elektor books

Advanced Programming with STM32 Microcontrollers

Master the software tools behind the STM32 microcontroller

Majid Pakdel

elektor
design > share > sell

Advanced Programming with STM32 Microcontrollers

Majid Pakdel

design > share > sell

an Elektor Publication

● This is an Elektor Publication. Elektor is the media brand of
Elektor International Media B.V.
78 York Street
London W1H 1DP, UK
Phone: (+44) (0)20 7692 8344

First published in the United Kingdom 2020

● British Library Cataloguing in Publication Data
Catalogue record for this book is available from the British Library

● ISBN: 978-3-89576-410-3

● EISBN: 978-3-89576-411-0

● EPUB: 978-3-89576-412-7

Prepress production: DMC ¦ daverid.com
Printed in the Netherlands by Wilco

Table of Contents

• Preamble

This book covers topics related to various programming methods, hardware, and software tools for the STM32. Various software tools have been used in the book, including Keil MDK ARM (uVison5 IDE), IAR Embedded Workbench for ARM, SW4STM32, Mbed online compiler, mikroC PRO for ARM, Arduino IDE (STM32duino), and MATLAB/Simulink embedded coder. Most of these are freely available over the internet. Proteus has been used for simulation of some programs and in PCB design of the proposed STM32F103RET6 based development board mentioned in chapter 1. Readers can use the STM32F103RET6 based header board, or lower cost, cheaper header boards including blue pill (STM32F103C8T6), NUCLEO-F103RB, STM32F030F4P6 Mini Development Board, Discovery kit with STM32F407VG MCU, etc. for hardware implementation of the programs contained in this book.

Chapter Overview:

Chapter 1
Introduction to STM32 ARM Cortex-M 32-bit Microcontrollers: An Introduction to various software tools used in STM32 microcontroller programming. Serial programming in boot mode is explained in detail.

Chapter 2
Beginner programming with STM32: Simple projects with SMT32CubeMX, IAR Embedded Workbench for ARM, and simulation using Proteus.

Chapter 3
Adding LCD and Delay Libraries to a Project: Learning how to add a library including the start-up functions of different external hardware and various functions to a project where they are not in HAL libraries.

Chapter 4
External Interrupts in STM32 Microcontrollers: External interrupts and their settings are discussed in detail.

Chapter 5
Analogue to Digital Converters (ADCs) in STM32 Microcontrollers: Analogue to digital converters (ADCs) are very important tools for designing hardware that produces analogue signals and their connection to STM32 microcontrollers. ADC settings are explained in detail.

Chapter 6
Digital to Analogue Converters (DACs) in STM32 Microcontrollers: Digital to analogue converters (DACs) operate inversely to analogue to digital converters and convert digital values to analogue. DAC settings are discussed in detail.

Chapter 7
Timers and Counters in STM32 Microcontrollers: All timing, scheduling, and delays

without stopping the program are managed by a timer operating in parallel with the processor. A timer is a counter that can count the regular pulses of an oscillator and generate regular times. This counter can count the external pulses applied to a microcontroller. Timer and counter settings are explained in detail.

Chapter 8
Pulse Width Modulation (PWM) in STM32 Microcontrollers: PWM wave is generated in Timer units which can produce PWM signals. PWM settings are discussed in detail.

Chapter 9
Real-Time Clock (RTC) in STM32 Microcontrollers: The Real-Time Clock (RTC) is one of the intelligent system design requirements for accessing real-time and date. RTC settings are discussed in detail.

Chapter 10
Serial Communication in STM32 Microcontrollers: Serial communication is one of the important sections in communication between a microcontroller and other devices. The serial communication protocol settings are explained in detail.

Chapter 11
Synchronous Peripheral Interface (SPI) in STM32 Microcontrollers: The SPI protocol is a synchronous bidirectional serial communication interface. This interface is one of the important and useful communication protocols in microcontrollers since it operates at a very high speed. SPI protocol settings are explained in detail.

Chapter 12
Watchdog Timer in STM32 Microcontrollers: The watchdog timer is one of the important fundamentals in microcontrollers. It is used to reset the CPU when it hangs. STM32 microcontrollers have two watchdog timer units, called the Independent Watchdog (IWDG) and Window Watchdog (WWDG). Watchdog timer settings are discussed in detail.

Chapter 13
Inter-Integrated Circuit (I^2C) in STM32 Microcontrollers: The Inter-Integrated Circuit (I^2C) is a serial bus protocol consisting of two signal lines known as SCL and SDL. These are used for communication. I^2C protocol settings are explained in detail.

Chapter 14
Direct Memory Access (DMA) in STM32 Microcontrollers: To avoid holding the CPU processing time, most advanced microcontrollers have a Direct Memory Access (DMA) controller. Data transfers using DMA are implemented between memory locations without the need for CPU. DMA settings for different projects are discussed in detail.

Chapter 15
Real-Time Operating System (RTOS) in STM32 Microcontrollers: This chapter focuses on how to use a real-time OS (RTOS) to create a real-time application on an STM32 microcontroller.

Chapter 16
STM32 Microcontrollers Programming Using STM32duino: STM32 microcontrollers are particularly popular because they are programmable using the Arduino IDE (STM32duino). In this way, everybody can easily get started and build projects with STM32. In this chapter, we use the Arduino IDE to program the STM32

Chapter 17
Finite State Machine (FSM) Programming in STM32 Microcontrollers: A common design technique used in industrial automation control systems is the venerable finite state machine (FSM). Industrial automation control designers use this programming method to break complex problems into manageable states and transitions. The FSM programming method is explained using mikroC PRO for ARM and Mbed online compilers in detail.

Chapter 18
STM32 Microcontrollers Programming Using MATLAB/Simulink: The user is rapidly able to produce a model using Simulink that would otherwise require hours to build in the laboratory environment. Simulink supports linear and non-linear systems, modelled in continuous time, sampled time, or a hybrid of the two. In this chapter, we show how to generate code for STM32 microcontrollers using Simulink.

Chapter 1 ● Introduction to STM32 ARM Cortex-M Microcontrollers

1.1 ● Introduction

The STM32 series is a popular, cheap, high-performance microcontroller variant. They also have lots of support from various microcontroller development software suites. STM32 microcontrollers offer a large number of peripherals that can be interfaced with all kinds of electronic components including sensors, displays, electric motors, etc. The range of performance available with the STM32 is very extensive. Some of the most basic STM32 microcontrollers (STM32F0 and STM32F1 series) start with a 24 MHz clock frequency and are available in packages with as few as 16 pins while the STM32H7 operates at up to 400 MHz, and is available in packages with 240 pins. Moreover, the STM32L series has been designed for low-power applications running from a small battery. Development software is required to develop code, program the microcontroller, and debug. Development tools must include a compiler, debugger, and an in-circuit serial programmer (ICSP) as shown in Figure 1.1.

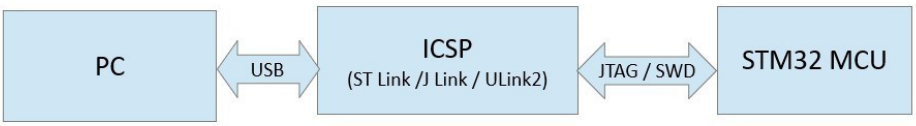

Figure 1.1: ICSP programmer diagram

1.2 ● Software Tools

There are several software tools available for programming STM32 microcontrollers. Most are available as Integrated Development Environments (IDE).

Common programming software tools include:

- Keil MDK ARM (uVison5 IDE) – The MDK ARM IDE is a very stable development environment allowing the development of programming code.
- IAR Embedded Workbench for ARM – IAR Embedded Workbench and the included IAR C/C++ compiler generates the fastest performing, most compact code in the industry for ARM-based applications.
- SW4STM32 – The System Workbench toolchain, called SW4STM32, is a free multi-OS software development environment based on Eclipse, which supports the full range of STM32 microcontrollers and associated boards. The SW4STM32 toolchain may be obtained from the website www.openstm32.org, which includes forums, blogs, and training for technical support. The System Workbench toolchain and its collaborative website have been built by AC6, a service company providing training and consultancy on embedded systems.

- Mbed online compiler – Jump directly into application development with Mbed OS without installing anything. Create a Mbed account to get started.
- CoIDE – A free toolchain based on the Eclipse IDE integrated along with an embedded ARM version of the free GCC compiler.
- mikroC PRO for ARM – C compiler for writing fast multimedia applications for ARM Cortex M3 and M4 devices using mikroC programming environment.
- Arduino IDE – STM32duino (https://github.com/stm32duino/wiki/wiki/Getting-Started).
- Matrix-Flowcode 8 – ARM MCUs including the popular STM32 ARM family are also supported in Flowcode.
- MATLAB/Simulink embedded coder or supported hardware packages including STM32 development boards.

ARM Cortex-M microcontrollers support two programming protocols: JTAG (named by the electronics industry association the Joint Test Action Group) and Serial Wire Debug (SWD). The block diagram of peripheral devices connected to a microcontroller with some communication protocols (which is called a system-level block diagram) is shown in Figure 1.2.

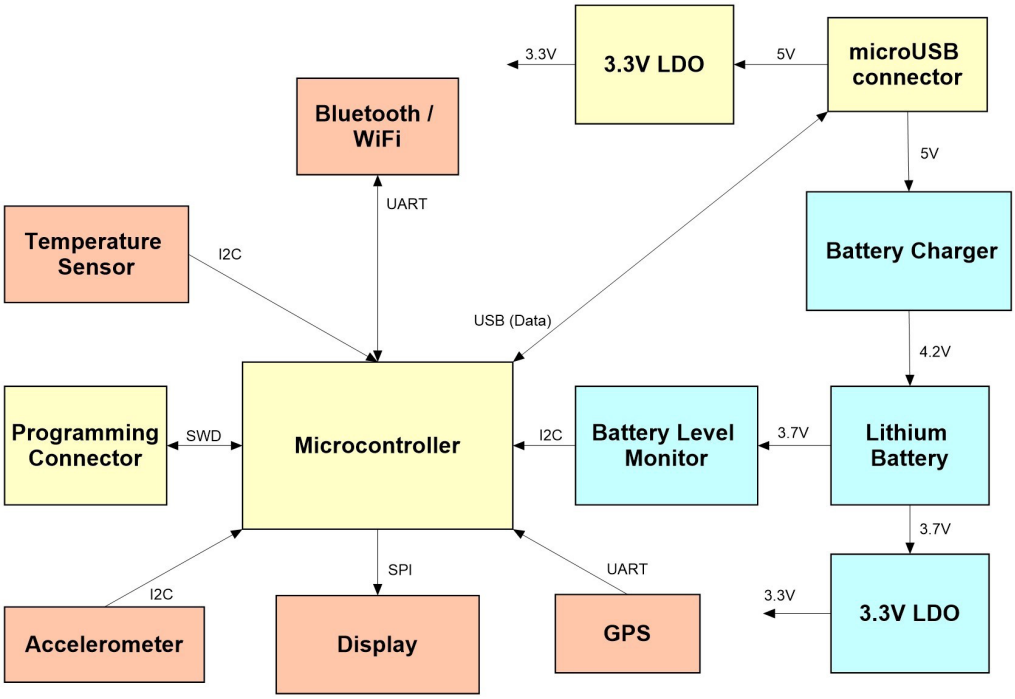

Figure 1.2: System-level block diagram

ARM Cortex-M is a 32-bit microcontroller which is generally the best choice for computationally intensive tasks compared to what is available from older 8-bit microcontrollers such as

the 8051, PIC, and AVR microcontrollers. ARM microcontrollers come in multiple types including the Cortex-M0, M1, M3, M4, and M7. Some versions are available with a Floating Point Unit (FPU) and are designated with an F in the Cortex-M4F microcontrollers. One of the benefits of Arm Cortex-M processors is their low price and high performance. Even if an 8-bit microcontroller is sufficient for your application, it might be better to consider a 32-bit Cortex-M microcontroller. There are Cortex-M microcontrollers available with very comparable pricing to some of the older 8-bit chips. A 32-bit microcontroller gives you more flexibility to extend your program and add additional features when required. STM32 microcontrollers can be divided into several subseries as shown in Figure 1.3.

STM32 Series	Cortex-Mx	Max clock (MHz)	Performance (DMIPS)
F0	M0	48	38
F1	M3	72	61
F3	M4	72	90
F2	M3	120	150
F4	M4	180	225
F7	M7	216	462
H7	M7	400	856
L0	M0	32	26
L1	M3	32	33
L4	M4	80	100
L4+	M4	120	150

Figure 1.3: Comparison of various STM32 microcontrollers

1.3 ● STM32 Hardware Design and Serial Programmer

A typical STM32F103RET6 based development board is shown in Figure 1.4. A USB to serial converter IC (FT232RL) is used to produce a virtual serial port when connecting to the PC via USB port. The TXD and RXD pins with FT232RL are connected to the RXD1 pin (PA10) and TXD1 pin (PA9) of the microcontroller. Since the FT232RL IC is generally expensive, we can use the PL2303 IC instead, as shown in Figure 1.5, or the PL2303-USB to TTL adapter module as shown in Figure 1.6. The programming of the microcontroller can be done by this virtual serial port. For programming the microcontroller, it should be set to BOOT mode. If the BOOT0 pin of the microcontroller is connected to VCC3.3 and its BOOT1 pin is connected to GND and the microcontroller is reset once using the RESET button, the microcontroller enters BOOT mode. In this mode, the microcontroller can be programmed through the virtual serial port. After programming the microcontroller for exiting from BOOT mode, BOOT0 should be connected to GND. In this case, BOOT1 can be used freely by the user. In normal operation, the BOOT0 pin of the microcontroller should be connected to GND. The STM32 Flash Loader Demonstrator software as illustrated in Figure 1.7 is used for serial port programming. For programming using this software, the microcontroller should be in BOOT mode. After programming, the microcontroller should exit from BOOT mode.

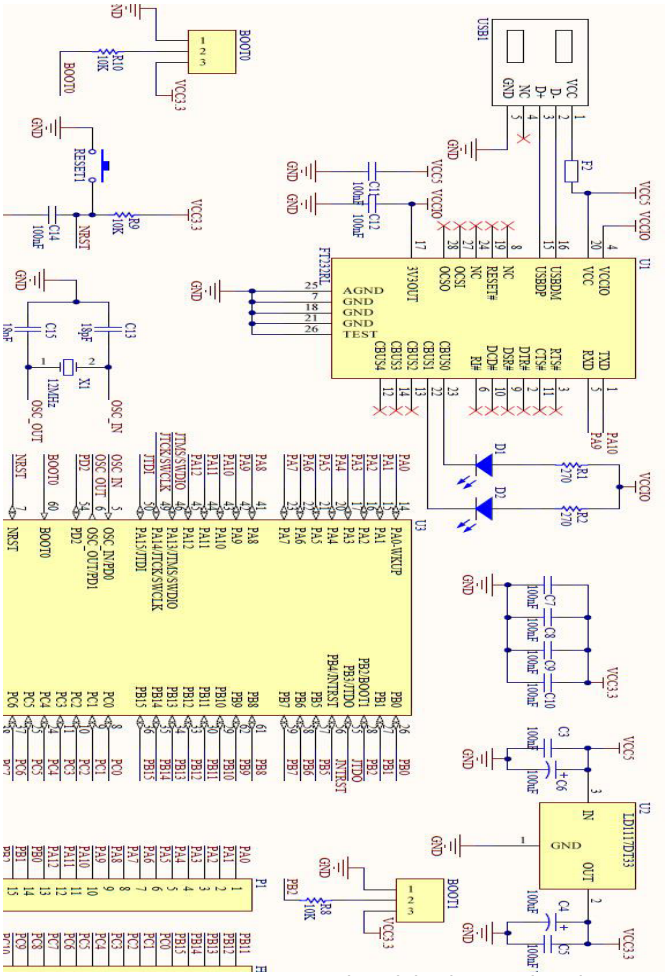

Figure 1.4: STM32F103RET6 based development board

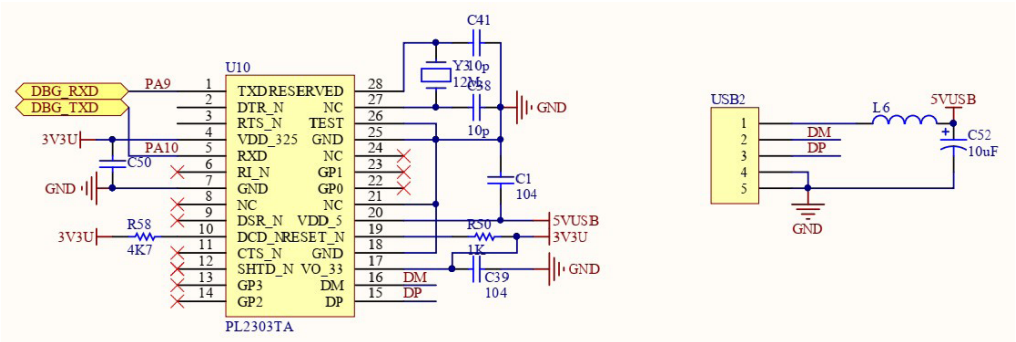

Figure 1.5: STM32F103RET6 serial programming using PL2303

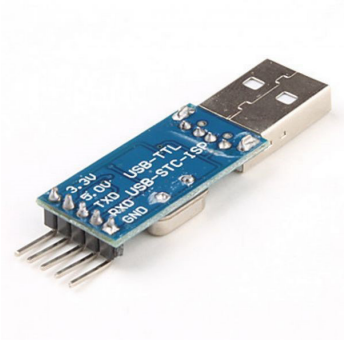

Figure 1.6: PL2303-USB to TTL adapter module

Figure 1.7: STM32 flash loader demonstrator software

1.4 ● Proposed STM32F103RET6 Header Board

The proposed minimum design schematic for the STM32F103RET6 microcontroller, the PCB layout using Proteus software, and the hardware implementation of the header board designed for the STM32F103RET6 microcontroller are shown in Figures 1.8, 1.9, and 1.10 respectively.

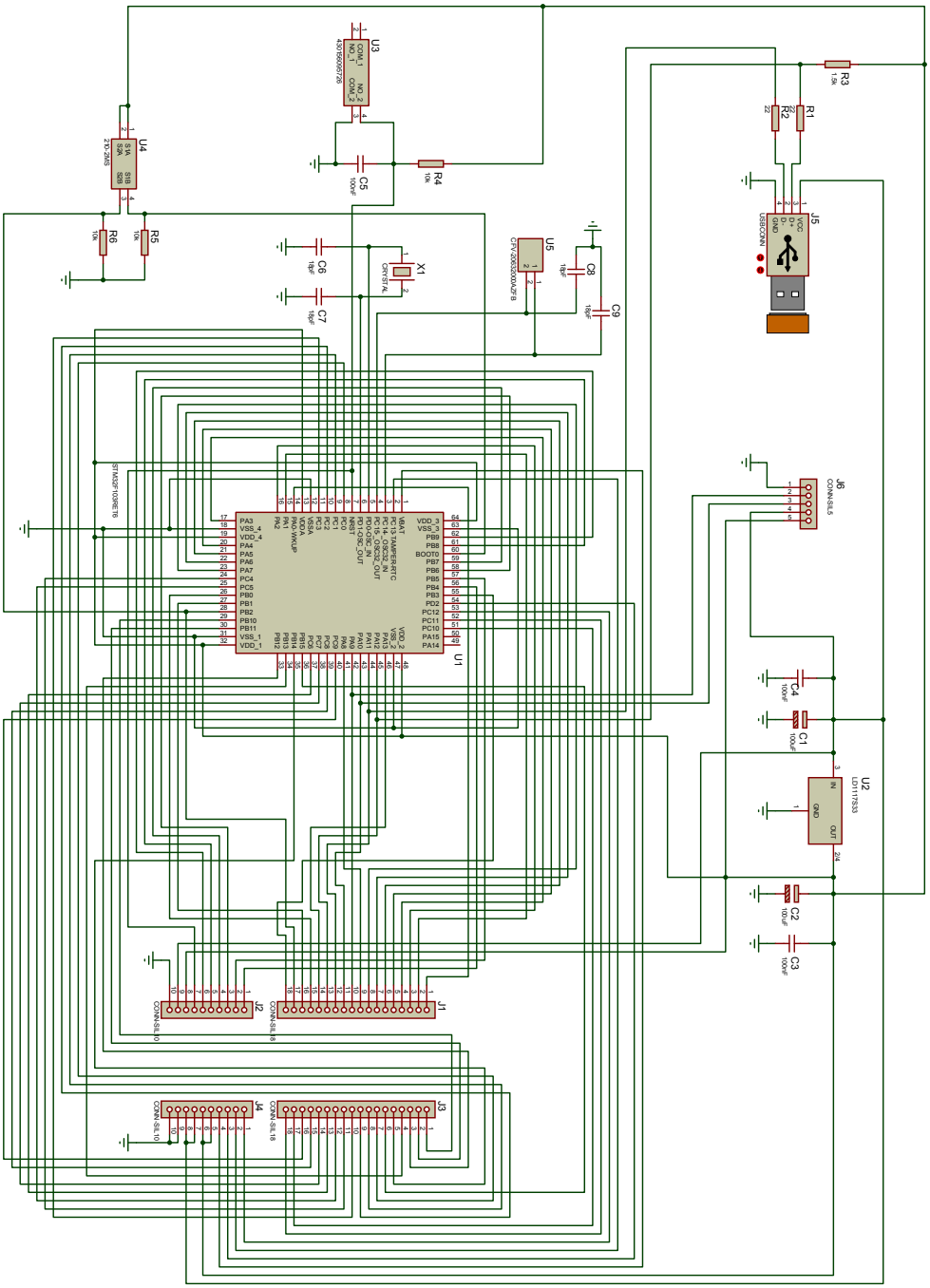

Figure 1.8: The proposed minimum designed schematic for the STM32F103RET6 microcontroller

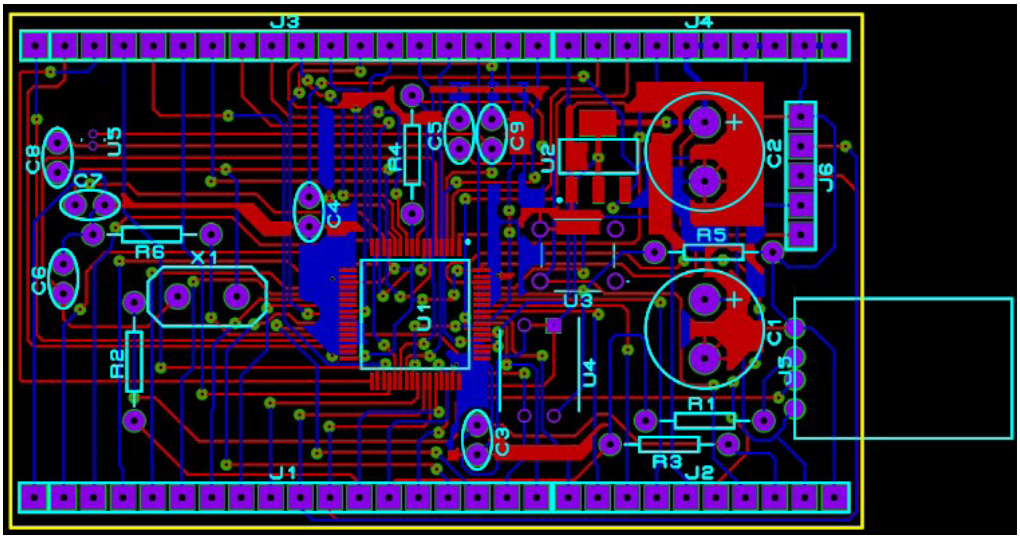

Figure 1.9: The PCB layout for the proposed STM32F103RET6 microcontroller header board

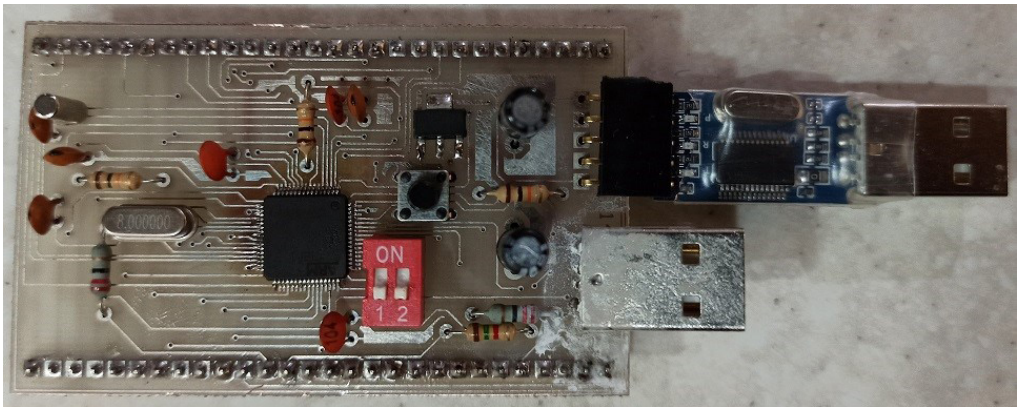

Figure 1.10: Hardware implementation of the designed header board using the STM32F103RET6 microcontroller

1.5 ● Summary

The STM32 microcontroller series is well-liked and used in a wide variety of applications. They are inexpensive and have high-performance capabilities. They also support various microcontroller development software suites. The ARM Cortex-M is a 32-bit microcontroller which is the best choice for more computationally intensive tasks compared to what is available from older 8-bit microcontrollers such as the 8051, PIC, and AVR microcontrollers. An Introduction to various software tools for STM32 microcontrollers programming and serial programming in boot mode was explained in detail. Readers can use the proposed STM32F103RET6-based header board, or lower cost, cheaper alternatives such as blue pill (STM32F103C8T6), NUCLEO-F103RB, STM32F030F4P6 Mini Development Board, Discovery

kit with STM32F407VG MCU, etc. Hardware implementations of programs are presented in the next chapters.

Chapter 2 ● Beginning First Projects with STM32 Microcontroller

2.1 ● Introduction

Firstly, STM32CubeMX software should be installed and executed. For creating a new project, the libraries of the specifically used microcontroller family should be installed. From the Help menu, install NEW Libraries, as shown in Figure 2.1.

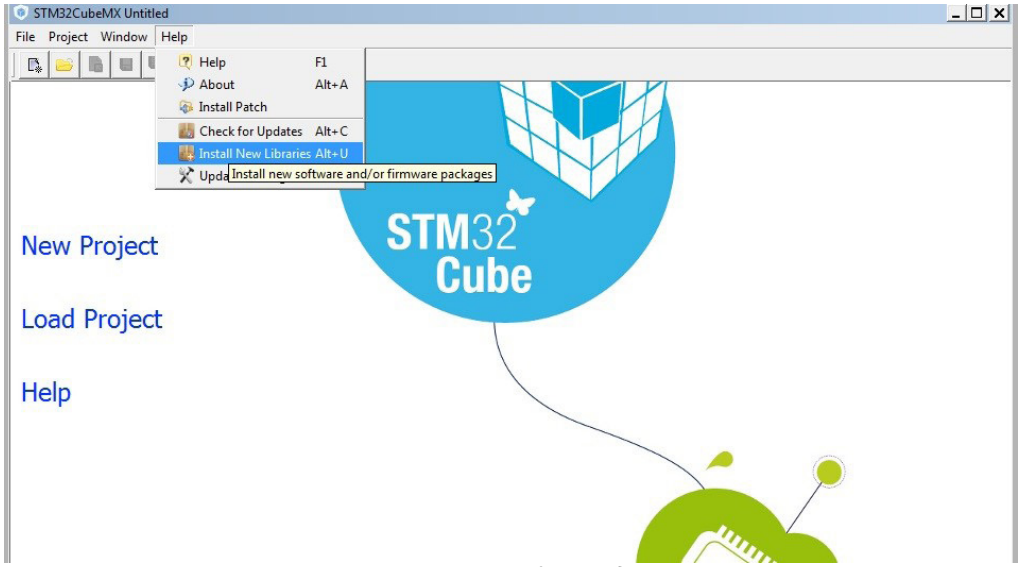

Figure 2.1: STM32CubeMX software

In the pop-up window, choose the Firmware Package for Family STM32F1 and click the 'Install Now' button as shown in Figure 2.2. After installation, a green square is shown beside the selected icon.

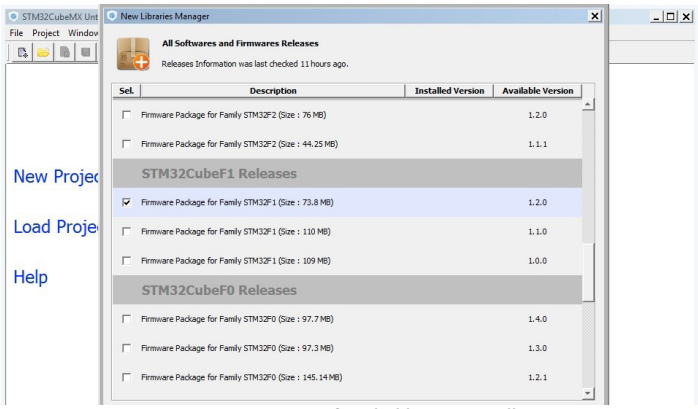

Figure 2.2: STM32F1 family library installation

2.2 • Beginning the First Project

After installation of the STM32F1 family library, come back to the software main window and select the 'New Project' icon to open the New Project window. From the MCU Selector tab, the STM32F103RETx is selected as illustrated in Figure 2.3.

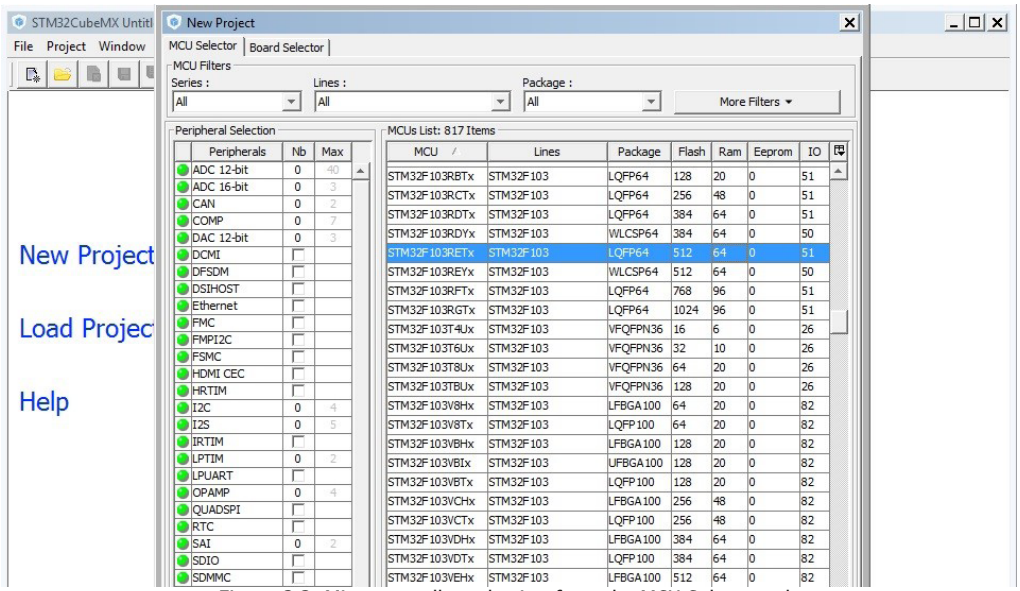

Figure 2.3: Microcontroller selecting from the MCU Selector tab

After clicking the OK button, a window pops up as shown in Figure 2.4.

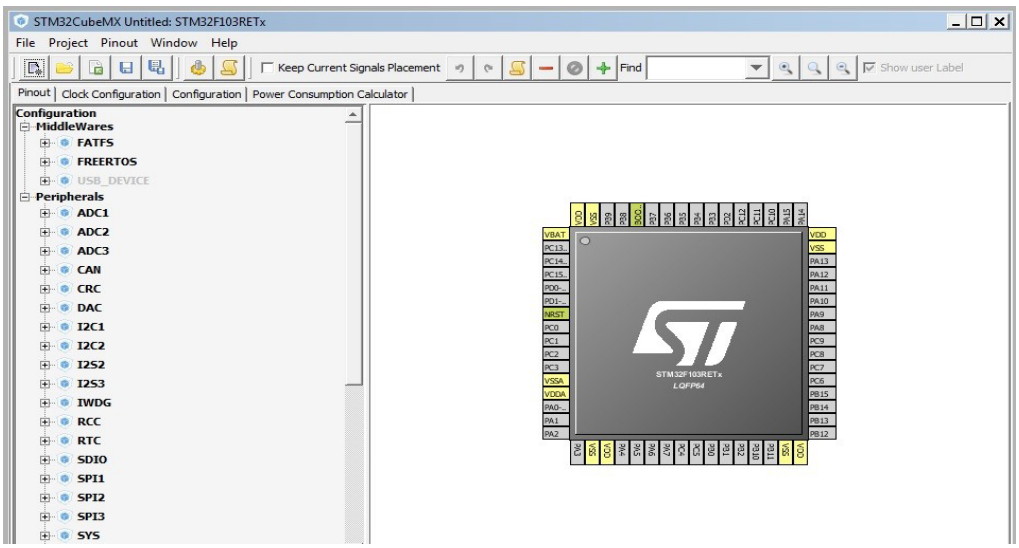

Figure 2.4: The STM32F103RETx microcontroller

We click on the PA2 pin and select the 'GPIO_Output' icon as shown in Figure 2.5.

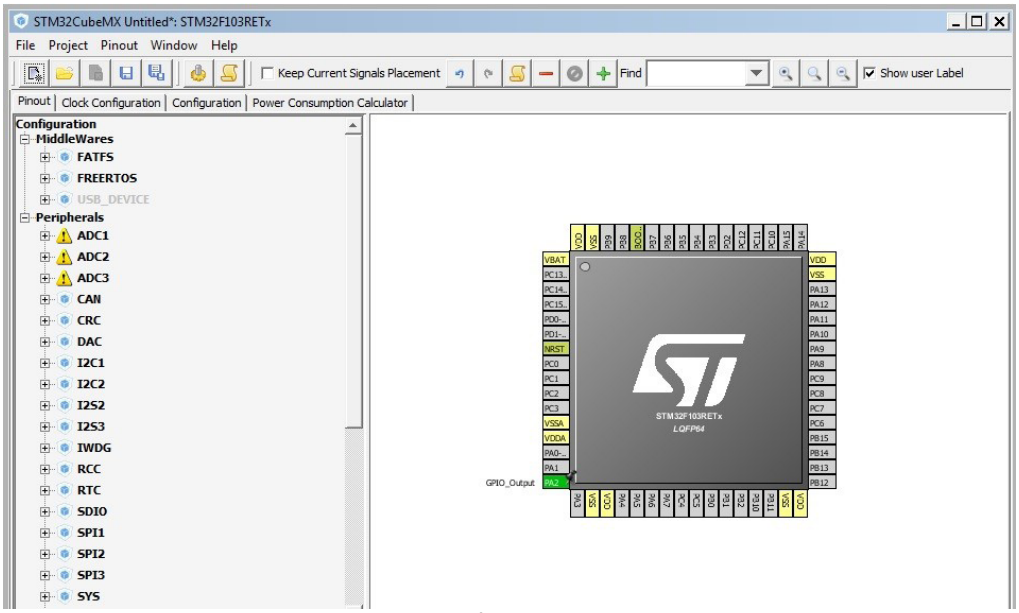

Figure 2.5: Setting the PA2 pin as GPIO_Output

The Clock Configuration tab is used for setting the frequency of the microcontroller clock. This microcontroller has two internal oscillators (8 MHz and 40 kHz). In Figure 2.6, HSE and LSE correspond to High-Speed and Low-Speed External which are related to the supplied clock with an external resonator. In the same manner, HSI and LSI represent High-Speed Internal and Low-Speed Internal respectively.

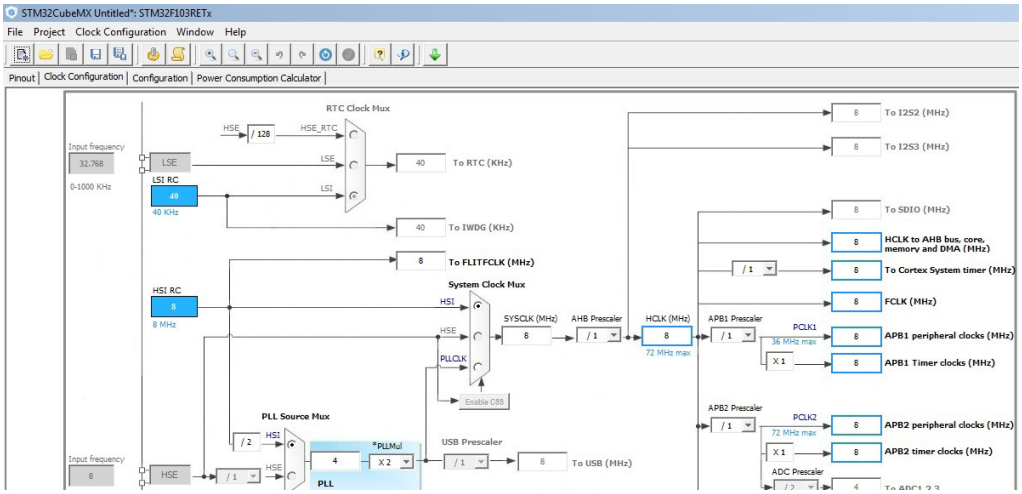

Figure 2.6: The Clock Configuration tab

The Configuration tab is used for setting the hardware enabled in the Pinout section. The GPIO button is in the system section as illustrated in Figure 2.7. When clicking on the GPIO button, the Pin Configuration window pops up, as shown in Figure 2.8.

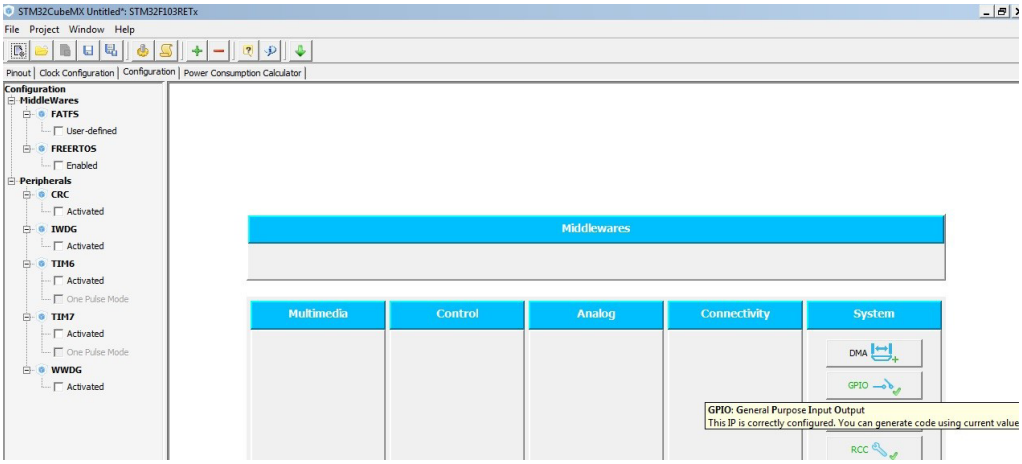

Figure 2.7: The Configuration section

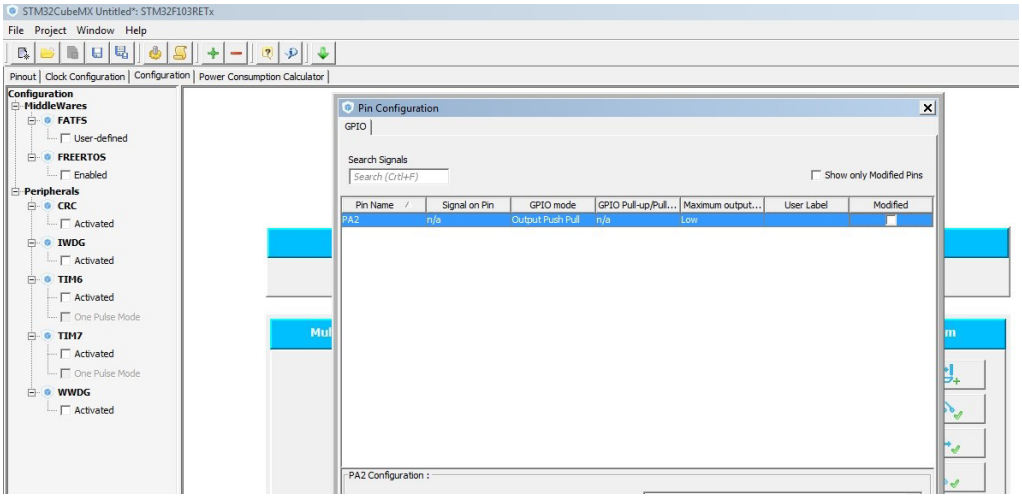

Figure 2.8: GPIO in the Pin Configuration section

In the Power Consumption Calculator section, the user can calculate the power consumed according to the hardware enabled by using a steady supply or battery and generating a report. If the power is supplied from a battery, the battery used should be selected by clicking on the 'Select Battery' button as shown in Figure 2.9.

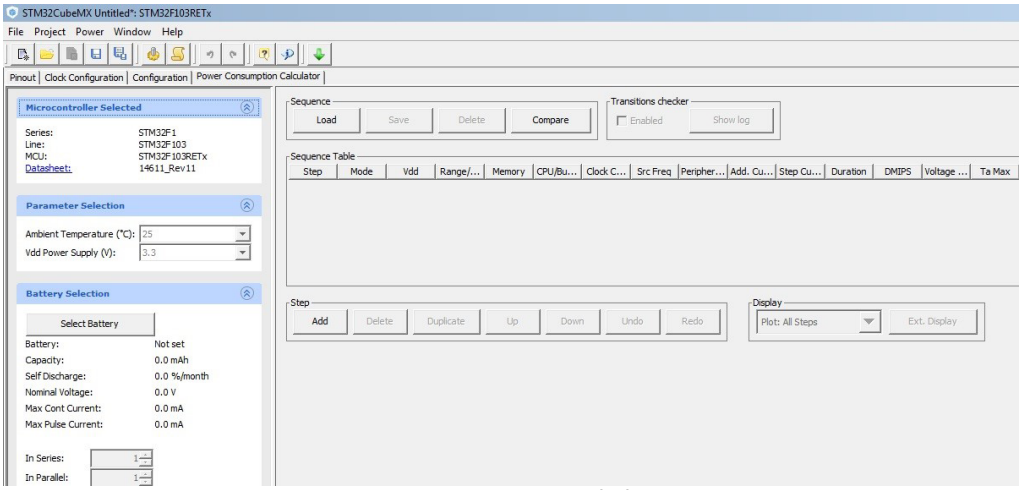

Figure 2.9: Power Consumption Calculator section

In the 'Available batteries' window, as shown in Figure 2.10, there is a list of batteries that the user can select. If the battery used is not available on the list, the user can add it by clicking on the 'Add Battery' button. The user can then specify the characteristics of the battery used in the pop window as shown in Figure 2.11.

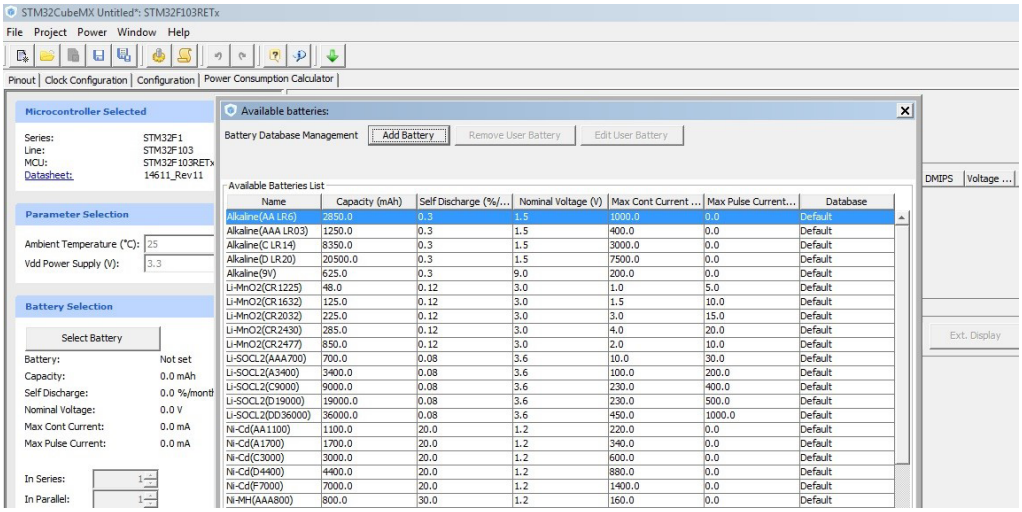

Figure 2.10: Used battery selection from the available battery list