GEBZE TECHNICAL UNIVERSITY

ELECTRONICS ENGINEERING

ELM335

Microprocessors Laboratory

LAB 1 Experiment Report

Prepared by
151024008 - Abdürrahim Deniz KUMBARACI
171024050 - Abdül Samet Karapınar
171024008 - Yasin Özbek

Introduction

It was aimed to recognize the microprocessor to be used in this experiment. And it was aimed to write led burning code for this microprocessor.

Problem 1

ICs:

U1- LDK120 (VOLTS POWER SUPPLY)

The LDK120 low drop voltage regulator provides 200 mA of maximum current from an input supply voltage in the range of 1.9 V to 5.5 V, with a typical dropout voltage of 100 mV. It is stabilized with a eramic capacitor on the output. The very low drop voltage, low quiescent current and low noise features make it suitable for low power battery-powered applications. An enable logic control function puts the LDK120 in shutdown mode allowing a total current consumption lower than 1 µA. The device also includes a short- circuit constant current limiting and thermal protection.

U2- STM32G031x8 (MicroController Unit)

The STM32G031x8 mainstream microcontrollers are based on high-performance Arm Cortex-M0+ 32-bit RISC core operating at up to 64 MHz frequency. Offering a high level of integration, they are suitable for a wide range of applications in consumer, industrial and appliance domains and ready for the Internet of Things solutions. The devices incorporate a memory protection unit **MPU**, high-speed embedded memories (8 Kbytes of SRAM and up to 64 Kbytes of Flash program memory with read protection, write protection, proprietary code protection, and securable area), DMA, an extensive range of system functions, enhanced I/Os, and peripherals. The devices offer standard communication interfaces (two I2Cs, two SPIs / one I2S, and two USARTs), one 12-bit ADC with up to 19 channels, an internal voltage reference buffer, a low-power RTC, an advanced control PWM timer running at up to double the CPU frequency, four generalpurpose 16-bit timers, a 32-bit general-purpose timer, two low-power 16-bit timers, two watchdog timers, and a SysTick timer. The devices operate within ambient temperatures from *-40 to 125°C*. They can operate with supply voltages from *1.7 V to 3.6 V*. Optimized dynamic consumption combined with a comprehensive set of power-saving modes, low-power timers and low-power **UART**, allows the design of low-power applications. VBAT direct battery input allows keeping RTC and backup registers powered. The devices come in packages with *8 to 48 pins*.

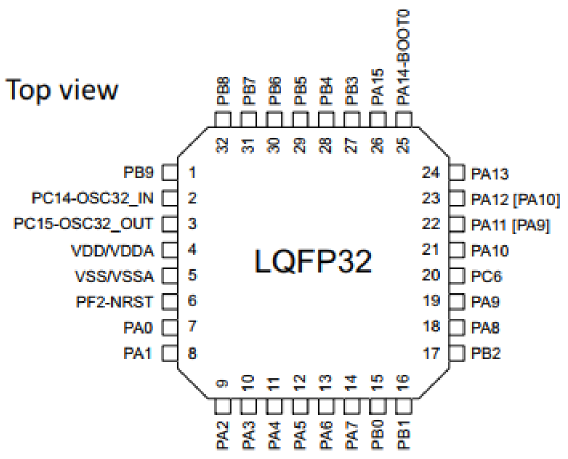


Figure 1: STM32G031KxT LQFP32 pinout

U3- LD39050 (V_{DD} POWER SOURCE)

The LD39050 provides 500 mA maximum current with an input voltage range from 1.5 V to 5.5 V and a typical dropout voltage of 200 mV. Stability is given by ceramic capacitors. The ultra low drop voltage, low quiescent current and low noise features make it suitable for low power batterypowered applications. Power supply rejection is 65 dB at low frequencies and starts to roll off at 10 kHz. The enable logic control function puts the *LD39050* in shutdown mode allowing a total current consumption lower than 1 µA. The device also includes short-circuit constant current limiting and thermal protection. Typical applications are mobile phones, hard disks and battery-powered systems.

U4- ESDALC6V1-1U2 (TVS Diode)

The ESDALC6V1-1U2 is a unidirectional single line *TVS diode (Transient Voltage Suppression)* designed to protect the data lines or other I/O ports against **ESD** transients. The device is ideal for applications where both reduced line capacitance and board space saving are required.

U5- STMPS2141STR (USB 5V POWER SWITCH)

The STMPS2141, STMPS2151, STMPS2161, STMPS2171 power distribution switches are intended for applications where heavy capacitive loads and short-circuits are likely to be encountered. These devices incorporate 90 mΩ N-channel MOSFET high-side power switches for power distribution. These switches are controlled by a logic enable input.

When the output load exceeds the current limit threshold or a short is present, the device limits the output current to a safe level by switching into a constant current mode. When continuous heavy overloads and short-circuits increase the power dissipation in the switch, causing the junction temperature to rise, a thermal protection circuit shuts the switch off to prevent damage. Recovery from a thermal shutdown is automatic once the device has cooled sufficiently. Internal circuitry ensures the switch remains off until a valid input voltage is present.

U6- USBLC6-2P6 (ST-LINK USB CONNECTOR)

The USBLC6-2P6 is a monolithic application specific device dedicated to ESD protection of high speed interfaces, such as USB 2.0, Ethernet links and video lines. The very low line capacitance secures a high level of signal integrity without compromising in protecting sensitive chips against the most stringently characterized **ESD** strikes.

U7- STM32F103xB (ST-LINK MicroController Unit)

The STM32F103xx medium-density performance line family incorporates the highperformance ARM Cortex-M3 32-bit RISC core operating at a 72 MHz frequency, highspeed embedded memories and an extensive range of enhanced I/Os and peripherals connected to two APB buses. All devices offer two 12-bit ADCs, three general purpose 16-bit timers plus one PWM timer, as well as standard and advanced communication interfaces: up to two I2Cs and SPIs, three USARTs, an USB and a CAN. The devices operate from a 2.0 to 3.6 V power supply. They are available in both the *–40 to +85 °C* temperature range and the *–40 to +105 °C* extended temperature range. A comprehensive set of power-saving mode allows the design of low-power applications. The STM32F103xx medium-density performance line family includes devices in six different package types: from *36 pins to 100 pins*. Depending on the device chosen, different sets of peripherals are included, the description below gives an overview of the complete range of peripherals proposed in this family. These features make the STM32F103xx medium-density performance line microcontroller family suitable for a wide range of applications such as motor drives, application control, medical and handheld equipment, PC and gaming peripherals, GPS platforms, industrial applications, PLCs, inverters, printers, scanners, alarm systems, video intercoms, and HVACs.

U8- LD1117S50TR (Low Drop Voltage Regulator)

The LD1117 is a low drop voltage regulator able to provide up to 800 mA of output current, available even in adjustable version (VREF = 1.25 V). Concerning fixed versions, are offered the following output voltages: 1.2 V, 1.8 V, 2.5 V, 2.85 V, 3.3 V and 5.0 V. The device is supplied in: SOT-223, DPAK, SO-8 and TO-220. The SOT-223 and DPAK surface mount packages optimize the thermal characteristics even offering a relevant space saving effect. High efficiency is assured by NPN pass transistor. In fact in this case, unlike than PNP one, the quiescent current flows mostly into the load. Only a very common 10 µF minimum capacitor is needed for stability. On chip trimming allows the regulator to reach a very tight output voltage tolerance, within ± 1 % at 25 °C. The adjustable LD1117 is pin to pin compatible with the other standard. Adjustable voltage regulators maintaining the better performances in terms of drop and tolerance.

Peripherals:

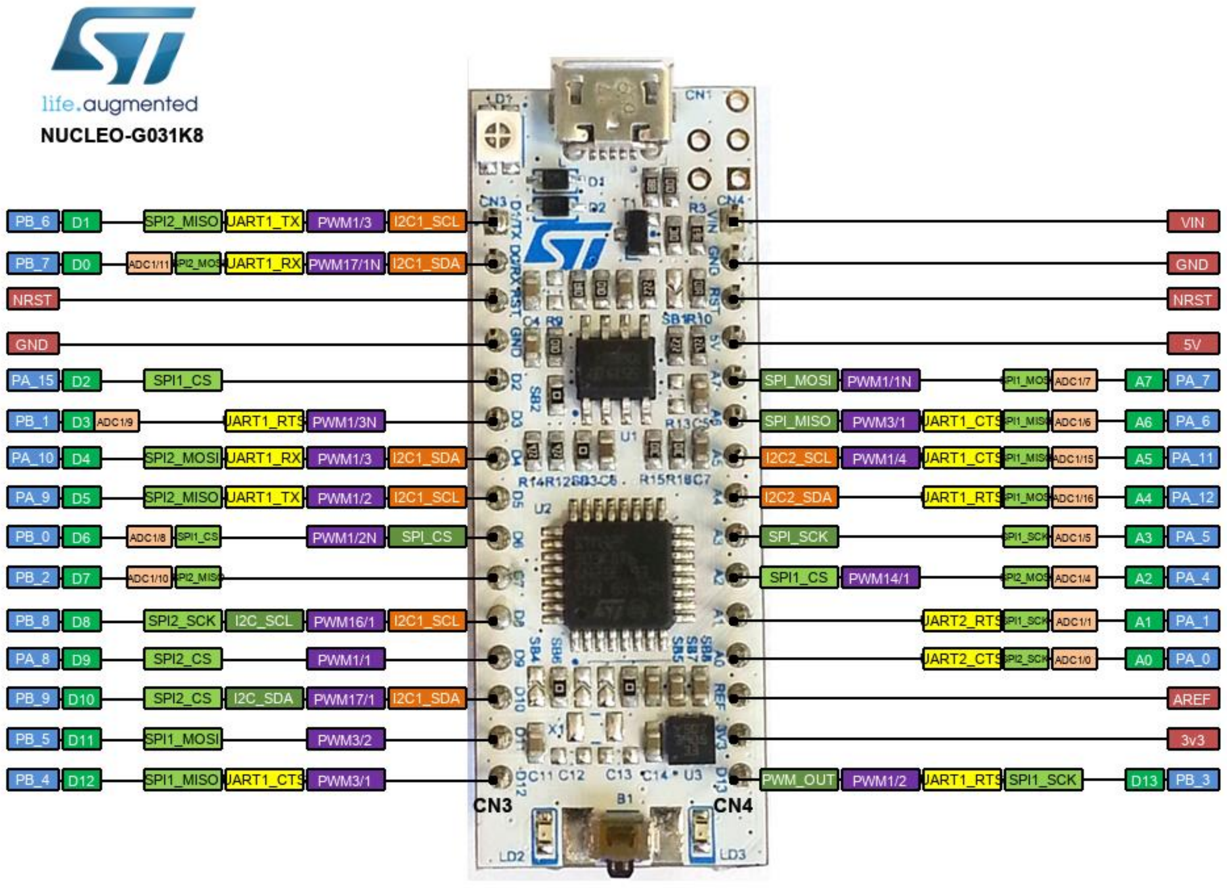

life.augmented
NUCLEO-G031K8
PB_6 D1 SPI2_MISO UART1_TX PWM1/3 I2C1_SCL
PB_7 D0 ADC1/11 SPI2_MOSI UART1_RX PWM17/1N I2C1_SDA
NRST
GND
PA_15 D2 SPI1_CS
PB_1 D3 ADC1/9 UART1_RTS PWM1/3N
PA_10 D4 SPI2_MOSI UART1_RX PWM1/3 I2C1_SDA
PA_9 D5 SPI2_MISO UART1_TX PWM1/2 I2C1_SCL
PB_0 D6 ADC1/8 SPI1_CS PWM1/2N SPI_CS
PB_2 D7 ADC1/10 SPI2_MISO
PB_8 D8 SPI2_SCK I2C_SCL PWM16/1 I2C1_SCL
PA_8 D9 SPI2_CS PWM1/1
PB_9 D10 SPI2_CS I2C_SDA PWM17/1 I2C1_SDA
PB_5 D11 SPI1_MOSI PWM3/2
PB_4 D12 SPI1_MISO UART1_CTS PWM3/1
CN3
CN4
VIN
GND
NRST
5V
SPI_MOSI PWM1/1N SPI1_MOSI ADC1/7 A7 PA_7
SPI_MISO PWM3/1 UART1_CTS SPI1_MISO ADC1/6 A6 PA_6
I2C2_SCL PWM1/4 UART1_CTS SPI1_MISO ADC1/15 A5 PA_11
I2C2_SDA UART1_RTS SPI1_MOSI ADC1/16 A4 PA_12
SPI_SCK SPI1_SCK ADC1/5 A3 PA_5
SPI1_CS PWM14/1 SPI2_MOSI ADC1/4 A2 PA_4
UART2_RTS SPI1_SCK ADC1/1 A1 PA_1
UART2_CTS SPI2_SCK ADC1/0 A0 PA_0
AREF
3v3
PWM_OUT PWM1/2 UART1_RTS SPI1_SCK D13 PB_3

Figure 2: Peripherals and Pins

It can be seen in the figure 2 that peripherals connected to which pins.

Analog to Digital Converter (ADC)

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 19 multiplexed channels allowing it to measure signals from 16 external and 3 internal sources. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 16-bit data register. The analog watchdog feature allows the application to detect if the input voltage goes outside the user-defined higher or lower thresholds. An efficient low-power mode is implemented to allow very low consumption at low frequency. A built-in hardware oversampler allows analog performances to be improved while offloading the related computational burden from the CPU.

ADC input/output pins

Pin Name	***Signal type***	***Remarks***
VDDA	*Input, analog power supply*	*Analog power supply and positive reference voltage for the ADC, VDDA ≥ VDD*
VSSA	*Input, analog supply ground*	*Ground for analog power supply. Must be at VSS potential*
VREF+	*Input, analog reference positive*	*The higher/positive reference voltage for the ADC.*
ADC_INx	*Analog input signals*	*16 external analog input channels*

Digital to Analog Converter (DAC)

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data could be left- or right-aligned. The DAC features two output channels, each with its own converter. In dual DAC channel mode, conversions could be done independently or simultaneously when both channels are grouped together for synchronous update operations. An input reference pin, VREF+ (shared with others analog peripherals) is available for better resolution. An internal reference can also be set on the same input. Refer to *voltage reference buffer (VREFBUF)* section. The DAC_OUTx pin can be used as general purpose input/output (GPIO) when the DAC output is disconnected from output pad and connected to on chip peripheral. The DAC output buffer can be optionally enabled to allow a high drive output current. An individual calibration can be applied on each DAC output channel. The DAC output channels support a low power mode, the Sample and hold mode.

DAC input/output pins

Pin name	Signal type	Remarks
VREF+	Input, analog reference positive	The higher/positive reference voltage for the DAC, VREF+ ≤ VDDAmax
VDD	Input, analog supply	Analog power supply
VSS	Input, analog supply ground	Ground for analog power supply
DAC_OUTx	Analog output signal	DAC channelx analog output

Serial Peripheral Interface (SPI)

The SPI allows synchronous, serial communication between the MCU and external devices. Application software can manage the communication by polling the status flag or using dedicated SPI interrupt.

SPI input/output pins

Pin Name	Remarks
MISO	Master In / Slave Out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode
MOSI	Master Out / Slave In data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode
SCK	Serial Clock output pin for SPI masters and input pin for SPI slaves
NSS	Slave select pin. Depending on the SPI and NSS settings

Low Power Universal Asynchronous Receiver Transmitter (LPUART)

The LPUART is an UART which enables bidirectional UART communications with a limited power consumption. Only 32.768 kHz LSE clock is required to enable UART communications up to 9600 baud/s. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock. Even when the device is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

LPUART bidirectional communications requires a minimum of two **pins**: Receive Data In **(RX)** and Transmit Data Out **(TX):**
RX (Receive Data Input) is the serial data input. **TX** (Transmit Data Output) When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In Single-wire mode, this I/O is used to transmit and receive the data.

Advanced Control Timer (TIM1)
The advanced-control timer can be seen as a three-phase PWM unit multiplexed on 6 channels. It has complementary PWM outputs with programmable inserted dead-times. It can also be seen as a complete general-purpose timer. The four independent channels can be used for input capture, output compare, PWM output (edge or center-aligned modes) with full modulation capability (0-100%), one-pulse mode output. In debug mode, the advanced-control timer counter can be frozen and the PWM outputs disabled, so as to turn off any power switches driven by these outputs.

General Purpose Timers (TIM2, 3, 14, 16, 17)
TIM2, **TIM3** are full-featured general-purpose timers: TIM2 with 32-bit auto-reload up/downcounter and 16-bit prescaler. TIM3 with 16-bit auto-reload up/downcounter and 16-bit prescaler
They have four independent channels for input capture/output compare, PWM or onepulse mode output. They can operate in combination with other general-purpose timers via the Timer Link feature for synchronization or event chaining. They can generate independent DMA request and support quadrature encoders. Their counter can befrozen in debug mode.

TIM14 is based on a 16-bit auto-reload upcounter and a 16-bit prescaler. It has one channel for input capture/output compare, PWM output or one-pulse mode output. Its counter can be frozen in debug mode.

TIM16, **TIM17** are general-purpose timers featuring: 16-bit auto-reload upcounter and 16-bit prescaler ;1 channel and 1 complementary channel
All channels can be used for input capture/output compare, PWM or one-pulse mode output. The timers can operate together via the Timer Link feature for synchronization or event chaining. They can generate independent DMA request. Their counters canbe frozen in debug mode.

Low-power timers (LPTIM1 and LPTIM2)

These timers have an independent clock. When fed with LSE, LSI or external clock, they keep running in Stop mode and they can wake up the system from it.

Independent watchdog (IWDG)

The independent watchdog is based on an 8-bit prescaler and 12-bit downcounter with user-defined refresh window. It is clocked from an independent 32 kHz internal RC (LSI). Independent of the main clock, it can operate in Stop and Standby modes. It can be used either as a watchdog to reset the device when a problem occurs, or as a free-running timer for application timeout management. It is hardware or software

SysTick timer

This timer is dedicated to real-time operating systems, but it can also be used as a standard down counter.

Features of SysTick timer: 24-bit down counter, Autoreload capability, Maskable system interrupt generation when the counter reaches 0, Prog -configurable through the option bytes. Its counter can be frozen in debug mode.

Pin	Pin name	Main feature / optional feature
1	PB9	ARD_D10: SPI1_CS(2) / TIM1_CH4
2	PC14-OSC32_IN	LSE clock input
3	PC15-OSC32_OUT	LSE clock output
4	VDD	VDD voltage supply
5	VSS	Ground ARD_A4: DC2_IN13
6	PF2-NRST	RESET – USER button
7	PA0	ARD_A0: ADC_IN0
8	PA1	ARD_A1: ADC_IN1
9	PA2	VCP_TX: USART1_Tx
10	PA3	VCP_RX: USART1_Rx
11	PA4	ARD_A2: ADC_IN4
12	PA5	ARD_A3: ADC_IN5
13	PA6	ARD_A6: ADC_IN6
14	PA7	ARD_A7: ADC_IN7
15	PB0	ARD_D6: PWM: TIM3_CH2
16	PB1	ARD_D3 - PWM: TIM3_CH4
17	PB2	ARD_D7: I/O
18	PA8	ARD_D9: PWM: TIM1_CH1
19	PA9	ARD_D5: TIM1_CH2 / I2C1_SCL
20	PC6	USER LED
21	PA10	ARD_D4: TIM1_CH3 / I2C1_SDA
22	PA11 [PA9]	ARD_A5: ADC_IN15 / I2C2_SCL
23	PA12 [PA10]	ARD_A4: ADC_IN16 / I2C2_SDA
24	PA13	SWDIO
25	PA14-BOOT0	SWCLK
26	PA15	ARD_D2: I/O
27	PB3	ARD_D13: SPI1_SCK
28	PB4	ARD_D12: SPI1_MISO
29	PB5	ARD_D11: SPI1_MOSI / TIM3_CH2
30	PB6	ARD_D0: USART1_RX
31	PB7	ARD_D1: USART1_TX
32	PB8	ARD_D8: I/O

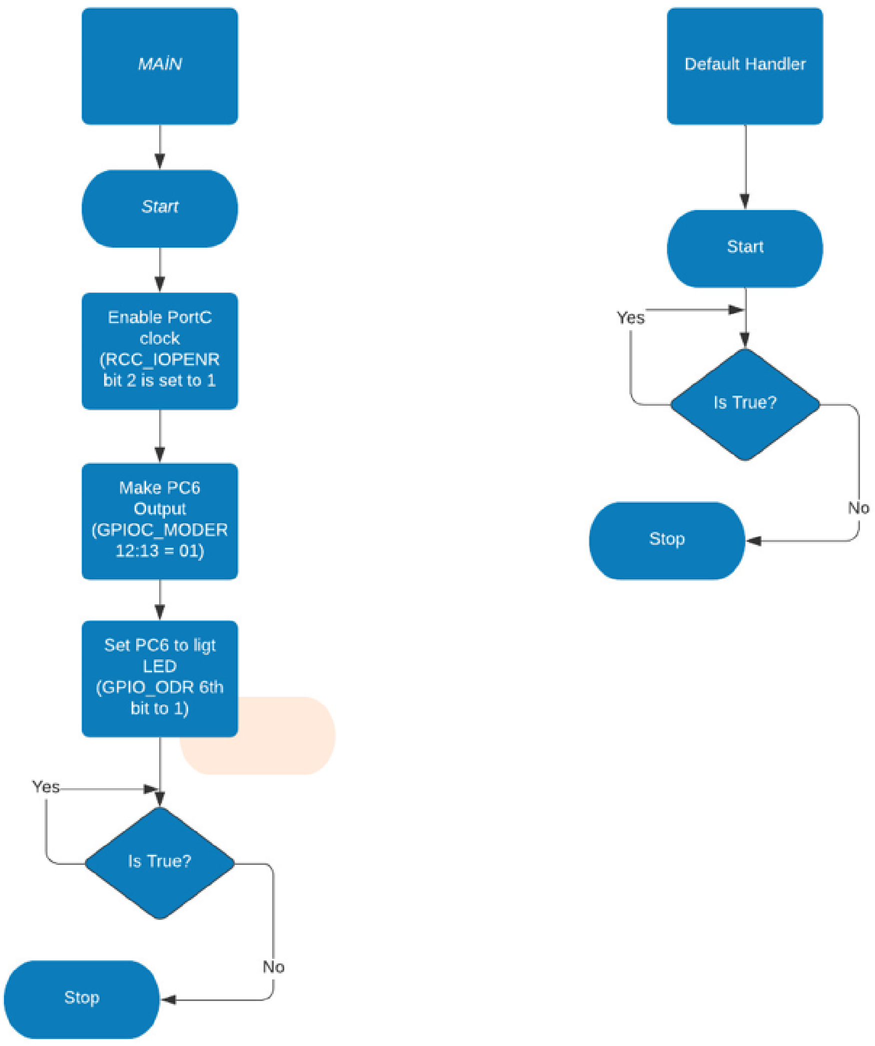
Figure 3: Nucleo-32 I/O assignment

Comments

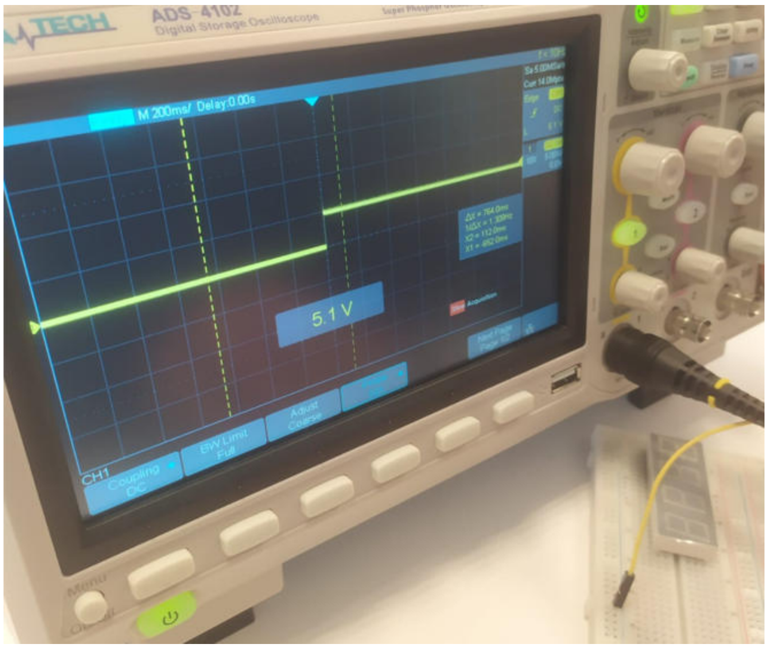
The fact that Nucleo G031K8 board has on-board debugger is particularly useful for it makes programming the M0+ particularyly easier.Also it having a regular USB connection and compatability (hardware-wise) with some other popular micro-controllers like arduino makes it versitale and good for being an entry level board

Problem 2

Flow Chart


MAIN
Start
Enable PortC clock (RCC_IOPENR bit 2 is set to 1
Make PC6 Output (GPIOC_MODER 12:13 = 01)
Set PC6 to ligt LED (GPIO_ODR 6th bit to 1)
Yes
Is True?
No
Stop
Default Handler
Start
Yes
Is True?
No
Stop

Oscilloscope Photos



Code

```
/*
* asm.s
*
* author: Furkan Cayci
* modified: Abdürrahim Deniz Kumbaracı,Yasin Özbek, Abdül Samet Karapınar

*/



.syntax unified
.cpu cortex-m0plus
.fpu softvfp
.thumb

/* make linker see this */
.global Reset_Handler

/* get these from linker script */
.word _sdata
.word _edata
.word _sbss
.word _ebss

/* define peripheral addresses from RM0444 page 57, Tables 3-4 */
.equ RCC_BASE, (0x40021000) // RCC base address
.equ RCC_IOPENR, (RCC_BASE + (0x34)) // RCC IOPENR register offset

.equ GPIOC_BASE, (0x50000800) // GPIOB base address
.equ GPIOC_MODER, (GPIOC_BASE + (0x00)) // GPIOB MODER register offset
.equ GPIOC_ODR, (GPIOC_BASE + (0x14)) // GPIOB ODR register offset

/* vector table, +1 thumb mode */
.section .vectors
vector_table:
.word _estack /* Stack pointer */
.word Reset_Handler +1 /* Reset handler */
.word Default_Handler +1 /* NMI handler */
.word Default_Handler +1 /* HardFault handler */
/* add rest of them here if needed */
 /* reset handler */
 .section .text
 Reset_Handler:
 /* set stack pointer */
 ldr r0, =_estack
 mov sp, r0
```

```
  /* initialize data and bss
 * not necessary for rom only code
 * */
 bl init_data
 /* call main */
 bl main
 /* trap if returned */
 b .

/* initialize data and bss sections */
.section .text
init_data:

 /* copy rom to ram */
ldr r0, =_sdata
ldr r1, =_edata
ldr r2, =_sidata
movs r3, #0
b LoopCopyDataInit

 CopyDataInit:
ldr r4, [r2, r3]
str r4, [r0, r3]
adds r3, r3, #4

 LoopCopyDataInit:
adds r4, r0, r3
cmp r4, r1
bcc CopyDataInit

 /* zero bss */
ldr r2, =_sbss
ldr r4, =_ebss
movs r3, #0
b LoopFillZerobss

 FillZerobss:
str r3, [r2]
adds r2, r2, #4

 LoopFillZerobss:
cmp r2, r4
bcc FillZerobss

 bx lr

/* default handler */
.section .text
Default_Handler:
b Default_Handler
```

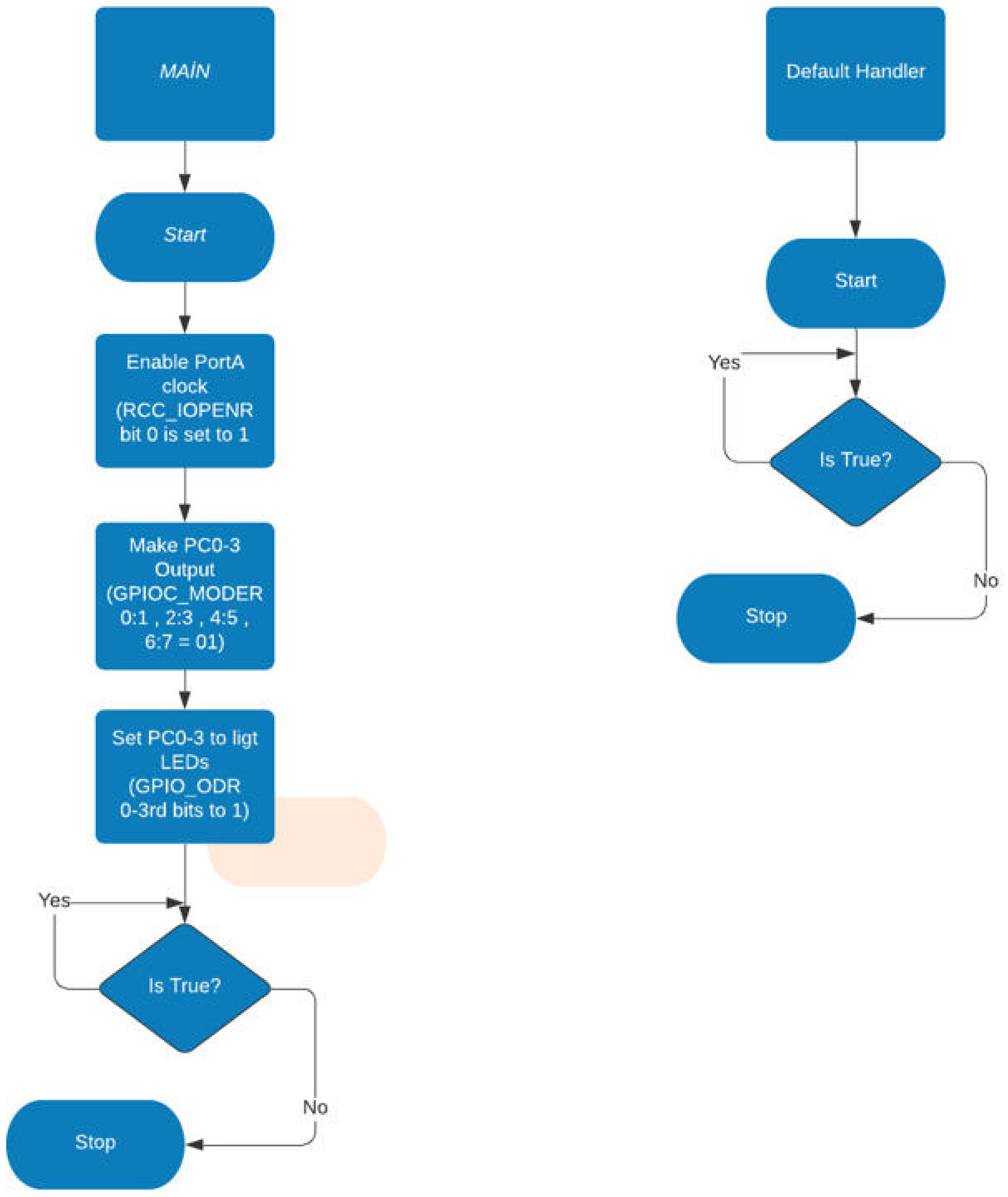
```
/* main function */
.section .text
main:
/* enable GPIOC clock, bit 3 on IOPENR */
ldr r0, =RCC_IOPENR
ldr r1, [r0]
movs r2, 0x4
orrs r1, r1, r2
str r1, [r0]

/* Now The GPIOB is clock enabled let's move on to Make GPIOC Mode 6 GPO */

 ldr r1, =GPIOC_MODER
ldr r2, [r1]
movs r3, 0x3
lsls r3, #6
bics r2, r2, r3
movs r3 , 0x1
lsls r3, #6
orrs r2, r2, r3
str r2, [r1]

/*Now we can ligt up the LEDs*/
ldr r0, =GPIOC_ODR
ldr r1, [r0]
movs r2, 0x1
lsls r2, #6
orrs r1,r1,r2



/* for(;;); */
b .

 /* this should never get executed */
nop
```

Comments

In the stm32g031k8 datasheet under 5.3.1 General operating conditions, table 2.1 , it is tated that the operating voltage range of I/O ports are min -0.3
to Max of Min(VDD + 3.6, 5.5) and the measured voltages in the lab is consistend with these findings.

Problem 3

Flow Chart



MAİN
Start
Enable PortA clock (RCC_IOPENR bit 0 is set to 1
Make PC0-3 Output (GPIOC_MODER 0:1 , 2:3 , 4:5 , 6:7 = 01)
Set PC0-3 to ligt LEDs (GPIO_ODR 0-3rd bits to 1)
Yes
Is True?
No
Stop
Default Handler
Start
Yes
Is True?
No
Stop

Code

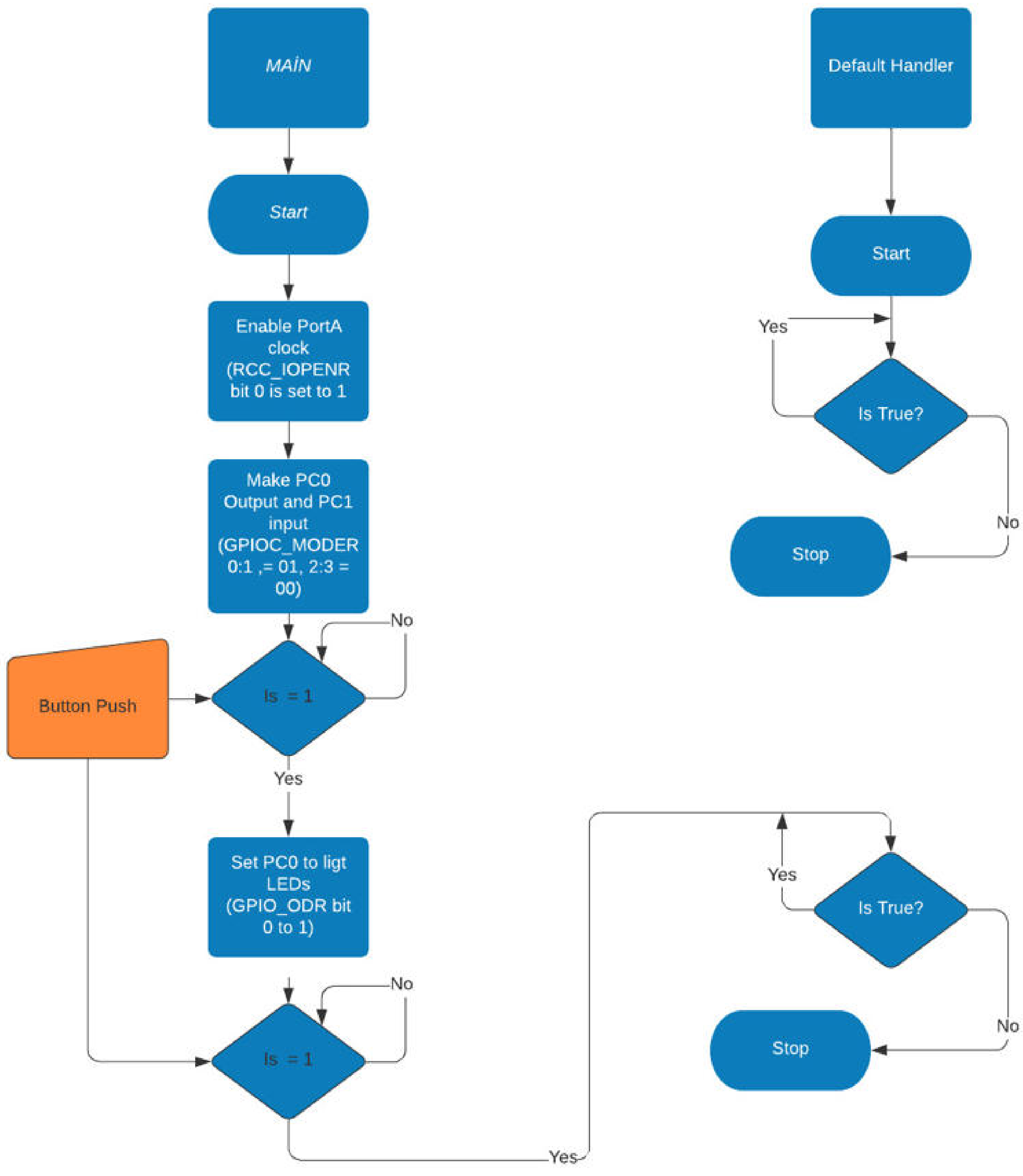
```
.syntax unified
.cpu cortex-m0plus
.fpu softvfp
.thumb
/* make linker see this */
.global Reset_Handler
/* get these from linker script */
.word _sdata
.word _edata
.word _sbss
.word _ebss
/* define peripheral addresses from RM0444 page 57, Tables 3-4 */
.equ RCC_BASE,         (0x40021000)          // RCC base address
.equ RCC_IOPENR,       (RCC_BASE   + (0x34)) // RCC IOPENR register offset
.equ GPIOA_BASE,       (0x50000000)          // GPIOC base address
.equ GPIOA_MODER,      (GPIOA_BASE + (0x00)) // GPIOC MODER register offset
.equ GPIOA_ODR,        (GPIOA_BASE + (0x14)) // GPIOC ODR register offset
/* vector table, +1 thumb mode */
.section .vectors
vector_table:
    .word _estack             /*     Stack pointer */
    .word Reset_Handler +1    /*     Reset handler */
    .word Default_Handler +1  /*       NMI handler */
    .word Default_Handler +1  /* HardFault handler */
    /* add rest of them here if needed */
/* reset handler */
.section .text
Reset_Handler:
    /* set stack pointer */
    ldr r0, =_estack
    mov sp, r0
    /* initialize data and bss
     * not necessary for rom only code
     * */
    bl init_data
    /* call main */
    bl main
    /* trap if returned */
    b .
/* initialize data and bss sections */
.section .text
  init_data:
      /* copy rom to ram */
      ldr r0, =_sdata
      ldr r1, =_edata
      ldr r2, =_sidata
      movs r3, #0
      b LoopCopyDataInit
     CopyDataInit:
            ldr r4, [r2, r3]
            str r4, [r0, r3]
            adds r3, r3, #4
    LoopCopyDataInit:
        adds r4, r0, r3
        cmp r4, r1
        bcc CopyDataInit
    /* zero bss */
    ldr r2, =_sbss
    ldr r4, =_ebss
    movs r3, #0
    b LoopFillZerobss
    FillZerobss:
        str  r3, [r2]
        adds r2, r2, #4
    LoopFillZerobss:
        cmp r2, r4
        bcc FillZerobss
    bx lr
/* default handler */
.section .text
Default_Handler:
    b Default_Handler
/* main function */
.section .text
main:
/* First We Must Enable RCC */
    ldr r0, =RCC_IOPENR
    ldr r1, [r0]
    movs r2, #1
    orrs r1,r1,r2
    str r1, [r0]
/* Now That the clock is enabled let's set pins 0-3 output mode */
    ldr r0, =GPIOA_MODER
    ldr r1, [r0]
    movs r7, #0x4 /* A counter for labels to keep track */
ClearRegister:
    movs r3, #0x3
    bics r1,r1,r3
    lsls r3, #2
    subs r7, r7, #1
    cmp r7,0
    bne ClearRegister
    movs r7, #0x4
SetRegister:
    movs r3,#0x1
    orrs r1,r1,r3
    lsls r3, #2
    subs r7, r7, #1
    cmp r7,0
    bne SetRegister
    str r1, [r0]
/*Now Let's modify ODR to make leds light up */

    ldr r0, =GPIOA_ODR
    ldr r1, [r0]
    movs r3, #0x3
    orrs r1, r1, r3
    lsls r3, #2
    orrs r1, r1, r3
    str r1, [r0]
    /* for(;;); */
    b .
```

Comments

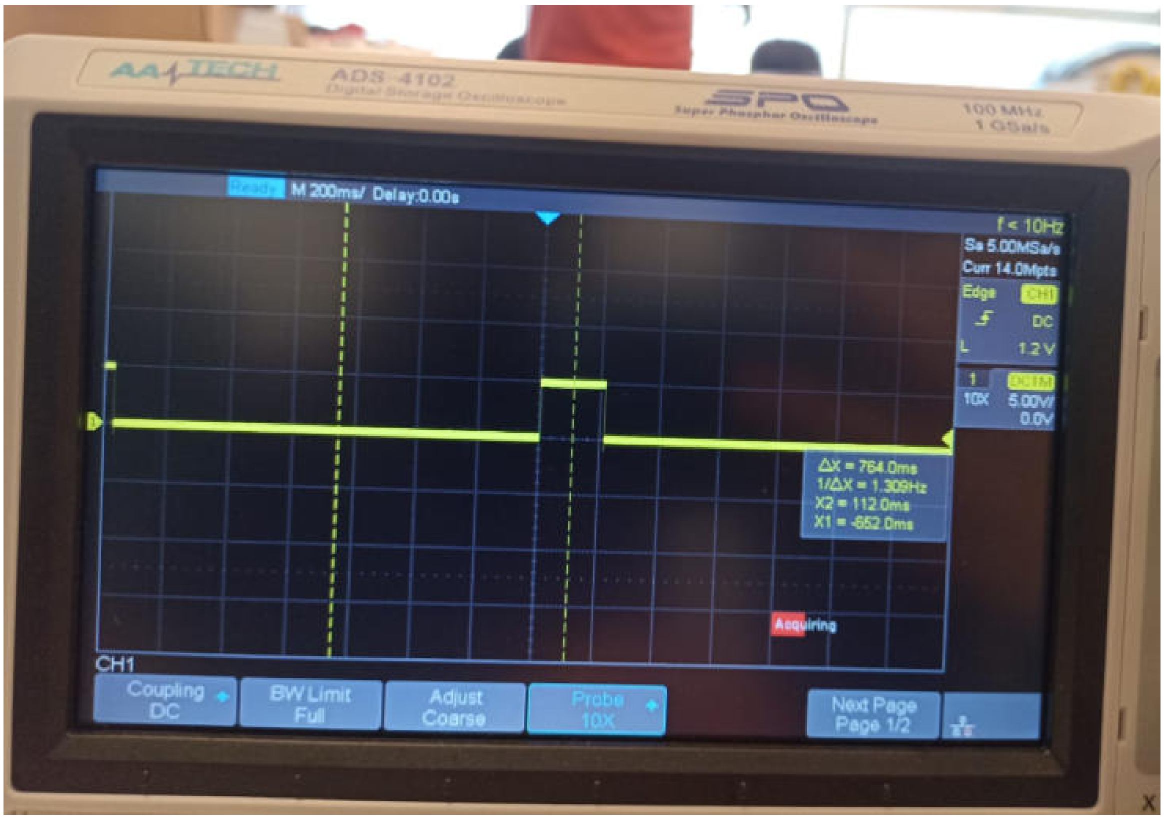
Since the reset handler points to the top of stack and sets the link register properly when the board is powered back on.We assume that the program burned to memory should execute properly in this scenerio

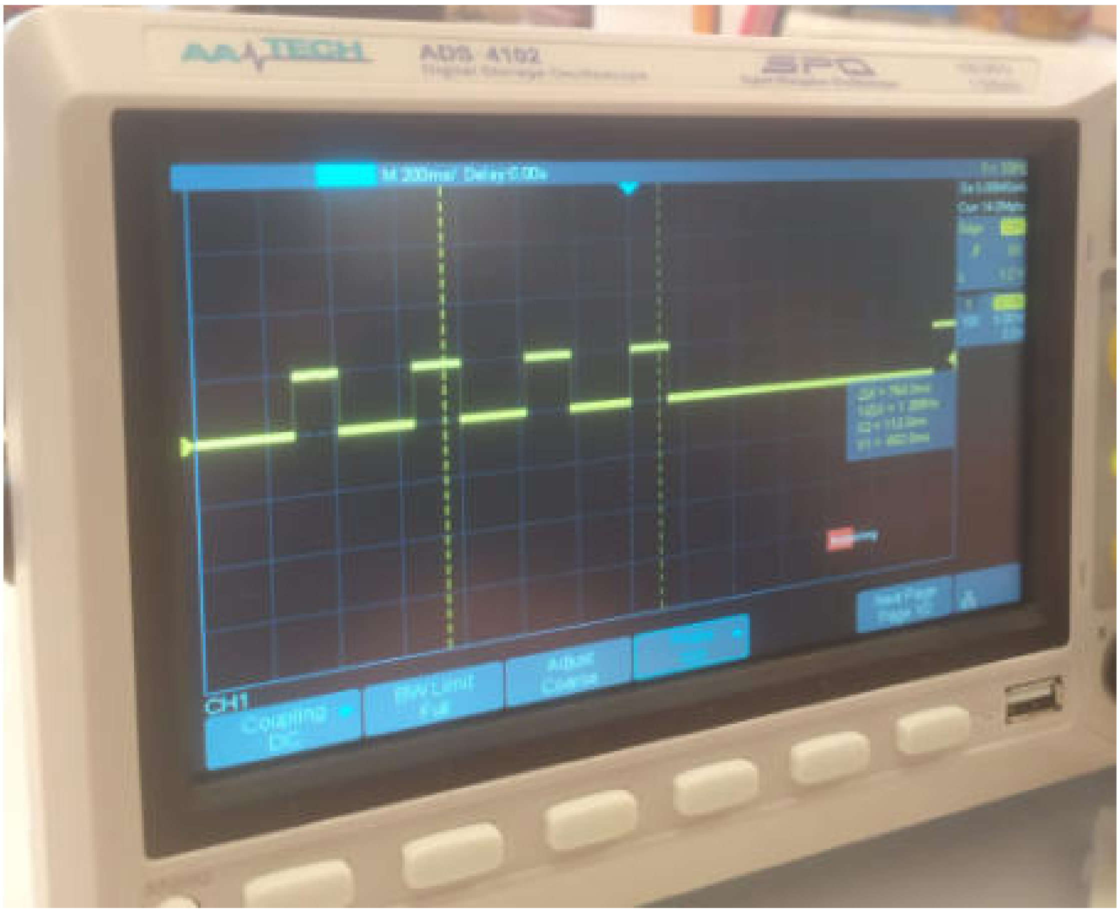
Problem 4

Flow Chart


MAİN
Start
Enable PortA clock (RCC_IOPENR bit 0 is set to 1
Make PC0 Output and PC1 input (GPIOC_MODER 0:1 ,= 01, 2:3 = 00)
No
Button Push
Is = 1
Yes
Set PC0 to ligt LEDs (GPIO_ODR bit 0 to 1)
No
Is = 1
Yes
Default Handler
Start
Yes
Is True?
No
Stop
Yes
Is True?
No
Stop

Oscilloscope Photos





When the button is pressed several times, the peak was seen as much as we pressed the oscilloscope.

Code

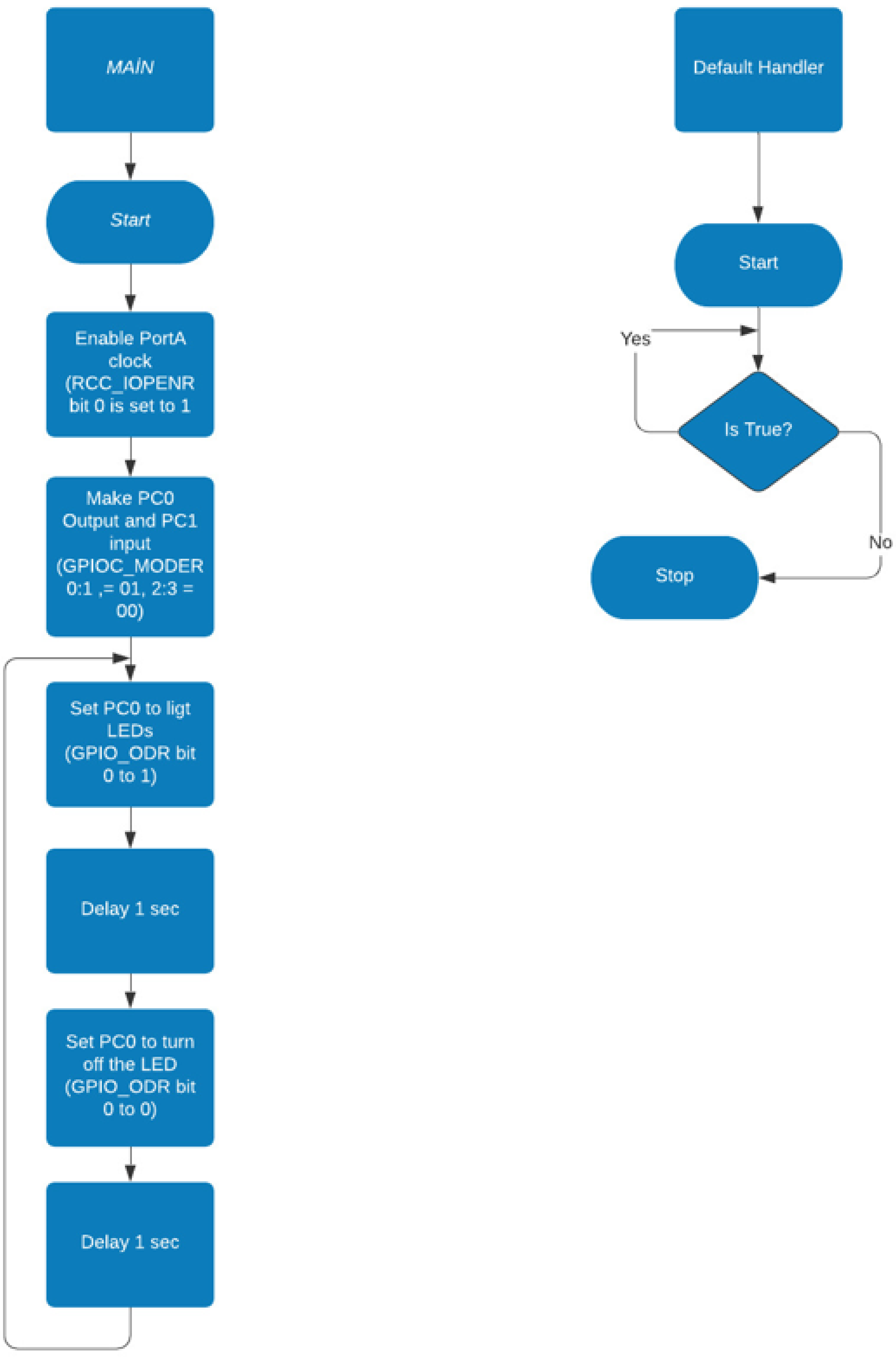
```
.syntax unified
.cpu cortex-m0plus
.fpu softvfp
.thumb
/* make linker see this */
.global Reset_Handler
/* get these from linker script */
.word _sdata
.word _edata
.word _sbss
.word _ebss
/* define peripheral addresses from RM0444 page 57, Tables 3-4 */
.equ RCC_BASE,         (0x40021000)          // RCC base address
.equ RCC_IOPENR,       (RCC_BASE   + (0x34)) // RCC IOPENR register offset
.equ GPIOA_BASE,       (0x50000000)          // GPIOB base address
.equ GPIOA_MODER,      (GPIOA_BASE + (0x00)) // GPIOB MODER register offset
.equ GPIOA_ODR,        (GPIOA_BASE + (0x14)) // GPIOB ODR register offset
.equ GPIOA_IDR,        (GPIOA_BASE + (0x10)) // GPIOB IDR register offset
/* vector table, +1 thumb mode */
.section .vectors
vector_table:
    .word _estack             /*     Stack pointer */
    .word Reset_Handler +1    /*     Reset handler */
    .word Default_Handler +1  /*       NMI handler */
    .word Default_Handler +1  /* HardFault handler */
    /* add rest of them here if needed */
/* reset handler */
.section .text
Reset_Handler:
    /* set stack pointer */
    ldr r0, =_estack
    mov sp, r0
    /* initialize data and bss
     * not necessary for rom only code
     * */
    bl init_data
    /* call main */
    bl main
    /* trap if returned */
    b .
/* initialize data and bss sections */
.section .text
init_data:
    /* copy rom to ram */
    ldr r0, =_sdata
    ldr r1, =_edata
    ldr r2, =_sidata
    movs r3, #0
    b LoopCopyDataInit
    CopyDataInit:
        ldr r4, [r2, r3]
        str r4, [r0, r3]
        adds r3, r3, #4
        LoopCopyDataInit:
             adds r4, r0, r3
             cmp r4, r1
             bcc CopyDataInit
          /* zero bss */
          ldr r2, =_sbss
          ldr r4, =_ebss
          movs r3, #0
          b LoopFillZerobss
           FillZerobss:
                  str  r3, [r2]
                  adds r2, r2, #4
              LoopFillZerobss:
                  cmp r2, r4
                  bcc FillZerobss
              bx lr
          /* default handler */
          .section .text
          Default_Handler:
              b Default_Handler
          /* main function */
          .section .text
          main:
          /* enable GPIOB clock, bit 2 on IOPENR */
              ldr r0, =RCC_IOPENR
              ldr r1, [r0]
              movs r2, 0x2
              orrs r1, r1, r2
              str r1, [r0]
          /* Now The GPIOB is clock enabled let's move on to Make GPIOB Mode 0 GPO and Mode 1 GPI  */
              ldr r0, =GPIOA_MODER
              ldr r1, [r0]
              movs r2, 0x3
              bics r1,r1,r2
              lsls r2, #2
              bics r1, r1, r2
              movs r2, 0x1
              orrs r1, r1, r2
              str r1, [r0]
           /* Now Lets Define a Function that will ligt the led when the button is pushed */
           ButtonLight:
              ldr r0, =GPIOA_IDR
              ldr r1, [r0]
              movs r2, 0x2
              ands r1, r1, r2
              cmp r1, #0
              bne ButtonOnState
              b ButtonLight
          ButtonOnState:
              ldr r0, =GPIOA_ODR
              ldr r1, [r0]
              movs r2, 0x1
              orrs r1, r1, r2
              str r1, [r0]
              b ButtonLight
              /* for(;;); */
              b .
              /* this should never get executed */
              nop
```

Comments

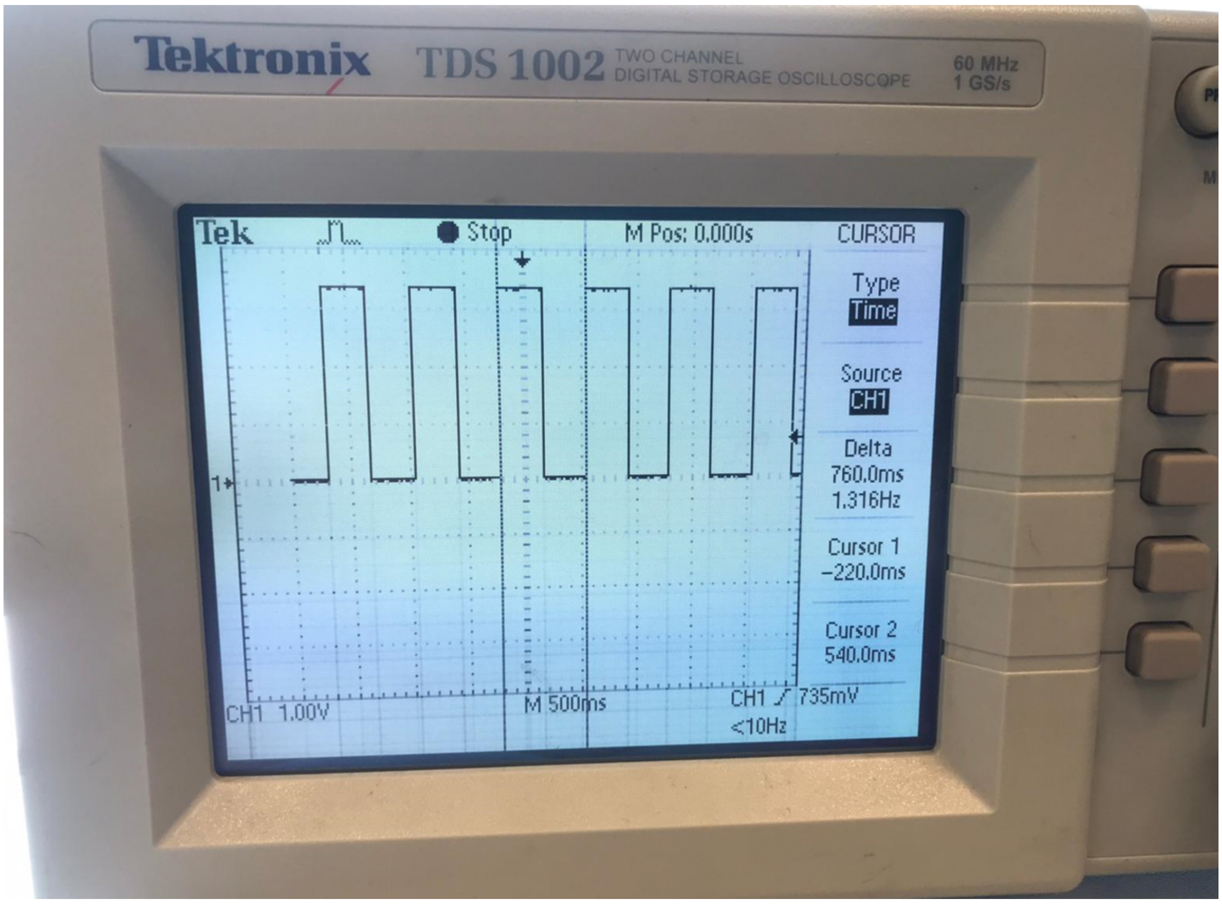
We only had 1 button to capture this and with the button we had we did not capture any significant bouncing happening that would turn off and on the LED unintentionally

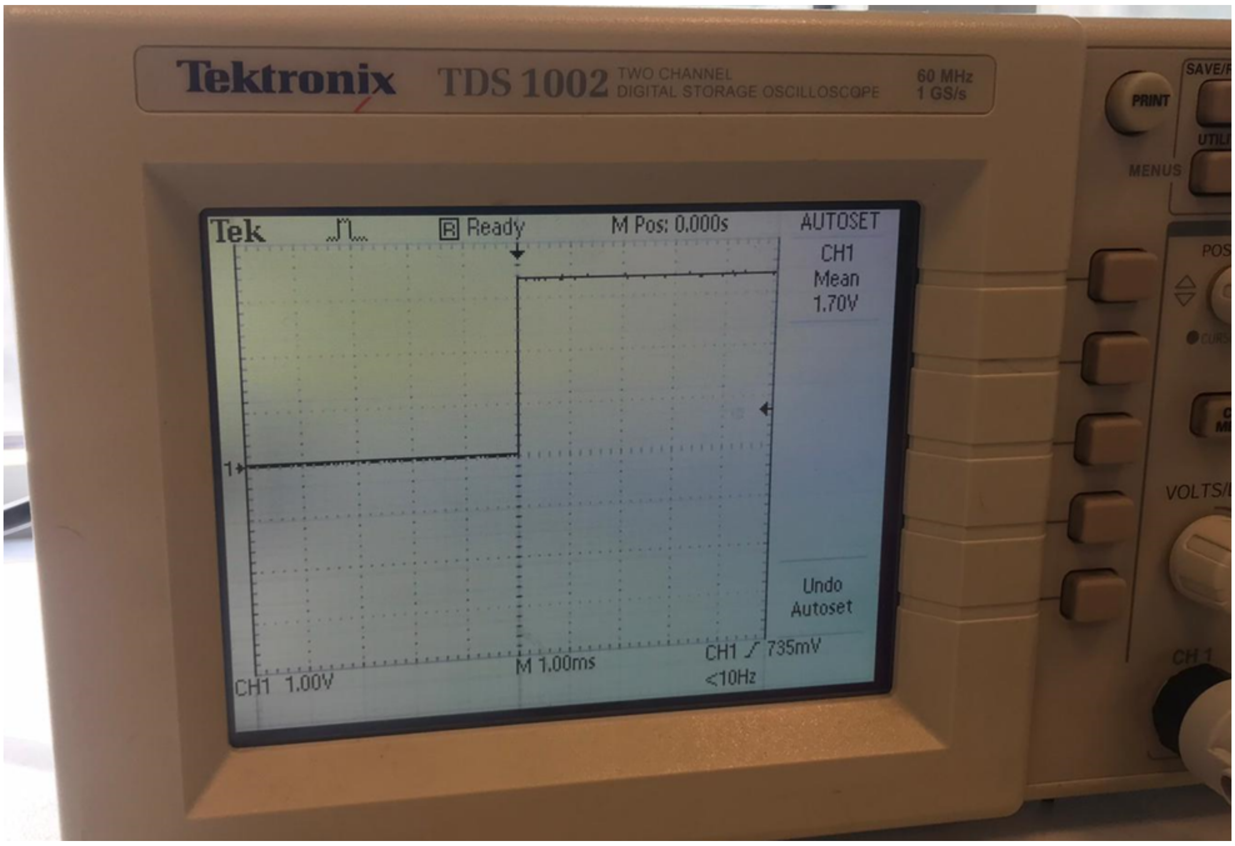
Problem 5

Flow Chart


MAİN
Start
Enable PortA clock (RCC_IOPENR bit 0 is set to 1
Make PC0 Output and PC1 input (GPIOC_MODER 0:1 ,= 01, 2:3 = 00)
Set PC0 to ligt LEDs (GPIO_ODR bit 0 to 1)
Delay 1 sec
Set PC0 to turn off the LED (GPIO_ODR bit 0 to 0)
Delay 1 sec
Default Handler
Start
Yes
Is True?
No
Stop

Oscilloscope Photos





Code

```
.syntax unified
.cpu cortex-m0plus
.fpu softvfp
.thumb
/* make linker see this */
.global Reset_Handler
/* get these from linker script */
.word _sdata
.word _edata
.word _sbss
.word _ebss
/* define peripheral addresses from RM0444 page 57, Tables 3-4 */
.equ RCC_BASE,         (0x40021000)          // RCC base address
.equ RCC_IOPENR,       (RCC_BASE   + (0x34)) // RCC IOPENR register offset
.equ GPIOA_BASE,       (0x50000000)          // GPIOB base address
.equ GPIOA_MODER,      (GPIOA_BASE + (0x00)) // GPIOB MODER register offset
.equ GPIOA_ODR,        (GPIOA_BASE + (0x14)) // GPIOB ODR register offset
/* vector table, +1 thumb mode */
.section .vectors
vector_table:
    .word _estack             /*     Stack pointer */
    .word Reset_Handler +1    /*     Reset handler */
    .word Default_Handler +1  /*       NMI handler */
    .word Default_Handler +1  /* HardFault handler */
    /* add rest of them here if needed */
/* reset handler */
.section .text
Reset_Handler:
    /* set stack pointer */
    ldr r0, =_estack
    mov sp, r0
    /* initialize data and bss
     * not necessary for rom only code
     * */
    bl init_data
    /* call main */
    bl main
    /* trap if returned */
    b .
/* initialize data and bss sections */
.section .text
init_data:
    /* copy rom to ram */
    ldr r0, =_sdata
    ldr r1, =_edata
    ldr r2, =_sidata
    movs r3, #0
    b LoopCopyDataInit
    CopyDataInit:
        ldr r4, [r2, r3]
        str r4, [r0, r3]
        adds r3, r3, #4
    LoopCopyDataInit:
        adds r4, r0, r3
        cmp r4, r1
        bcc CopyDataInit
```

```
   /* zero bss */
    ldr r2, =_sbss
    ldr r4, =_ebss
    movs r3, #0
    b LoopFillZerobss
    FillZerobss:
        str  r3, [r2]
        adds r2, r2, #4
    LoopFillZerobss:
        cmp r2, r4
        bcc FillZerobss
    bx lr
/* default handler */
.section .text
Default_Handler:
    b Default_Handler
/* main function */
.section .text
main:
    /* enable GPIOB clock, bit 2 on IOPENR */
    ldr r0, =RCC_IOPENR
    ldr r1, [r0]
    movs r2, 0x1
    orrs r1, r1, r2
    str r1, [r0]
    /* Let's Modify The MODER data to make MODE 0 into output */
    ldr r0, =GPIOA_MODER
    ldr r1, [r0]
    movs r2, 0x3
    bics r1, r1, r2
    movs r2, 0x1
    orrs r1, r1, r2
    str r1, [r0]
    /*Now that the Mode 6 is in GPO Lets Define a Loop to Blink the LED */
LedBlink:
    ldr r0, =GPIOA_ODR
    ldr r1, [r0]
    movs r2, 0x1
    orrs r1,r1,r2
    str r1, [r0]
    ldr r7, =#2000000
    bl delay
    ldr r0, =GPIOA_ODR
    ldr r1, [r0]
    movs r2, 0x1
    bics r1,r1,r2
    str r1, [r0]
    ldr r7, =#2000000
    bl delay
    b LedBlink
delay:
    subs r7,r7,#1
    bne delay
    bx lr
    /* for(;;); */
    b .
    /* this should never get executed */
    nop
```

Comments

In the lab we measured the interval to be aroun .7 secs for our code wich had around 2M instructions.But i dont know how to calculate the CPI

Conclusion

In conclusion in this lab we got ourselves familiarised with the assembly language in thumb mode and basic instructions of it.Also we had hands-on experience with measuring the voltages internal and external to the board.With simple exercises we prepared ourselves for more complex applications such as 7-segment display in later weeks.