhance their manufacturing processes, create environmentally friendly products, and communicate with consumers who are growing more conscious of and demanding sustainable development.

Engaging customers and raising their awareness are also key aspects of life cycle evaluation. People who understand the environmental impact of items can make more educated selections and select more sustainable options.

With increased emphasis being paid to environmental challenges, life cycle assessment is becoming a crucial tool for societal progress and the preservation of natural resources for future generations. Life cycle assessment is a step toward developing efficient and ecologically friendly production and consumption systems that contribute to the harmonic reconciliation of people's and nature's interests.

1. STM32 Materials

When selecting a product, the life cycle and environmental impact are frequently ignored even in cases where the functionality is similar. This project aims to evaluate the STM32's life cycle, emphasizing waste, energy, and materials.

A credit card-sized microprocessor called the STM32 addresses the demand for computer literacy. It may function as a desktop computer or boost the performance of smart gadgets.

The STM32 is made up of several complicated components, including raw materials and manufacturing methods. A complicated manufacturing method is used to create the PCB, comprised of fiberglass and copper. The fiberglass is made from sand, which has an impact on biodiversity. The extraction of silicon from sand and the intricate manufacture of semiconductors is required for the processor, which is essential to every electronic device.

Silver, palladium, or solid gold are examples of precious metals that are used in the device's electrical ports. The life cycle necessitates the use of fossil fuels for transportation in order to complete the processes of raw material acquisition, manufacture, distribution, consumption, and recycling.

Depending on how users modify the STM32, there are several phases of use. Updating the software is part of maintenance; hardware is typically left alone.

Electronic equipment recycling and disposal produce a large amount of garbage. Larger devices are more suited for recovering precious metals; however, the STM32's modest size restricts its use.

As a result, the whole life cycle of the STM32, from production to recycling, requires a complicated combination of resources, energy, and trash. The purpose of this analysis is to promote environmentally friendly products and increase customer awareness. To reduce energy use and leave a cleaner environmental imprint, solutions should give priority to the life cycle of the product and its effects on the environment.

1.1 The Environmental Impact Throughout the Life Cycle of STM32

With powers much beyond its small size, the STM32 is a credit card-sized computer that stands as a tribute to technological brilliance in the era of digital innovation. Its small form may be appealing, but its complicated life cycle necessitates a thorough assessment of its environmental effect. Every phase of the production process, including raw material extraction, shipping, usage, disposal, and recycling at the end of the device's life, contributes significantly to the minicomputer's total environmental impact.

Purchasing raw materials: Determining expenses

The extraction of raw materials is the first step in making an STM32. Mining operations for silicon and copper produce important components, but they also leave waste rock and tailings in their wake. Tailings ponds, which need to be carefully designed to avoid environmental problems, are where these leftovers eventually find resting places. In the meanwhile, natural gas and oil are needed for the manufacturing of plastic, a crucial part of the STM32. The extraction process produces drilling waste that is high in chemicals, salt, heavy metals, and radioactive elements. This waste needs to be disposed of carefully, frequently with the use of injection wells.

1.2 Manufacturing: an ensemble of elements

Plastic is made by a chemical cycle occurring when raw components are mixed together during the manufacturing process. Crude oil or natural gas components are transformed into hydrocarbon monomers by high-pressure processing, which is subsequently utilized to create polymers. Nevertheless, hazardous substances like trichloroethane, acetone, methylene chloride, styrene, sulfur oxides, nitrous oxides, methanol, ethylene oxide, and volatile organic compounds are released into the atmosphere during this chemical choreography. The manufacturing process of silicon wafers, an essential part of printed circuit boards, involves the use of potentially harmful materials like arsenic. The environmental expenses associated with wafer fabrication are substantial.

1.3 Production and assembly: The electric shock

So much power is used during the assembling process, and there are environmental consequences. The excessive dependence on fossil fuels for the production of electricity results in a rise in the atmospheric emissions of greenhouse gases, nitrogen oxides, and carbon monoxide.

1.4 Transportation: Getting about on roads and in the carbon skies

Due to the STM32's widespread appeal, extensive plane and train travel is necessary. A variety of substances, such as carbon dioxide, water vapor, hydrocarbons, carbon monoxide, nitrogen oxides, sulfur oxides, and soot, are released by airplanes powered by hydrocarbons. In a similar vein, rail travel increases greenhouse gas emissions. There are growing requests for better efficiency and a switch to renewable energy sources as the environmental effects of transportation become more apparent.

Utilizing the product: the issue with energy

Once in the hands of customers, the STM32 requires a minimum of 0.52 watts of electricity to function like a mini-computer. Its extensive reliance on power, however, connects its use to the more significant environmental problem of energy production. Greenhouse gases, nitrogen oxides, and carbon monoxide are released by fossil fuel-fueled power plants, underscoring the necessity of moving to renewable energy sources.

1.5 End-of-Life: From retirement to revival

The adventure of the STM32 doesn't stop when its functionality becomes outdated. Retirement creates the possibility of recycling or disposal, which is problematic for the environment. Due to its toxic and iron-rich nature, copper—a crucial component of STM32 wires—presents waste disposal issues. In contrast to recycling new copper, recycling copper has the potential to provide a glimpse of sustainability with far reduced greenhouse gas emissions.

There are three types of plastic that are often found in the environment: epoxy, polyvinyl chloride (PVC), and pi-polymethylpentene (PPP). They all provide different recycling issues, which highlights the necessity of appropriate disposal procedures. Hydrometallurgical processing is one of the sophisticated recycling processes needed for e-waste that contains silicon and other hazardous materials.

In Conclusion:

The STM32's life cycle study paints a complicated picture of its environmental effect. Every step has a weight of its own, from the mines to the assembly line, from transportation to disposal. But there is still hope in this narrative. The path toward a more sustainable future is being steered by initiatives that support recycling technology, increase consumer awareness, and support renewable energy. Because of its small size, the STM32 challenges us to consider not just what it can do computationally but also the environmental impact that every bit and circuit has.

2. STM32's Environmental Impact Compared to AVR's

Microprocessor development is not an exception to the rule that environmental effects are an inevitable byproduct of modern technological advancement. This essay will examine the effects of the STM32 and AVR microprocessors on the environment. Despite the fact that electronics uses both of these gadgets extensively, their effects on the environment might be very different.

The device's energy efficiency is one of the most important factors affecting the environment. For example, Atmel microcontrollers employ AVR microprocessors, which are renowned for having very low power consumption. For embedded systems, where every watt might matter, they are made.

On the other hand, since the STM32 is a multifunctional, full-fledged computer, it can need more power. The utilization of STM32 in various applications may result in increased power usage, which may have an impact on the environment due to greenhouse gas emissions during the electricity generation process.

The gadgets' capacity for scaling and recycling is another crucial feature. Usually found in low-resource, basic electronics, AVR microprocessors are smaller and simpler to recycle. Their straightforward designs also facilitate the separation of elements for recycling.

On the other hand, because of its intricate design and numerous parts, recycling the STM32 may be more challenging. It may be challenging to separate and reuse recovered material because of its size and multi-layered structure, which might lead to waste and have an adverse effect on natural resources.

Other significant elements that affect how environmentally friendly microprocessors are include availability and cost. Because of its extensive use in embedded systems and simplicity, AVR devices may be more reasonably priced.

However, the STM32 has a lot of characteristics that, in certain cases, make its price justified. However, the huge appeal of STM32s can also result in more waste and output, which has an impact on the environment.

The particular application and project context determine how AVR and STM32 microprocessors affect the environment. AVRs could be a preferable option if simple recycling and low power consumption are top concerns.

In order to strike a balance between technological advancement and environmental preservation, it is ultimately critical to take environmental factors into account while developing and utilizing electronics.

3. How to Minimize the Environmental Impact of STM32

The STM32 has transformed the computer landscape and become a powerful and adaptable tool in an era of fast technological growth. But even as we make use of its powers, we must consider the environmental effects of its life cycle. So how can we mix technical innovation with environmental responsibility while reducing the environmental effect of the STM32?

3.1. Sustainable raw material supply: Starting at the source is the first step towards sustainability. The detrimental effects of mining elements like copper and silicon can be lessened by forming partnerships with suppliers who use ecologically friendly mining techniques. Manufacturers of STM32s should prioritize their suppliers that follow ethical mining standards, reduce waste creation, and make sure that byproducts are disposed of responsibly by promoting openness in the supply chain.

3.2. Eco-friendly production techniques: A study of the manufacturing procedures is necessary to shape the future of STM32. Toxic chemicals and emissions related to the manufacture of silicon wafers and plastics can be greatly decreased by implementing green chemistry concepts and investing in cleaner production techniques. Enhancing manufacturing facilities' integration of sustainable technology, such as renewable energy, helps better synchronize production processes with environmental management.

3.3. Energy efficiency and the utilization of renewable energy sources: The main factor affecting the environment is the amount of energy used in the manufacture and consumption of goods. Energy-efficient procedures must to be given top priority by manufacturers in their manufacturing sites. Additionally, the carbon footprint connected to the STM32's life cycle may be greatly decreased by switching to sustainable energy sources like solar or wind power. A computer industry that is more environmentally conscientious can be facilitated by a commitment to sustainable energy techniques.

3.4. Conscientious shipping methods: Given the STM32's worldwide dissemination, careful planning must go into transportation. Manufacturers have to investigate low-carbon transportation choices and offer priority to shipping companies that follow eco-friendly policies. The carbon footprint connected with shipping may be further decreased by looking at local manufacturing options and using technology to improve shipping routes.5. User education and energy conservation: Encouraging users to understand how their gadgets affect the environment makes them feel more accountable. Manufacturers ought to educate consumers on energy-saving techniques and encourage the use of renewable energy sources. More sustainable usage of computers may be achieved by encouraging users to switch off the STM32 when not in use and by supporting energy-efficient code.

3.5. Disassembly and recycling-friendly design: The STM32's own design plays a significant role in reducing its environmental effect. The management of the equipment at the end of its life cycle can be enhanced by putting design concepts into practice that make disassembly and recycling easier. Using recyclable materials, cutting down on complicated parts, and properly labeling things for disposal are some examples of how to do this. It is recommended that manufacturers collaborate with e-waste recycling initiatives to guarantee the appropriate disposal and retrieval of precious materials.

3.6. Increase product longevity and upgradeability: You can drastically cut down on e-waste by encouraging consumers to get more usage out of their STM32 devices by giving them software updates and upgrade choices. Durability should be considered in product design so that consumers may replace worn-out parts without having to discard the complete gadget. This supports the growth of a circular economy in addition to being consistent with sustainable consumption habits.

3.7. Recycling efforts and take-back programs: To handle electronics ethically at the end of their life cycle, it is imperative to establish strong recycling initiatives and take-back programs. In order to give consumers reasonably priced options for recycling their old STM32 devices, manufacturers have to collaborate with e-waste recycling centers. Education initiatives may help people understand the value of recycling e-waste and the significance of correct disposal.

In conclusion, minimizing the environmental effect of STM32 calls for an all-encompassing strategy that includes end-of-life recycling, user education, energy efficiency, sustainable sourcing, and manufacturing techniques. The STM32 community can lead the way in a paradigm change toward a more ecologically

conscious and sustainable computing era by implementing these suggestions. Not only is technology evolving, but it is evolving with a conscience, working to make a good impact on the earth we live on.
It is clear from looking at the STM32's life cycle and the trash and emissions it produces that there have been environmental difficulties along the way for this amazing computing gadget. Every stage of the process, including raw material extraction, manufacture, transportation, usage, and disposal or recycling, has an impact on the environment. There is a method to accomplish sustainable growth and reduce environmental effects, despite all the challenges and worries.

The technology industry needs to adopt good practices because to the waste produced during the mining of copper and silicon, air pollutants during industrial operations, and the carbon footprint connected with transportation. An important and indisputable environmental component arises from the manufacture and usage of items that rely on fossil fuels for energy. These difficulties do, yet, also provide chances for creativity and constructive development.

Sustainable sourcing, eco-friendly production techniques, responsible energy usage, and promoting environmentally conscious user behavior are all important components of any effort to reduce waste and emissions. Because of its prolonged product life, flexibility to be upgraded, and capacity for recycling, the STM32's design has the potential to be a catalyst for change.

When the STM32 loses service, the adventure is not over. A circular economy is being ushered in by programs like take-back programs and collaboration with recycling facilities, which emphasize the responsible disposal or recycling of e-waste. Recycling resources like plastic, silicon, and copper offers a chance to lessen environmental effects and transition to a more sustainable strategy.

Conclusions

In summary, this chapter revealed that the life cycle of the STM32 is a story of technical innovation entwined with environmental responsibility. The call to action is obvious as we negotiate a complicated web of emissions and waste: an industry-wide dedication to energy efficiency and sustainable practices, as well as a shared duty for a future where innovation coexists peacefully with the environment. The STM32, a representation of accessibility and creativity, has the power to advance technology and establish a benchmark for environmental responsibility in the computer sector.

SECTION 6
OCCUPATIONAL SAFETY AND HEALTH

In the ever-evolving landscape of technology, engineers play a pivotal role in crafting innovative solutions that drive progress and efficiency. The advent of automation, particularly in equipment development utilizing advanced platforms such as the STM32 microcontroller, has significantly transformed the engineering landscape. As engineers delve into the intricacies of automated workplaces, the spotlight on occupational safety and health measures becomes increasingly crucial. The integration of STM32 microcontrollers in equipment development symbolizes the seamless convergence of software and hardware, promising heightened precision, speed, and functionality. While these advancements bring about a myriad of benefits, they also introduce new challenges, especially concerning the well-being of the engineers tasked with harnessing the potential of such cutting-edge technologies. The working environment for engineers engaged in the development of equipment based on STM32 microcontrollers presents a unique set of challenges and potential hazards. Recognizing and mitigating these harmful factors is paramount to ensure the health and safety of the workforce.
In the dynamic realm of technology, engineers at the forefront of innovation face an array of challenges and hazards while developing equipment based on STM32 microcontrollers. This essay delves into the multifaceted factors that can compromise the well-being of engineers in this automated workplace, outlining standards and legislative requirements crucial for safeguarding their health.

6.1. Harmful and hazardous working factors

One significant risk stems from electrical hazards, wherein engineers may encounter the potential for shocks or burns. Compliance with international standards, such as IEC 60950, is imperative, coupled with comprehensive training in electrical safety practices. Ergonomic challenges, manifested through musculoskeletal disorders and eye strain, necessitate adherence to standards like ISO 9241, emphasizing the need for ergonomic workstations and regular breaks.

Chemical exposure during manufacturing or testing poses health risks, demanding compliance with regulations like the Globally Harmonized System (GHS) and the provision of appropriate personal protective equipment (PPE). Noise pollution, an inherent part of equipment development, requires adherence to occupational noise exposure standards and the provision of hearing protection.

The potential for working in confined spaces mandates compliance with standards like OSHA's 29 CFR 1910.146, ensuring proper training and safety protocols. Radiation exposure, especially in certain equipment development processes, requires adherence to standards set by the International Commission on Radiological Protection (ICRP) and adequate protective measures.

Stress and mental health issues, stemming from the fast-paced nature of development, necessitate the implementation of occupational stress management programs and adherence to mental health support standards. Machine and equipment hazards call for compliance with safety standards like ISO 12100, coupled with thorough training and safety measures.

Biological hazards, depending on the equipment's nature, require adherence to biological safety standards and the provision of protective measures. Fire and explosion risks demand compliance with fire safety standards like NFPA 70 and the implementation of adequate prevention and emergency response measures.

In conclusion, mitigating these harmful factors requires a holistic approach. Employers must adhere to relevant standards and legislation, providing comprehensive training, protective measures, and a supportive work environment. Continuous monitoring and improvement of safety protocols are essential to foster a culture of well-being in the workplace, ensuring that engineers can contribute to technological progress in a safe and sustainable manner.

Electrical Hazards:

Description:

Engineers working with equipment based on STM32 microcontrollers face potential electrical hazards, including the risk of shocks and burns. This risk arises from direct contact with electrical components or circuits during the development, testing, and maintenance phases.

Possible Harm:

Exposure to electrical hazards can result in injuries ranging from mild shocks to severe burns, and in extreme cases, it can lead to fatalities. The harm depends on factors such as the magnitude and duration of the electric shock, as well as the pathway it takes through the body.

Comparison with Normative Recommendations:

Normative recommendations, such as those outlined in IEC 60950, establish safety requirements for information technology equipment. Compliance involves ensuring that the equipment meets insulation and clearance distances, grounding requirements, and protective measures to minimize electrical hazards. Regular assessments of the electrical systems against these standards are essential to maintain a safe working environment.

Protective Measures:

Training: Ensure that engineers undergo comprehensive training in electrical safety, emphasizing proper handling of equipment and the use of personal protective equipment (PPE).

Isolation and Lockout/Tagout Procedures: Implement strict isolation procedures when working on live circuits, including the use of lockout/tagout systems to prevent accidental energization.

PPE: Provide appropriate PPE, such as insulated gloves, safety glasses, and flame-resistant clothing, to mitigate the risk of electrical injuries.

Regular Equipment Inspections: Conduct routine inspections of electrical equipment to identify and address potential hazards promptly.

6.2. Analysis of working conditions and development of protective measures

6.2.1 Electrical Hazards:

Engineers working with equipment based on STM32 microcontrollers face potential electrical hazards, including the risk of shocks and burns. This risk arises from direct contact with electrical components or circuits during the development, testing, and maintenance phases.

Exposure to electrical hazards can result in injuries ranging from mild shocks to severe burns, and in extreme cases, it can lead to fatalities. The harm depends on factors such as the magnitude and duration of the electric shock, as well as the pathway it takes through the body.

Normative recommendations, such as those outlined in IEC 60950, establish safety requirements for information technology equipment. Compliance involves ensuring that the equipment meets insulation and clearance distances, grounding requirements, and protective measures to minimize electrical hazards. Regular assessments of the electrical systems against these standards are essential to maintain a safe working environment.

Protective Measures:

Training: Ensure that engineers undergo comprehensive training in electrical safety, emphasizing proper handling of equipment and the use of personal protective equipment (PPE).

Isolation and Lockout/Tagout Procedures: Implement strict isolation procedures when working on live circuits, including the use of lockout/tagout systems to prevent accidental energization.

PPE: Provide appropriate PPE, such as insulated gloves, safety glasses, and flame-resistant clothing, to mitigate the risk of electrical injuries.

Regular Equipment Inspections: Conduct routine inspections of electrical equipment to identify and address potential hazards promptly.

6.2.2 Chemical Exposure:

Engineers in STM32 microcontroller-based equipment development may encounter hazardous chemicals during manufacturing, testing, or maintenance activities. Chemical exposure can occur through direct skin contact, inhalation of fumes, or accidental ingestion.

Chemical exposure can lead to various health issues, ranging from skin irritation and respiratory problems to more severe conditions, depending on the toxicity and concentration of the chemicals. Long-term exposure may result in chronic health problems.

Compliance with the Globally Harmonized System (GHS) and other relevant regulations is crucial. This includes proper labeling of chemicals, providing Material Safety Data Sheets (MSDS), and ensuring that exposure levels are within permissible limits set by regulatory bodies.

Here are some protective measures that can prevent the worker from such exposures:

Ventilation Systems: Install effective ventilation systems in work areas to control and reduce airborne concentrations of hazardous chemicals.

Personal Protective Equipment (PPE): Provide appropriate PPE, such as gloves, safety goggles, respirators, and coveralls, based on the nature of the chemicals being used.

Training and Awareness: Conduct regular training sessions to educate engineers about the potential hazards associated with specific chemicals, proper handling procedures, and emergency response protocols.

Substitution of Hazardous Substances: Explore opportunities to replace hazardous chemicals with less toxic alternatives to minimize overall risk.

Regular Monitoring: Implement regular monitoring of air quality to ensure that chemical concentrations are within permissible exposure limits, and take corrective actions if necessary.

By addressing electrical hazards and chemical exposure through comprehensive protective measures, employers can create a safer working environment for engineers, reducing the risk of injuries and long-term health issues. Ongoing monitoring and adherence to established standards are essential components of a proactive safety approach.

6.3. Fire Safety Rules at the workspace

Fire safety is a critical consideration in the workplace, demanding strict adherence to established rules and practices. A comprehensive approach to fire safety is crucial for protecting employees, property, and maintaining business continuity.

One fundamental aspect is the development and communication of a clear emergency evacuation plan. This plan should outline evacuation routes, assembly points, and the specific procedures to be followed during a fire emergency. Ensuring that all exits are clearly marked, unobstructed, and easily accessible is essential. The installation and regular maintenance of fire alarm systems and smoke detectors contribute to early fire detection. Similarly, providing strategically placed fire extinguishers and conducting employee training on their effective use enhances the overall fire response capability. Enforcing strict no-smoking policies within the workspace, particularly in areas with flammable materials, is a basic yet critical rule. Electrical safety measures, including regular inspections of electrical systems and equipment, help prevent electrical fires. Additionally, adhering to safety guidelines for the use of electrical appliances is paramount. Proper storage of flammable materials in designated areas with adequate ventilation, away from potential ignition sources, is imperative. This practice aligns with safety data sheets (SDS) guidelines for storage and handling. Employee training is a cornerstone of fire safety, covering evacuation procedures, proper use of firefighting equipment, and awareness of potential fire hazards. A well-equipped first aid kit and designated first aid responders contribute to a more comprehensive emergency response. Implementing a hot work permit system for activities involving open flames or welding helps ensure these tasks are conducted safely and under supervision. Regular fire drills serve to familiarize employees with evacuation procedures, identify weaknesses in the emergency response plan, and reinforce a culture of preparedness. Maintaining clear workspaces by eliminating clutter and properly storing combustible materials minimizes fire risks. Installing clear and visible fire safety signage, including exit signs, fire extinguisher locations, and emergency contact information, aids in effective communication during emergencies.Regular inspections of fire safety equipment, systems, and infrastructure, coupled with prompt resolution of identified issues, are vital for sustained fire safety. This proactive approach contributes to a culture of safety within the workplace, promoting a collective commitment to minimizing fire hazards and ensuring a secure working environment.

Conclusions

In conclusion, the diploma work has made noteworthy contributions to both scientific knowledge and practical applications in the domain of cyber security, specifically focusing on embedded systems utilizing the STM32 microcontroller. The key achievements can be summarized as follows:

The primary scientific innovation lies in the successful development of a cyber-protected workplace based on the STM32 microcontroller. This represents a significant advancement, addressing a previously understudied area in the realm of cyber security.

The comprehensive analysis of threats and vulnerabilities for embedded systems is another substantial contribution, offering a foundational understanding of potential attacks and guiding the development of effective protection measures.

The implementation of security mechanisms, including robust authentication and authorization systems, enhances the overall security posture of embedded systems, providing reliable protection against cyber threats.

In terms of practical value, the developed cyber-protected workplace finds application across diverse industries, such as automotive, medical, and industrial automation. This tool serves to safeguard embedded systems within these sectors.

Furthermore, the secure mechanisms devised in this study have broader implications for improving the security of Internet of Things (IoT) devices. In light of the increasing prevalence of connected devices, the developed mechanisms offer a relevant and effective solution to enhance IoT security.

The findings of this research hold practical significance for organizations and enterprises seeking to fortify the protection of their information systems and embedded devices. The implemented security measures contribute to safeguarding critical assets and data.

Moreover, the work lays the groundwork for future research endeavors in the field of cyber protection for embedded systems and microcontrollers. It provides a valuable platform for researchers to delve deeper into the complexities of securing these systems.

In summary, the diploma work not only advances our understanding of cyber security in embedded systems but also provides tangible tools and insights with wide-ranging applications. The research contributes to ongoing efforts to secure information systems, protect embedded devices, and establishes a foundation for continued exploration in this vital and evolving field.

References

[1] Programming 32-bit Microcontrollers in C (Lucio Di Jacio)

[2] PROGRAMMING LANGUAGE (BRIAN W.KERNIGHAN & DENNIS M.RITCHIE)

[3] USB Complete (JAN AXELSON)

[4] Getting Started with LabVIEW (National Instrument_web site or LabVIEW help)digital.ni.com/manuals.nsf/websearch/07F9CD824F66237A86257046006F7C5D

[5] LabVIEW For Everyone (JEFFREY TRAVIS & JIM KRING)

[6] Introduction to LabVIEWTM Six-Hour Course (National Instrument_web site) zone.ni.com/devzone/cda/tut/p/id/5241

[7] USB Instrument Control Tutorial (National Instrument_web site) zone.ni.com/devzone/cda/tut/p/id/4478

[8] LabVIEW Graphical Programming Course (Connexions) cnx.org/content/col10241/latest/

[9] Firmware_lib_users_manual

[10] REF_Manual_STM32F103

[11] Cortex_Technical_Reference_Manual

[12] IAR Embedded Workbench IDE User Guide users.ece.gatech.edu/~mmckeown3/ECE3884/pdf/IAR%20Embedded%20Workbench%20IDE%20User%20Guide%20for%20ARM.pdf

[13] Datablad_STM32F103

[14] STM32F10xxx USB development kit www.st.com/stonline/products/literature/um/13465.htm

[15] Analogue-to-Digital Converters ww1.microchip.com/downloads/en/devicedoc/adc.pdf

[16] USB Documents on website www.beyondlogic.org/usbnutshell/usb1.htm (USB in a Nutshell)

www.usb.org/developers/usb20/ (Universal Serial Bus)