bare-metal firmware development for ARM-based microcontrollers

JUNAID HASAN MOHAMMED NIJAMUDEEN
ELECTRICAL AND COMPUTER ENGINEERING
ONTARIO TECH UNIVERSITY
OSHAWA, CANADA
JUNAIDHASAN.MOHAMMEDNIJAMUDEEN@ONTARIOTECHU.NET

Abstract— **With a programming-based method, this project is intended to build a solid foundation in firmware development of ARM microcontrollers. This work uses the STMicroelectronics STM32F4-NUCLEO which has an ARM Cortex-M4 microcontroller.**

I. LITEREATUTE REVIEW

ARM is a design company based in Cambridge, UK. They don’t manufacture microcontrollers where as they come up with the designs of architecture and sell the authorization to the important semiconductor builders across the globe.

ARM is accustomed to the RISC perspective, which is based on implementing multiple instructions in one clock cycle. Implementing multiple instructions in parallel adds complexity to the compiler, but CISC architecture adds complexity to the processor. RISC is implemented using four basic design strategies. H. Reduce the number of cycles required for an instruction, use the pipeline to execute multiple instructions, use a large number of general-purpose registers, and use registers for load and store operations instead of memory.

The STM32 device is based on CMSIS ("Cortex microcontroller interface standard") developed by ARM. CMSIS enables communication with peripherals, real-time operating systems, and middleware components, enables software reusability and portability, and provides a platform for debugging.

STM 32 microcontroller offers a wide range of serial and parallel communication peripherals which could be used to connect to a wide range of electronic devices such as sensors, camera, LCD, motors. Here are some of the types of the STM32 board in the market ordered in terms of their simplicity.

- Bluepill
- Discovery
- Nucleo
- EVAL

The bluepill being the most popular one and it is so small it can directly fit into a bread board. EVAL board being the most expensive one with wide range of features.

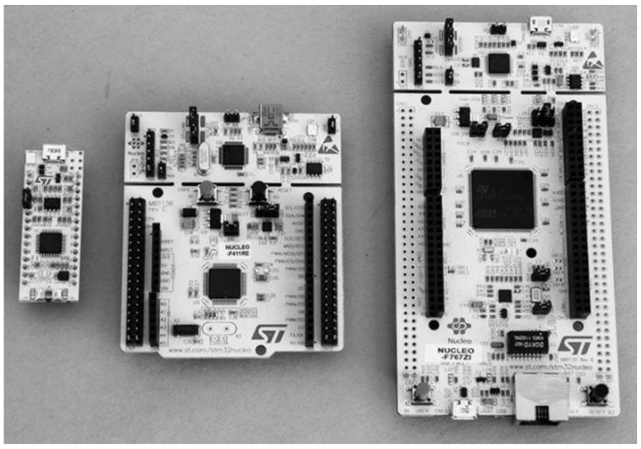


Figure 1: Three basic nucleo boards

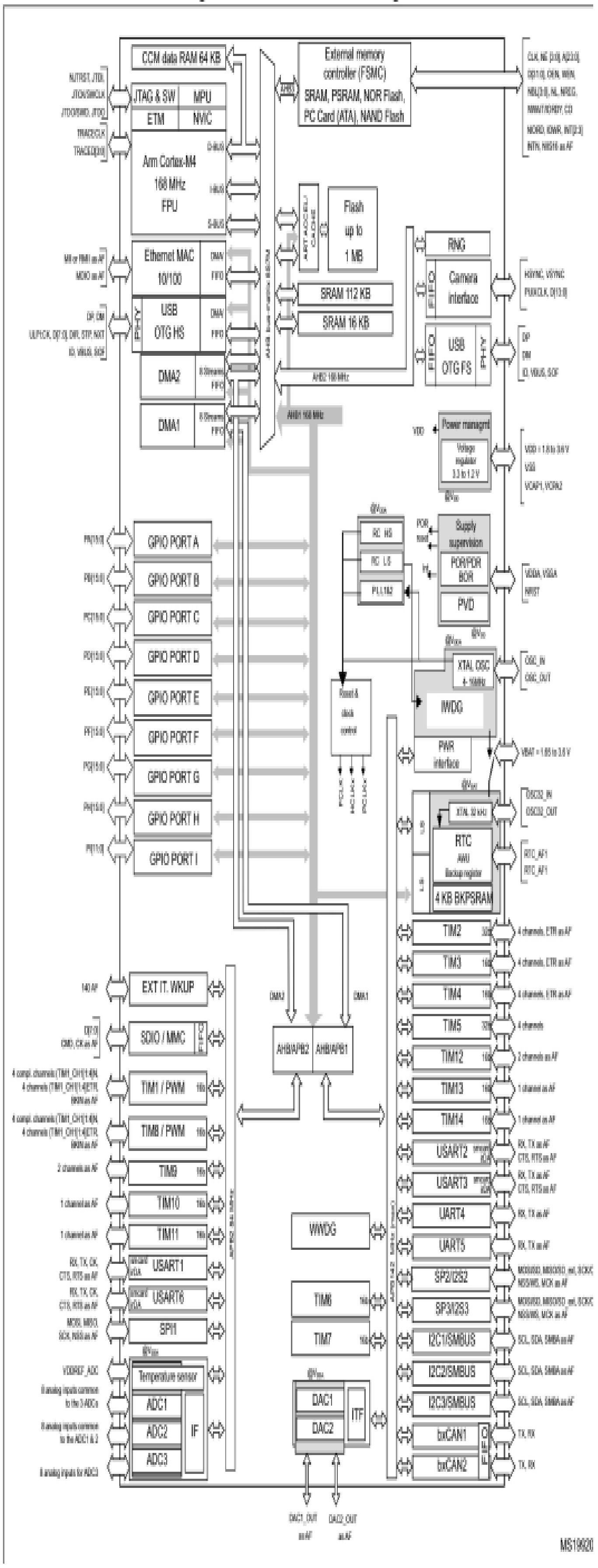
The three boards represent NUCLEO-32, NUCLEO-64, NUCLEO-128, from left to right and the number in each board represents the number of pins present in the microcontroller chip, this project uses NUCLEO-64 board for experiment and results.

II. FUNCTIONAL OVERVIEW

STM32F401xD Features:

- Core is made of ARM-M4 processor.
- The microcontroller chip in PQFP package64 terminals.
- One 12-bit ADC module with 19 channels
- CRC calculation unit
- 512kb flash
- 128kb RAM
- 16-stream DMA with FIFO and burst support
- 5 serial peripheral interfaces
- RTC: sub second accuracy, hardware calendar
- 8 timers
- 3 USART module [1][2]

Figure 5. STM32F40xxx block diagram



III. PROJECT PROPOSAL

STM32 is a STMicroelectronics line of 32-bit microcontroller integrated chips.

This project's purpose is to learn STM 32 BareMetal programming. I feel that at the end of the course, I will have a solid understanding of the microcontroller development environment. The project is based on the STM32F4-Nucleo board from STMicroelectronics, which contains an ARM Cortex-M4 microprocessor.

This project teaches me how to read and analyze datasheets and reference manuals for microcontrollers.

In outline, this project work would an excellent chance to

1. Research CMSIS ("Cortex-microcontroller interface standard") a leading professional firmware development standard.
2. Understand the Cortex-M architecture.
3. Compose Analog-to-digital converter (ADC) drivers utilizing bare-metal embedded C language.
4. Compose Pulse width modulator (PWM) driver utilizing embedded C.
5. Compose UART drivers utilizing embedded C.
6. Compose TIMER drivers utilizing embedded C.
7. Compose General purpose input output driver (GPIO) driver utilizing embedded C.
8. Compose Interrupt drivers utilizing embedded C.
9. Compose Serial Peripheral interface (SPI) driver utilizing embedded C.
10. Understand every single line of code and use the debugger effectively to resolve any bugs.
11. Analyze the chip documentation.
12. Create registers from addresses.
13. Locate the peripherals address range in documentation and effectively outline in my code.
14. Master STM32 IDE software environment
15. Gain expertise in embedded development skills and build a foundation in the professional environment.

As a first step, I gathered the paperwork of the STM32 nucleo user guide, data sheet, and reference manual, which contains the addresses of the different registers and peripherals, and I downloaded the STM 32 IDE from the STM website. Over the next several weeks, I'll be able to learn how and why to write firmware for the STM 32.

IV. Methodology

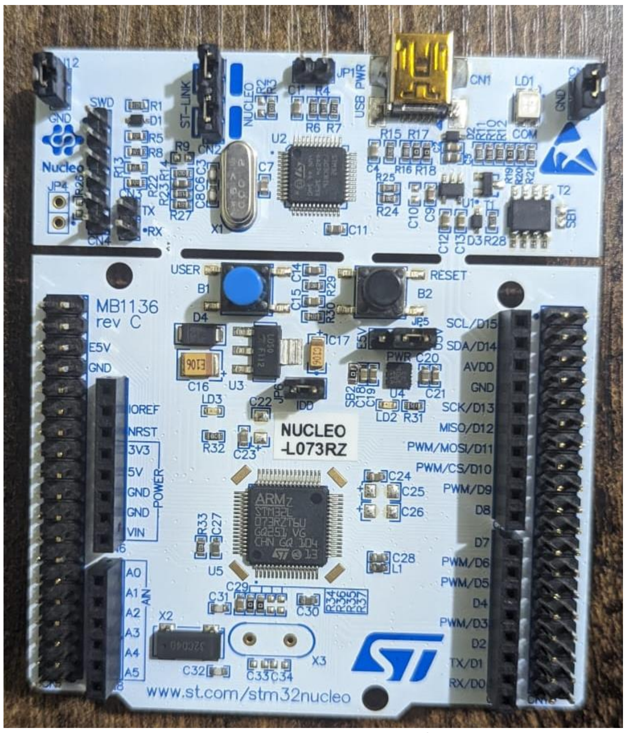


Figure 4: NUCLEO L073RZ board used for experiments

The objective on this project is to design experiments of different peripherals of STM32 L073RZ board, The microcontroller has a 32-bit RISC high performance system, consisting of a core ARM family cortex-M4processor. The design experiments are to realize different set of STM32 peripherals: -

1) Timer/counter: The timers is responsible for various operations, The universal timer is used for precise timing to blink LED ON or OFF, and program it to realize the LED turn ON and OFF in seconds.

2) USART serial communication: The communication between serial port of computer and the USART of STM32, . The computer sends characters to the serial port of STM32 through the keyboard. STM32 transmits the received characters back to the computer, and displays the results through the serial port debugging assistant on the computer

3) DMA reading on-chip flash: The contents of the 32bit data buffer in the processor's on-chip Flash are transferred to the buffer defined in the RAM using the DMA channel.

4) ADC voltage acquisition: The voltage on the potentiometer is collected by ADC of STM32, and the collected results are sent to the computer through USART. The collected results are observed by serial debugging assistant. [8]

The STM32CubeIDE is a integrated development environment (IDE) to develop code for all STM32MCU's ranging from NUCLEO, discovery or custom board and it allows much better control over the MCU. This IDE uses development code in C/C++ language.

A sample of working space of IDE is shown in the figure 5 about the toggling of LED buttons using GPIOA peripherals.

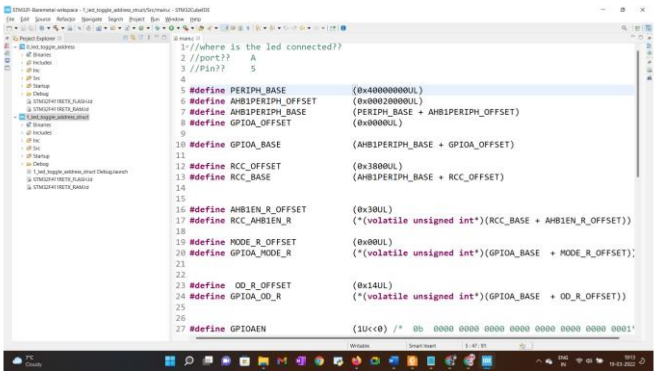
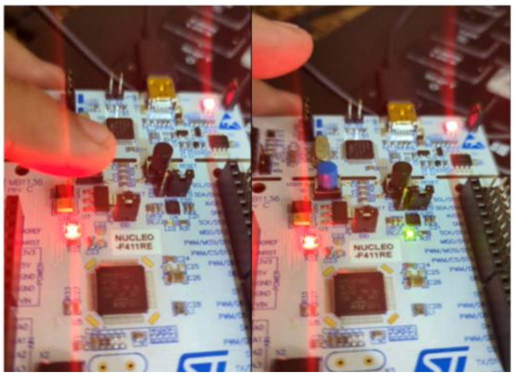

```
//where is the led connected??
//port??    A
//Pin??     5

#define PERIPH_BASE             (0x40000000UL)
#define AHB1PERIPH_OFFSET       (0x00020000UL)
#define AHB1PERIPH_BASE         (PERIPH_BASE + AHB1PERIPH_OFFSET)
#define GPIOA_OFFSET            (0x0000UL)

#define GPIOA_BASE              (AHB1PERIPH_BASE + GPIOA_OFFSET)

#define RCC_OFFSET              (0x3800UL)
#define RCC_BASE                (AHB1PERIPH_BASE + RCC_OFFSET)


#define AHB1EN_R_OFFSET         (0x30UL)
#define RCC_AHB1EN_R            (*(volatile unsigned int*)(RCC_BASE + AHB1EN_R_OFFSET))

#define MODE_R_OFFSET           (0x00UL)
#define GPIOA_MODE_R            (*(volatile unsigned int*)(GPIOA_BASE  + MODE_R_OFFSET)


#define  OD_R_OFFSET            (0x14UL)
#define GPIOA_OD_R              (*(volatile unsigned int*)(GPIOA_BASE  + OD_R_OFFSET))


#define GPIOAEN                 (1U<<0) /*  0b  0000 0000 0000 0000 0000 0000 0000 0001
```

Figure 5: STM32CUBE IDE work space

V. Experimentation

To understand register programming, set of Four different experiments are performed over the course of this project.

1.To toggle LED: In a STM32 nucleo board, led is connected at PA5 which refers to the GPIO port A and Pin 5 The User button is connected at PC13 which refers to the GPIO port C and Pin 13. The clock access is enabled for PA5 and PC13 using RCC AHB1 clock enable register, which enables clock access to AHB1 bus. The GPIO port mode register is used to set up functionality of pin. This register uses a set of two bits for each pin and these bits are used to configure the I/O direction approach. The MODER 13 is set to 00 to act as input mode and similarly MODER 5 is set to 01 to act as output mode. The GPIO BSR register is used to set or reset the LED and Bits 5 and 21 is set respectively.
As the User button being active low button, A condition is set with whenever the User button is pressed, The LED turns off and it is lifted the LED glows.



2. USART serial communication: To begin this assembly for USART Tx line and USART Rx, IO pin A2 and A3 is enabled respectively. The serial communication between the CPU and microcontroller is realized from the USART module.

To configure UART IO Pin
1)allow clock access to GPIOA
2)Set PA2 and PA3 in alternative function mode using the GPIO moder register
3)PA2 and PA3 is set to alternate function type AF07
To enable USART Tx and RX

To configure the USART module
1) Enable clock access to USART2
2)Configure the USART baud rate
3)Configure the transfer direction
4) Enable USART module

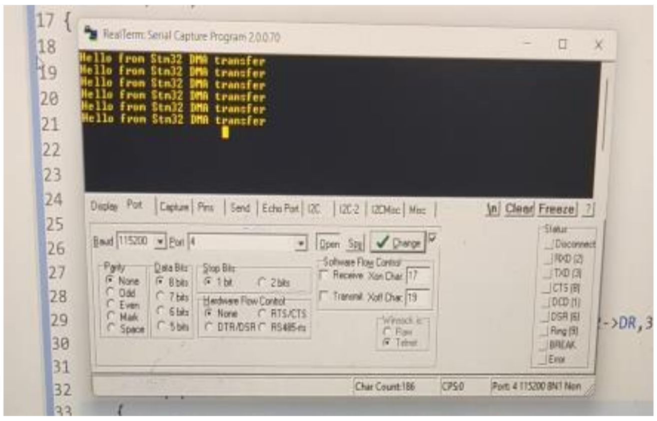
The USART peripheral unit may function in two modes: synchronous and asynchronous. When asynchronous, two lines are utilized to provide full duplex. With A2 and A3 IO pins, the Tx line is set to output and the Rx line is set to input. The USART component is undoubtedly adaptable: data length, parity mode, and baud rate are all essentially modifiable.
The baud rate is set using a 32-bit UBR (USART baud rate) register. To select the precise pace of transmission, the main clock is split by multiples of two. Furthermore, USART can induct three interrupts. When new inward data is prepared to read by USART data register, the function receive is called. When the transmission is finished, the transmission complete interrupt is called. When the buffer is empty, an interrupt is called.[5]

In STM32 IDE, USART in reception mode was tested and debugged, and any character entered from the keyboard is converted and received to its corresponding ASCII value by the microcontroller. A condition was also evaluated, such that when the input character typed is key = "1," the LED should glow, and the LED should stop blinking when the received character is not key="1." This kind of communication is far more secure and is utilized in a variety of applications, including routers, Bluetooth, and wireless communication.

3. Direct memory access (DMA) is a way of moving data between peripheral and memory without using the CPU. While the CPU is executing duties, the DMA controller is used to transport data from memory to peripheral or other device, preventing data overloading.
The STM32 nucleo-F411RE contains two DMA controllers for data transport. The experiment is set up to create a DMA driver for the USART peripheral and to deliver data via stream 6 channel 4 of the DMA controller.
Clock access to DMA 1 is first enabled using the RCC enable register, then DMA 1 is verified, and all interrupt flags for this DMA stream are cleared. The source and destination buffers are specified, followed by the message length to be transmitted. Finally, the DMA controller's transfer direction is set.

The STM32 DMA may be used in either circular or conventional mode. In circular mode, the address is written to first, then auto incremented to the next address until the last address is reached, and then returned to the first class. In normal mode, the DMA shuts itself after the final transfer.
Transmission between the peripheral and memory takes place without the CPU. During communication, the DMA controller checks the status of the source and the destination address and the amount of information of data to be sent. Then the increment in destination or source address decides the circular or normal mode of transfer.

Setting up DMA driver between memory and USART 2 peripheral and verified using Real term tool. Real term can be used for transmitting and reception of data sent by the communication protocol. It can be a excellent in logging, debugging and testing serial data. The Real term interface displayed the message "Hello from STM32 DMA transfer" sent by the USART2 protocol. [3][4]


RealTerm: Serial Capture Program 2.0.0.70

```
Hello from Stm32 DMA transfer
Hello from Stm32 DMA transfer
Hello from Stm32 DMA transfer
Hello from Stm32 DMA transfer
Hello from Stm32 DMA transfer
Hello from Stm32 DMA transfer
```

Baud 115200 Port 4
Parity: [✓ None] Odd Even Mark Space
Data Bits: [✓ 8 bits] 7 bits 6 bits 5 bits
Stop Bits: [✓ 1 bit] 2 bits
Hardware Flow Control: [✓ None] RTS/CTS DTR/DSR RS485-rts
Software Flow Control: Receive Xon Char 17; Transmit Xoff Char 19
Winsock is: Raw [✓ Telnet]
Char Count:186 CPS:0 Port: 4 115200 8N1 None

4. Analog to digital conversion (ADC):
The nucleo-STM32F411 board consists of a single 12-bit ADC. It has connections to 19 channels in which 16 of them can be used to communicate with external devices, two channels for internal, and one V_{bat} for monitoring battery voltage.

The A/D translation of the networks can be achieved in a single, continuous, scan or sporadic conversion.

The reference voltage (Vref) of ADC is 5 volts and it can measure voltage in the range between 0 to 5. The 12-bit ADC has 4096 (=2^{12}) steps and minimum voltage the ADC can detect is 1.2 milli volt (5/4096).[6]
In this experiment, we selected I/O pin A1 as our analog input and clock access to I/O is enabled using RCC_AHB1 register, Next, the I/O pin A1 is set in analog mode using MODER register.
Then to enable ADC module, the clock access is provided to ADC using RCC_APB2 register.
To select single conversion, the CONT bit is set to zero and conversion of channels is done by setting the SWSTART bit of the ADC control register.
After the conversion, the converted data is stored in ADC data register, then EOC flag is set to end the process.

For Continuous conversion mode, the next channel is converted once the previous channel conversion ends and this process is done by setting the CONT bit to one in the ADC control register. Next the converted data is stored in the ADC_DR register and EOC flag is set or interrupt is generated to end the process.

VI. Conclusion

Bare metal programming approach for ARM based microcontrollers using STM32F411RE nucleo-board is performed and analyzed using STM32 Cube IDE. Real term software is used for debugging and capture of the results. The major challenge in this project was to gather the right information from the datasheet and reference manual. Next challenge was to perform the same experiments in different board (STM32L073RZ) board. In a nutshell, I was successfully able to compose ADC drivers, USART drivers and different peripherals using embedded C language. The subsequent work would be to master the STM32 environment and build my own Real-time OS on STM32.

VII. Acknowledgement

I would like to show my appreciation to ST microelectronics company for offering valuable data sheet and reference manual which I used in my project. The assistance provided by Mr. Israel Gbati (Udemy lecturer) was greatly appreciated and reference manual which I used in my project.

VIII. References

[1] "STM32F401RE - STMicroelectronics", *STMicroelectronics*, 2022. [Online]. Available: . [https://www.st.com/en/microcontrollers-microprocessors/stm32f401re.html Accessed: 9- Feb- 2022].

[2] "datasheets STM Microelectronics", 2022. [Online]. . http://www.st.com/en/microcontrollers/stm32f446re.html

[3] [I. Gbati, 2022. [Online]. Available:https://www.udemy.com/course/embedded-systems-bare-metal-programming:. [Accessed: 9- Feb- 2022].

[4] D. Norris, *Programming with STM32*, 1st ed. McGraw-Hill Education, 2018.

[5] P. Jacko, D. Kováč, R. Bučko, T. Vince and O. Kravets, "The parallel data processing by nucleo board with STM32 microcontrollers," *2017 International Conference on Modern Electrical and Energy Systems (MEES)*, 2017, pp. 264-267, doi: 10.1109/MEES.2017.8248906

[6] Y. Bai, *Practical microcontroller engineering with ARM® technology*. Hoboken: Wiley, 2016, pp. 430-510.

[7] G. Brown, *Discovering the STM32 Microcontroller*. Indiana University, 2016.

[8] Song, J., & Meng, H. (2019). "Research and Teaching Practice on Experiment Course of STM32-Embedded Microcontroller."Proceedings of the 2019 3rd International Conference on Education, Economics and Management Research (ICEEMR 2019)

[9] W. Carvajal and W. Van Noije, "An Optimization-Based Reconfigurable Design for a 6-Bit 11-MHz Parallel Pipeline ADC with Double-Sampling S&amp;H", International Journal of Reconfigurable Computing, vol. 2012, pp. 1-17, 2012. Available: 10.1155/2012/786205.

[10] 建. 刘, "Design and Implementation of Intelligent Reading Robot Based on STM32F4", *Artificial Intelligence and Robotics Research*, vol. 09, no. 03, pp. 170-181, 2020. Available: 10.12677/airr.2020.93020.

[11] A. Agarwal, A. Anil, R. Nair and K. Sivasankaran, "ASIC Implementation of DMA Controller", *International Journal of Electrical and Electronics Research*, vol. 4, no. 1, pp. 1-4, 2016. Available: 10.37391/ijeer.040101.

[12] A. Thomas and J. Becker, "New Adaptive Multi-grained Hardware Architecture for Processing of Dynamic Function Patterns (Neue adaptive multi-granulare Hardwarearchitektur)", *it - Information Technology*, vol. 49, no. 3, pp. 165-173, 2007. Available: 10.1524/itit.2007.49.3.165.