STM32

Mohammed H. IBRAHIM
Necmettin Erbakan University

Introduction

ARM has named by the abbreviation of (Acorn RISC Machine). ARM processors are 32-bit and 64-bit RISC (Reduced instruction set computer) based processors. ARM is preferred in various embedded applications due to its ability to reach high speeds, has 64-bit architecture, consume very little energy, and rich peripheral hardware. Moreover, ARM architecture processors are used in many electronic devices such as mobile phones, computers, and tablets. (See Figure 1 Areas use of ARM processors).

Figure 1. Areas use of ARM Processors

Many companies are used ARM processors to design ARM-based microcomputers using their own software, libraries, and names. Most of these designed microcomputers are used very widely in education and embedded systems projects. In general, the programming of these ARM-based microcomputers is based on the C language. However, since each company shapes the library features according to the microcomputer it produces, each brand has its own libraries. According to this information, although programming ARM is basically based on the C language, programming will also vary with each company, due to the changing libraries and characteristics of the designed microcomputer. For example; when you learn the ARM processor-based STM32 Discovery programming produced by STM, you cannot program ARM processors produced by TI using the same libraries and features.

As given in Figure 2, the entire ARM architecture is classified into 3 classes according to application areas: classic ARM processors, embedded cortex processors, and application cortex processors. Classic ARM processors contain ARM, embedded cortex processors contain ARM cortex-M and ARM cortex-R, and application cortex processors contain ARM cortex-A (Ngabonziza et al., 2016).

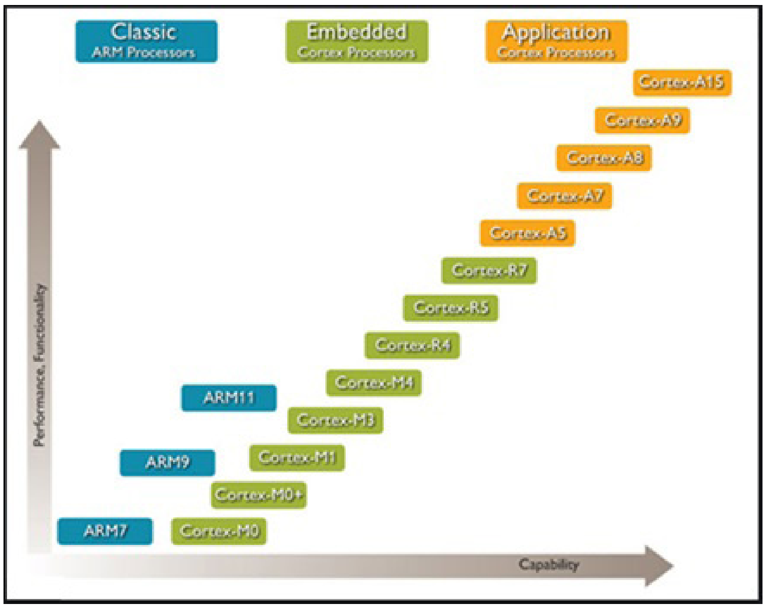


Figure 2. Classification of the ARM Processors

Cortex-A series: Cortex-A is a series developed for applications that require high speed due to its low power consumption and fast operation (Wan et al., 2012). Cortex-A5, Cortex-A7, Cortex-A8, Cortex-A9, and Cortex-A15 are types of ARM Cortex-A commonly used. In addition, the ARM Cortex-A series are commonly used in microcomputers such as Tablet, raspberry pi, Nvidia Jetson.

Cortex-R series: Cortex-R series are frequently used in real-time projects and technical devices. Cortex-R4, Cortex-R5, and Cortex-R7 are types of ARM Cortex-R commonly used. Cortex-R series are used in computer hardware devices (hard disk control, network connection equipment), automotive industry (airbag systems, brake and engine management), household electronic devices (food robots, washing machines, refrigerators, rugs), and enterprise printers. For example, Cortex-R4 is frequently used in the automotive industry (Iturbe et al., 2017).

Cortex-M series: Cortex-M series are a series developed to be used in the electronics industry projects where standard 8/16 bit microcontrollers are used, and projects that require a low cost / low power consumption. Moreover, due to its low cost, it is frequently preferred in educational control cards. Therefore, this article is described the STM32 control board with Cortex-M4 ARM processor, which is widely used today. Cortex-M0, Cortex-M1, Cortex-M3, and Cortex-M4 are types of ARM Cortex-M commonly used (Martin, 2016).

Classic ARM: Generally, classic ARM processors are used in many simpler and smaller-scale embedded systems. ARM7, ARM9, and ARM11 are families of ARM Cortex-A and ARM7 is still the highest shipping 32-bit processor (Fleisher & Bensoussan, 2015). Even though classic ARM processors can be used in many simpler and smaller-scale 32-bit devices, newer embedded systems are built using advanced ARM processors like Cortex-M processors and Cortex-R Processors.

ARM Processor Architecture

The ARM consists of a processor which is responsible for mathematical operations such as addition, subtraction, etc., as well as logical operations such as comparison. The ARM also contains a random access memory, which is called RAM, in which the program is stored temporarily while it is running and this memory is erased all its content from disconnect the power supply (Volatile), and it also contains an electronically erasable programmable read-only memory, which is called a EEPROM, as well as the general-purpose input/output and timers circuits. These aforementioned components interconnected with address, control, and data buses as illustrated in Figure 3.

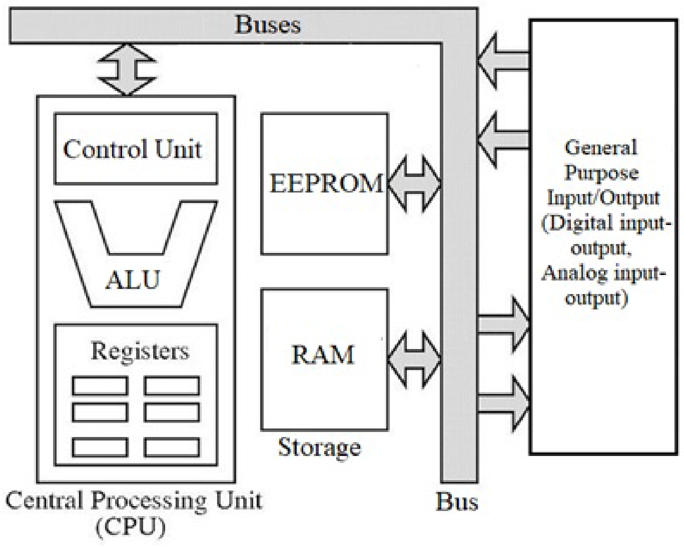

Buses
Control Unit
ALU
Registers
Central Processing Unit (CPU)
EEPROM
RAM
Storage
Bus
General Purpose Input/Output (Digital input-output, Analog input-output)

Figure 3. Components of ARM

The process of placing these components together and connecting them with each other will not lead to obtaining the required functions of the motherboard, which is represented in driving the computer as a whole, showing what is happening on the output and receiving various commands from the user through input terminals. There should be a certain methodology that organizes the flow of computer operations and all these tasks are fundamentally related to the concept of “architecture.”

Architecture is the mechanism by which different functional units are organized within the computer system. Thus, the architecture determines how data and information transfer between computer units such as the central processing unit, memory, and input and output ports. To evaluate any architecture there are three main criteria that must be considered:

- Performance
- Cost
- The maximum size of the program and data

Besides, there are many other criteria by which we can evaluate the architecture such as power consumption, size and weight (two criteria concerned with the physical structure of the processor), and ease of programming. But, the three criteria above are important in all applications.

All processors today depend on the Instruction Set Architecture (ISA), where the wizard implements one instruction from the program's instructions through several cycles, and the program's instructions are executed one by one. The instruction is the simplest function that the wizard can perform, and it consists of Opcode (represents the required function) and Operand, which is the recorder on which the required function is applied, for example:

ADD r1, r2, r3;

You add the value of r3 with r2 and put the result in r1, which is equivalent to r1 = r2 + r3. You can learn more information about the instruction set of ARM from (www.st.com:, 2019).

Von Neumann and Harvard Architectures

Von Neumann: The design philosophy in this type of architecture is based on storing Instructions and data in the same memory (Mariantoni et al., 2011). In this case, the processor would need to implement the instruction consisting of one opcode and one parameter into three cycles - a cycle for fetching and reading the opcode (by the address bus), a cycle for reading the parameter, and a final cycle for implementing the instruction content - which is a waste of time. Therefore, there is a mainline bus to transfer data and control commands to all computer components, where the processor fetches the program instructions stored in the random memory to the CPU to execute it as well as the data is fetched from the input means or from random memory or vice versa data is taken out for output devices or stored in memory (Ben-Sasson et al., 2014). ARM Cortex-M23 is an example of Von Neumann architecture (Yiu, 2020). (See Figure 4).

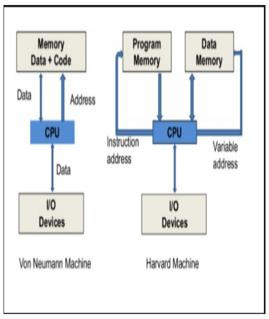


Figure 4. Von Neumann Architecture

Harvard: In this architecture, there are two separate memories, a memory for storing instructions called "Program Memory (PM") and another memory called "Data Memory (DM") for storing data. This is what needs an address bus and a special Data Bus for each memory CPU. This allows the processor to fetch information from program memory and access variables in the data memory at the same time. And since the instruction memory was separated from the data, the CPU can now write the opcode and parameter in one statement, as the CPU can read the entire instruction in only one session most of the time. This is the goal behind Harvard architecture, and this is what makes devices that rely on this architecture give much higher productivity than those based on von Neumann architecture. ARM Cortex-M4 is an example of Harvard architecture. (See Figure 5).

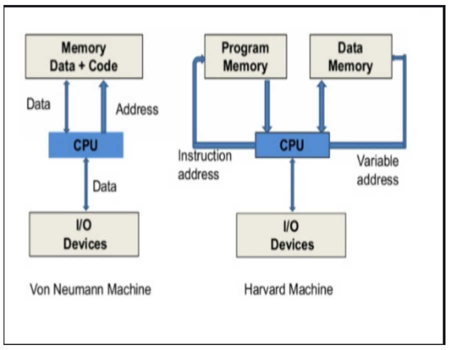


Figure 5. Harvard Architecture

Briefly, in microprocessors, instructions are stored in read-only memory (or fixed memory), that is, ROM, and variables are stored in random (or changeable) memories. RAM, so the memory in the von Neumann architecture is integrated between ROM, which saves the instructions and their operating codes, and RAM, to save the variables. The second reason ARM processors are fast is that they have RISC architecture. RISC architecture has a faster and lower cost than CISC architecture. On the other hand, CISC architectures have relatively more instruction sets than RISC architectures.

Cortex-M Family and STM32

Cortex-M family is used by many companies, especially companies that develop projects with embedded systems prefer this processor. As the usage areas of the Cortex-M series increased, companies such as ST (STMicroelectronics) and TI (Texas Instruments) started to manufacture ARM Cortex-M-based control boards with different properties. The application areas of ARM Cortex-M is given in Figure 6.

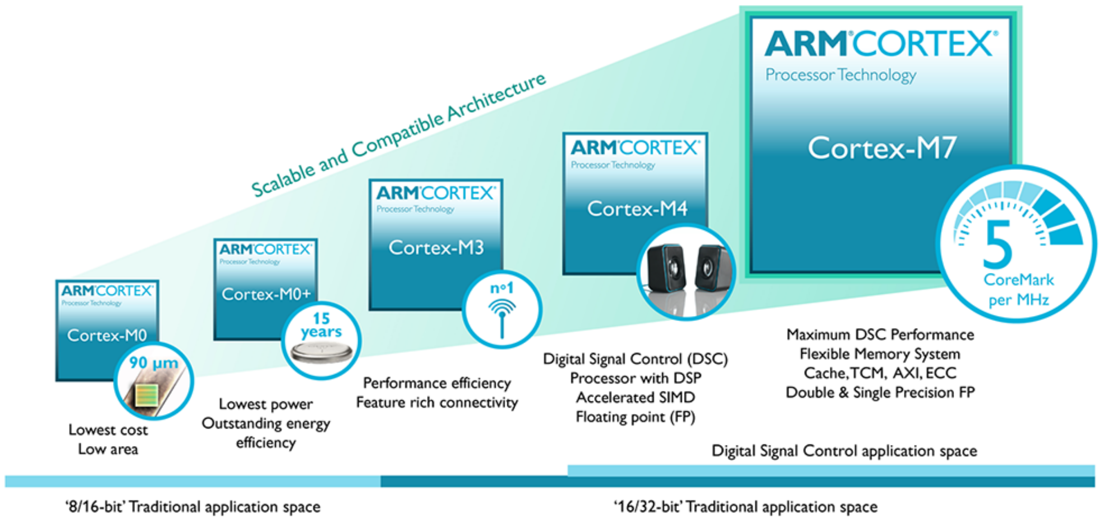

Scalable and Compatible Architecture
ARM CORTEX Processor Technology
Cortex-M0
90 µm
Lowest cost
Low area
Cortex-M0+
15 years
Lowest power
Outstanding energy efficiency
Cortex-M3
nº1
Performance efficiency
Feature rich connectivity
Cortex-M4
Digital Signal Control (DSC)
Processor with DSP
Accelerated SIMD
Floating point (FP)
Cortex-M7
5 CoreMark per MHz
Maximum DSC Performance
Flexible Memory System
Cache,TCM, AXI, ECC
Double & Single Precision FP
Digital Signal Control application space
'8/16-bit' Traditional application space
'16/32-bit' Traditional application space

Figure 6. Application Areas of ARM Cortex-M

STMicroelectronics was founded as a result of the merger of two companies, one French and the other Italian in 1987 and since then the company has managed to occupy an important place among the major electronics companies. For instance, STMicroelectronics came in fifth place in 2002 after Intel, Samsung, Texas Instruments, and Toshiba, and today STMicroelectronics is the largest European manufacturer of chips. In terms of electronic profits, the STMicroelectronics Company started in 2007 with the production of its first product STM32F1 from the family of STM32 controllers based on ARM processors. The STM32 family of controllers features a wonderful 32-bit Cortex-M processor allowing the user to use software tools from ARM and full support from many developments and programming environments. Moreover, the STM32 family also facilitates the process of moving from one controller to another within the STM32 family, so you do not have to worry if you program and develop on a microcontroller and want to move to another controller, you can do this very easily. The STM32 family of controllers is also faster and cheaper than some 8-bit controllers.

As seen in Figure 7, the STM32 family of controllers contains 11 groups divided into 3 classes. First in terms of high-performance STM32H7, STM32F7, STM32F4, and STM32F2 have high computing power and are supported by ART Accelerator technology that enables direct execution of instructions from flash memory without waiting and speeds up to 400MHz, Therefor, it is suitable for multimedia, graphics, and digital signal processing applications.

In terms of medium performance, it includes STM32F0, STM32F1, and STM32F3. This type of STM is characterized by its cheap price and small size, as it is convenient for many commercial applications and has speeds of up to 72MHz.

In terms of low energy consumption, it includes STM32L4, STM32L4 +, STM32L0, and STM32L1 and it is of course characterized by its low energy consumption, which makes it suitable for applications that depend on battery for operation. This type of STM is characterized by its support for Dynamic Voltage Scaling Technology - DVS, which allows improving the processor’s energy consumption based on the frequency used.

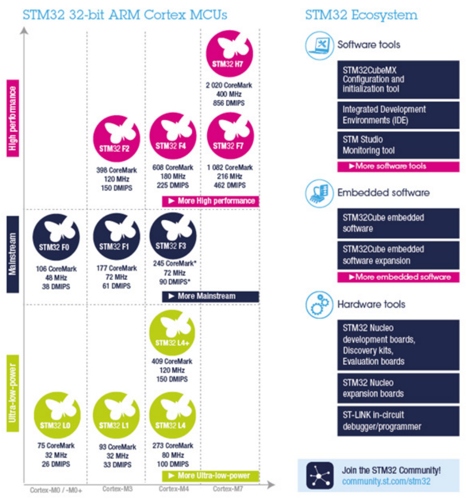


Figure 7. Class of the STM32 Cortex-M family

Technical properties of STM32

In this section, the STM32F4 discovery board, which is frequently used among STM32 boards, has examined in detail, the top layout of the STM32F4 board (www.st.com:, 2020) and the pinout of the STM32F4 board (Microcontrollerslab, 2021) are given in Figures 8 and 9, respectively.

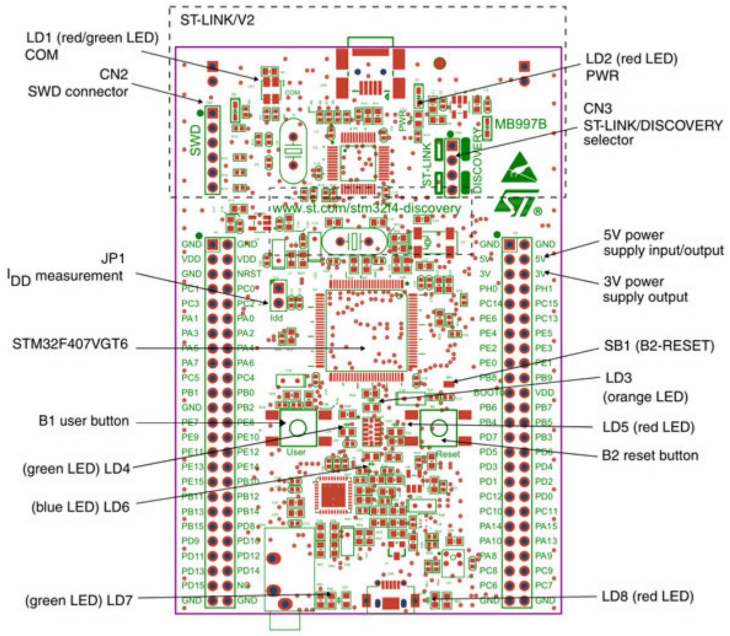


Figure 8. The top Layout of the STM32F4 Board

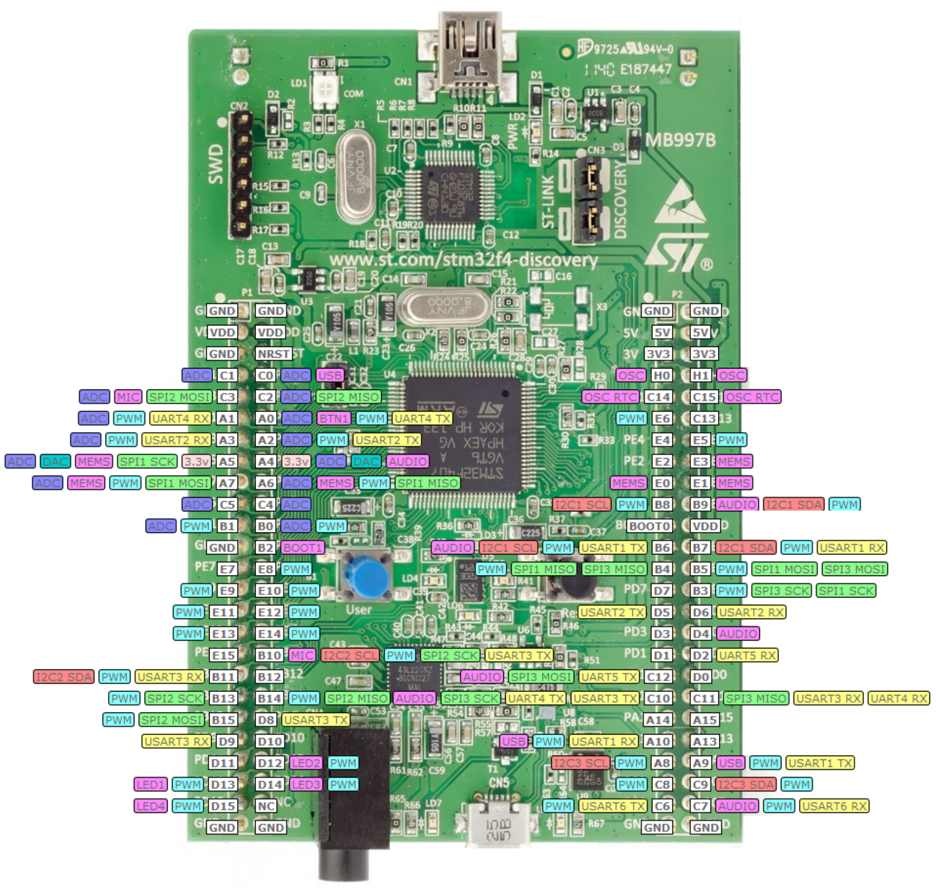

SWD
www.st.com/stm32f4-discovery
MB997B
ST-LINK
DISCOVERY
User
GND GND
VDD VDD
GND NRST
ADC C1 C0 ADC USB
ADC MIC SPI2 MOSI C3 C2 ADC SPI2 MISO
ADC PWM UART4 RX A1 A0 ADC BTN1 PWM UART4 TX
ADC PWM USART2 RX A3 A2 ADC PWM USART2 TX
ADC DAC MEMS SPI1 SCK 3.3v A5 A4 3.3v ADC DAC AUDIO
ADC MEMS PWM SPI1 MOSI A7 A6 ADC MEMS PWM SPI1 MISO
ADC C5 C4 ADC
ADC PWM B1 B0 ADC PWM
GND B2 BOOT1
E7 E8 PWM
PWM E9 E10 PWM
PWM E11 E12 PWM
PWM E13 E14 PWM
E15 B10 MIC I2C2 SCL PWM SPI2 SCK USART3 TX
I2C2 SDA PWM USART3 RX B11 B12
PWM SPI2 SCK B13 B14 PWM SPI2 MISO AUDIO SPI3 SCK UART4 TX
PWM SPI2 MOSI B15 D8 USART3 TX
USART3 RX D9 D10
D11 D12 LED2 PWM
LED1 PWM D13 D14 LED3 PWM
LED4 PWM D15 NC
GND GND
GND GND
5V 5V
3V3 3V3
OSC H0 H1 OSC
OSC RTC C14 C15 OSC RTC
PWM E6 C13
E4 E5 PWM
E2 E3 MEMS
MEMS E0 E1 MEMS
I2C1 SCL PWM B8 B9 AUDIO I2C1 SDA PWM
BOOT0 VDD
AUDIO I2C1 SCL PWM USART1 TX B6 B7 I2C1 SDA PWM USART1 RX
PWM SPI1 MISO SPI3 MISO B4 B5 PWM SPI1 MOSI SPI3 MOSI
D7 B3 PWM SPI3 SCK SPI1 SCK
USART2 TX D5 D6 USART2 RX
D3 D4 AUDIO
D1 D2 UART5 RX
AUDIO SPI3 MOSI UART5 TX C12 D0
USART3 TX C10 C11 SPI3 MISO USART3 RX UART4 RX
A14 A15
USB PWM USART1 RX A10 A13
I2C3 SCL PWM A8 A9 USB PWM USART1 TX
PWM C8 C9 I2C3 SDA PWM
PWM USART6 TX C6 C7 AUDIO PWM USART6 RX
GND GND

Figure 9. The Pinout of the STM32F4 Board

In addition, the technical properties of STM32F4 discovery which is commonly used has given bellow (www.st.com:, 2020).

Processor (Core): The STM32F4 discovery contains ARM Cortex-M4F core with a maximum clock rate of 84 / 100 / 168 / 180 MHz.

Oscillators: The oscillators in the STM32F4 discovery consists of internal (16 MHz, 32 kHz) and optional external (4 to 26 MHz, 32.768 to 1000 kHz).

Memory: The STM32F4 include flash memory and static RAM. The flash memory consists of up to 2 MB of dual bank and supporting read-while-write (RWW), 512 / 1024 / 2048 KB general-purpose, 512 bytes one-time programmable (OTP), and 30 KB system boot. Also contain dual bank option bytes for microcontroller configuration. The static RAM memory consists of up to 192 KB general-purpose and 64 KB core-coupled memory (CCM).

Peripherals:

Communication: The STM32F4 include Universal Serial Bus 2.0 On-The-Go (USB OTG), two Controller Area Network (CAN bus) 2.0B, one Serial Peripheral Interface (SPI) and two SPI or full-duplex Inter-IC Sound (I²S), three Inter-Integrated Circuit

(I²C), four universal synchronous and asynchronous receiver-transmitter (USART), two universal asynchronous receiver-transmitter (UART), Secure Digital Input Output (SDIO) for SD/MMC cards.

Timers: STM32F4 has a total of 17 timers with 16-bit and 32-bit including 10 general-purpose timers, two advanced control timers, two basic timers, one independent watchdog timer (IWDT), one window watchdog timer (WWDT), and one systematic timer.

General purpose timers: The general purpose timers include 16-bit or 32-bit timers up, down, up/down auto-reload counter. TIM3, TIM4, TIM9, TIM10, TIM11, TIM12, TIM13, and TIM14 are 16-bit otherwise TIM2 and TIM5 are 32-bit timers. These timers can be used for a diversity of purposes, including input capture or generating output waveforms (output compare and PWM). The block diagram of general purpose timers is given in Figure 10 (www.st.com:, 2021).

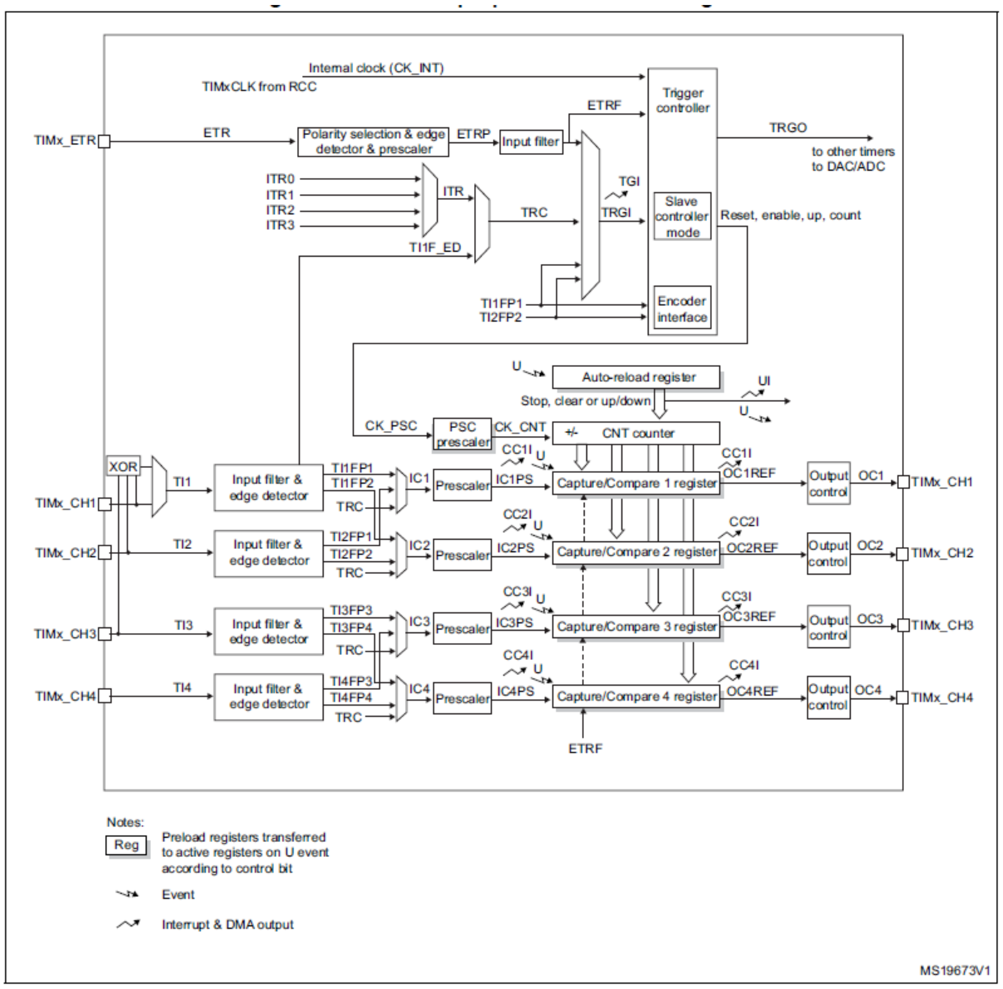


Figure 10. The Block Diagram of General Purpose Timers

The advanced-control timers include TIM1 and TIM8, theses timer are 16-bit auto-reload counter driven by a programmable prescaler and can be used for a diversity of purposes, including input capture or generating output waveforms (output compare and PWM).

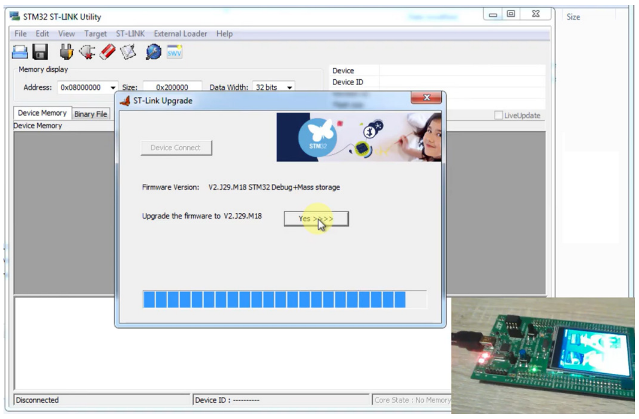
Basic timers include TIM6 and TIM7 timers which are 16-bit auto-reload counter driven by a programmable Prescaler. The TIM6 and TIM7 timers are able to drive digital-to-analog converter (DAC) through their trigger outputs because these timers internally connected to the DAC.

General-purpose input-output pins: The STM32F4 includes 51 to 140 general-purpose input-output pins, three ADCs with 12-bit, 10-bit, 8-bit, or 6-bit configurable resolution, and two Digital-to-analog converter (DAC) which can be configured in 8- or 12-bit mode and can be used in conjunction with the direct memory access (DMA) controller.

Important properties: The STM32F4 includes important properties such as direct memory access (DMA) that increases data transfer speed between memory and memory and between peripherals and memory, improved real-time clock (RTC), random number generator (RNG), cyclic redundancy check (CRC) engine, Ethernet MAC, LCD-TFT controller, and camera interface.

STM32 Programming

The STM32 board can be programmed in a low-level language like assembly and high-level language like c. Moreover, the STM32 board can be programmed with Cube which is used to graphically configure the processor of STM32. While ASM and C programming languages require sufficient knowledge of the processor, memory organization, and instruction set, programs such as Cube that make graphical settings do not require much knowledge about the processor, memory organization, and instruction set. There are many c based debuggers or editors such as Keil (Keil, 2021), MikroC (MikroC, 2021), CooCox (www.st.com, 2021a), and IAR developed (IAR, 2021) to program STM32; however, the TrueSTUDIO (www.st.com, 2021b) program a commercially enhanced C/C++ IDE built on Eclipse, GCC, and GDB is preferred because the TrueSTUDIO program is supported by ST. To get more information and for installing, you can visit the st.com website. As seen in Figure 11, after installing the TrueSTUDIO you should update the firmware of the device from the STM32 ST-LINK utility.



Memory display
Address: 0x08000000 Size: 0x200000 Data Width: 32 bits
Device
Device ID
Device Memory
Firmware Version: V2.J29.M18 STM32 Debug+Mass storage
Upgrade the firmware to V2.J29.M18
Disconnected
Device ID : ----------
Core State : No Memory

Figure 11. Update the Firmware of the STM32

Then we can easily create a project and assign functions to the STM32. The software development flow and the software compilation flow in the STM32 are given in figures 12 and 13, respectively.

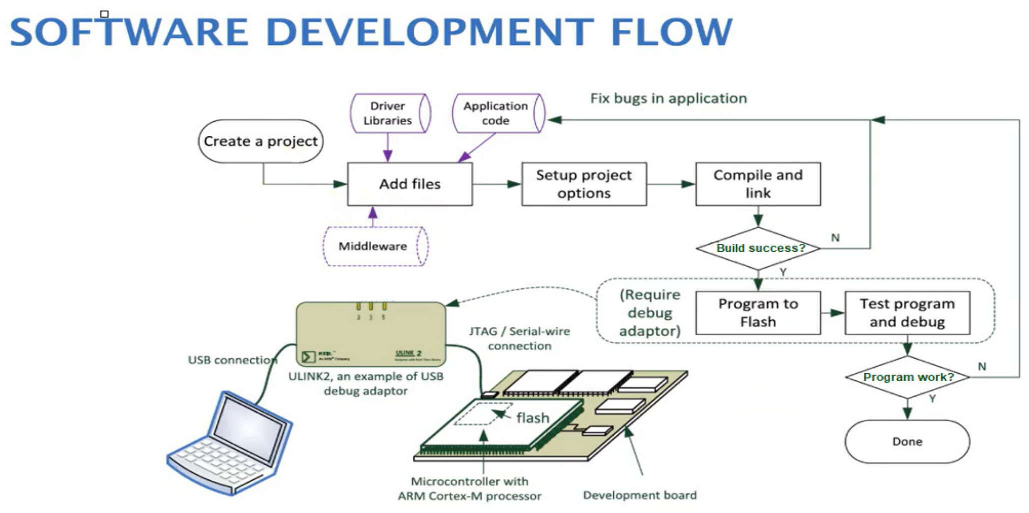


Figure 12. The Software Development Flow

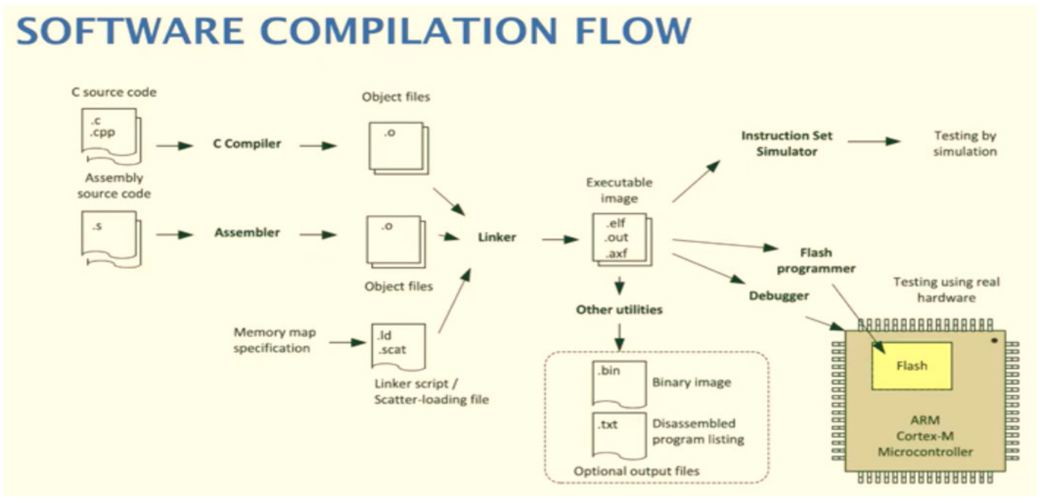


Figure 13. The Software Compilation Flow

As shown in Figures 12 and 13, the software written by the programmer can be embedded in the ARM microcontroller after debugging and linking by the compiler. In the debug process, a file suitable for the ARM processor is prepared by the compiler, this file contains instruction sets in the form of machine code. The instructions are processed sequentially by the ARM processor, and each instruction goes through at least three stages, fetch, decode and execute. Below is a short code about the general-purpose input/output ports of the ARM Microcontroller.

```
// Toggle output pins (D_Pin_0, D_Pin_1, D_Pin_2) with input pin (A_Pin_0)

#include "stm32f4xx.h"

#include "stm32f4_discovery.h"
```

```
GPIO_InitTypeDef GPIO_portstype;

int control = 0;

int main(void)

   {

         RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);

         // For port D (output)

         GPIO_portstype.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2;

         GPIO_portstype.GPIO_Mode = GPIO_Mode_OUT;

         GPIO_portstype.GPIO_OType = GPIO_OType_PP;

         GPIO_portstype.GPIO_PuPd = GPIO_PuPd_NOPULL;

         GPIO_portstype.GPIO_Speed = GPIO_Speed_50MHz;

         GPIO_Init(GPIOD, &GPIO_portstype);


         RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);

         // For port A (input)

         GPIO_portstype.GPIO_Pin = GPIO_Pin_0;

         GPIO_portstype.GPIO_Mode = GPIO_Mode_IN;

         GPIO_portstype.GPIO_OType = GPIO_OType_PP;

         GPIO_portstype.GPIO_PuPd = GPIO_PuPd_DOWN;

         GPIO_portstype.GPIO_Speed = GPIO_Speed_50MHz;

         GPIO_Init(GPIOA, &GPIO_portstype);

 while (1)

   {

         if(GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_0) == Bit_SET)

         {
```

```
                    while(GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_0));
                    if(kontrol == 0)
                    {
                              kontrol = 1;
                              GPIO_WriteBits(GPIOD, GPIO_Pin_0, Bit_SET);
                              GPIO_WriteBits(GPIOD, GPIO_Pin_1, Bit_SET);
                              GPIO_WriteBits(GPIOD, GPIO_Pin_2, Bit_SET);
                    }
                    else
                    {
                              kontrol = 0;
                              GPIO_WriteBits(GPIOD, GPIO_Pin_0, Bit_RESET);
                              GPIO_WriteBits(GPIOD, GPIO_Pin_1, Bit_RESET);
                              GPIO_WriteBits(GPIOD, GPIO_Pin_2, Bit_RESET);
                    }
          }
    } // while
  } //Main
void EVAL_AUDIO_TransferComplete_CallBack(uint32_t pBuffer, uint32_t Size){
  return;
}
uint16_t EVAL_AUDIO_GetSampleCallBack(void){
  return -1;
}
```

References

Ben-Sasson, E., Chiesa, A., Tromer, E., & Virza, M. (2014). Succinct non-interactive zero knowledge for a von Neumann architecture. 23rd USENIX Security Symposium (USENIX Security 14),

Fleisher, C. S., & Bensoussan, B. E. (2015). Business and competitive analysis: effective application of new and classic methods. FT press.

IAR. (2021). https://www.iar.com/ewarm.

Iturbe, X., Venu, B., Penton, J., & Ozer, E. (2017). A" high resilience" mode to minimize soft error vulnerabilities in ARM cortex-R CPU pipelines: work-in-progress. Proceedings of the 2017 International Conference on Compilers, Architectures and Synthesis for Embedded Systems Companion,

Keil. (2021). https://www.keil.com/.

Mariantoni, M., Wang, H., Yamamoto, T., Neeley, M., Bialczak, R. C., Chen, Y., Sank, D. (2011). Implementing the quantum von Neumann architecture with superconducting circuits. Science, 334(6052), 61-65.

Martin, T. (2016). The designer's guide to the Cortex-M processor family. Newnes.

Microcontrollerslab. (2021). https://microcontrollerslab.com/stm32f4-discovery-board-pinout-features-examples/.

MikroC. (2021). https://www.mikroe.com/mikroc-arm.

Ngabonziza, B., Martin, D., Bailey, A., Cho, H., & Martin, S. (2016). Trustzone explained: Architectural features and use cases. IEEE 2nd International Conference on Collaboration and Internet Computing.,

Wan, J., Wang, R., Lv, H., Zhang, L., Wang, W., Gu, C., . . . Gao, W. (2012). AVS video decoding acceleration on ARM Cortex-A with NEON. IEEE International Conference on Signal Processing, Communication and Computing (ICSPCC 2012),

www.st.com. (2021a). https://www.st.com/en/development-tools/coide.html.

www.st.com. (2021b). https://www.st.com/en/development-tools/truestudio.html.

www.st.com:. (2019). STM32 Cortex-M4 MCUs and MPUs programming manual.

www.st.com:. (2020). Discovery kit with STM32F407VG MCU (User manual UM1472).

www.st.com:. (2021). STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced Arm-based 32-bit MCUs (Reference manual RM0090).

Yiu, J. (2020). Definitive Guide to Arm Cortex-M23 and Cortex-M33 Processors. Newnes.

About the Authors

Mohammed H. IBRAHIM is Assistant Professor of computer engineering at Necmettin Erbakan University in Konya-Turkey. He received a PhD in Computer Engineering in 2019 from Selçuk University. Her research aims at developing methods in data mining and machine learning and the application of data mining and machine learning methods, as well as embedded system and control.

E-mail: mibrahim@erbakan.edu.tr Orcid: 0000-0002-6093-6105

Similarity Index

The similarity index obtained from the plagiarism software for this book chapter is 12%

To Cite This Chapter:

Ibrahim, M. H. (2021). STM32. In S. Kocer, O. Dundar & R. Butuner (Eds.), *Programmable Smart Microcontroller Cards* (pp. 67–81). ISRES Publishing.